# Manual do DestinyPatch em português

Documentação completa do DestinyPatch — a extensão do RPG Maker 2000/2003 que adiciona a linguagem DestinyScript (ponto flutuante, imagens, rede, arquivos, acesso às funções do RPG Maker e muito mais) — traduzida e reorganizada.

Manual completo · 915 tópicos · 778 páginas · gerado em 13/10/2026

Tradução para o português do manual alemão do DestinyPatch (autor: Bananen-Joe). O DestinyPatch não é uma extensão oficial da Enterbrain. Conteúdo traduzido e reorganizado; consulte a página "Informações legais (Impressum)" no final do manual.

# Sumário

**Primeiros passos** 19
**Introdução** 20
**Guia rápido** 22
**DestinyPatcher** 23
**DestinyPatcher** 24
Menu do DestinyPatcher 25
Categorias do DestinyPatcher 26
Categoria do Patcher: DestinyDLL 26
Categoria do Patcher: Tela de título 28
Categoria do Patcher: Janela 29
Categoria do Patcher: Imagens 31
Categoria do Patcher: Teclado 32
Categoria do Patcher: EnterHeroName 34
Categoria do Patcher: Debug 35
Categoria do Patcher: Blitting 36
Categoria do Patcher: Strings 38
Categoria do Patcher: Música 39
**Linguagem DestinyScript** 41
**Estrutura da linguagem de script** 42
Comentários 45
Tipos de dados e áreas de memória 46
Números 48
Strings 50
Operadores 52
Operadores de atribuição 52
Operadores aritméticos 53
Operadores de string 54
Operadores binários 54
Operadores lógicos 54
Operadores de comparação 55
Operadores de objeto 56
Sinais (positivo e negativo) 56
Expressões de fórmula 58
Elementos de estrutura 59
Ramificações 59
Ramificação simples (If) 60
Ramificação múltipla (Switch) 62
Laços de repetição 65
Laços com teste no início ou no fim 65
Laços de contagem (For) 67
Comandos de salto 70
Comandos de salto para laços e ramificações 71
Outros comandos de salto 71
**Referência** 74
**Lista de objetos** 75
Objeto: a 76
Propriedade: a.Count 77
Método: a.ReAlloc() 78
Propriedade: a[] 79

Objeto: Actor 80
Propriedade: Actor.Count 80
Objeto: Actor[] 81
Categoria: Equipamento 82
Propriedade: Actor[].Accessory 82
Propriedade: Actor[].Armor 83
Propriedade: Actor[].Helmet 84
Propriedade: Actor[].Shield 84
Propriedade: Actor[].Weapon 85
Categoria: Aparência 86
Propriedade: Actor[].Charframe 86
Propriedade: Actor[].Charset 87
Propriedade: Actor[].Faceframe 88
Propriedade: Actor[].Faceset 89
Propriedade: Actor[].Transparency 89
Objeto: Actor[].Default 90
Propriedade: Actor[].Default.Accessory 91
Propriedade: Actor[].Default.Armor 91
Propriedade: Actor[].Default.Charframe 92
Propriedade: Actor[].Default.Charset 92
Propriedade: Actor[].Default.Degree 93
Propriedade: Actor[].Default.Faceframe 93
Propriedade: Actor[].Default.Faceset 93
Propriedade: Actor[].Default.Helmet 94
Propriedade: Actor[].Default.Level 94
Propriedade: Actor[].Default.Name 95
Propriedade: Actor[].Default.Shield 95
Propriedade: Actor[].Default.Transparency 95
Propriedade: Actor[].Default.Weapon 96
Objeto: Actor[].LevelAttribute 96
Propriedade: Actor[].LevelAttribute.Count 97
Objeto: Actor[].LevelAttribute[] 97
Propriedade: Actor[].LevelAttribute[].Agility 98
Propriedade: Actor[].LevelAttribute[].Attack 98
Propriedade: Actor[].LevelAttribute[].Defense 99
Objeto: Actor[].LevelAttribute[].LearnableSkill 99
Propriedade: Actor[].LevelAttribute[].LearnableSkill.Count 100
Propriedade: Actor[].LevelAttribute[].LearnableSkill[] 100
Propriedade: Actor[].LevelAttribute[].MaxHP 101
Propriedade: Actor[].LevelAttribute[].MaxMP 102
Propriedade: Actor[].LevelAttribute[].Mind 102
Categoria: Nomes 102
Propriedade: Actor[].Degree 103
Propriedade: Actor[].Name 104
Categoria: Outras propriedades 104
Propriedade: Actor[].AttributeEffect[] 105
Propriedade: Actor[].Condition[] 106
Propriedade: Actor[].ConditionEffect[] 107
Propriedade: Actor[].ControlledByAI 108
Propriedade: Actor[].CriticalHitChance 109
Propriedade: Actor[].FixedEquipment 110
Método: Actor[].HasCondition() 110
Método: Actor[].HasSkill() 111
Propriedade: Actor[].MaxLevel 112
Propriedade: Actor[].Skill[] 113
Propriedade: Actor[].StrongDefense 114

Propriedade: Actor[].TwoSwordStyle 114
Propriedade: Actor[].UnarmedBattleAnimation 115
Propriedade: Actor[].UseSkillName 116
Objeto: Actor[].AttributeEffect 116
Propriedade: Actor[].AttributeEffect.Count 116
Objeto: Actor[].Condition 117
Propriedade: Actor[].Condition.Count 117
Objeto: Actor[].ConditionEffect 118
Propriedade: Actor[].ConditionEffect.Count 118
Objeto: Actor[].Skill 119
Propriedade: Actor[].Skill.Count 119
Categoria: Valores de status 120
Propriedade: Actor[].AgilityDiff 120
Propriedade: Actor[].AttackDiff 121
Propriedade: Actor[].DefenseDiff 122
Propriedade: Actor[].EXP 123
Propriedade: Actor[].HP 123
Propriedade: Actor[].Level 124
Propriedade: Actor[].MaxHPDiff 125
Propriedade: Actor[].MaxMPDiff 125
Propriedade: Actor[].MindDiff 126
Propriedade: Actor[].MP 127
Objeto: Attribute 128
Propriedade: Attribute.Count 128
Objeto: Attribute[] 129
Propriedade: Attribute[].Name 129
Objeto: Attribute[].Rate 130
Propriedade: Attribute[].Rate.Count 130
Propriedade: Attribute[].Rate[] 131
Propriedade: Attribute[].Type 131
Objeto: Client 132
Método: Client.CloseAll() 133
Propriedade: Client.Count 133
Método: Client.NextFree() 134
Método: Client.ReAlloc() 135
Objeto: Client[] 136
Categoria: Propriedades gerais 137
Método: Client[].Close() 138
Método: Client[].Connect() 139
Método: Client[].GetRecvLength() 141
Propriedade: Client[].State 142
Propriedade: Client[].Type 144
Objeto: Client[].Destiny 145
Método: Client[].Destiny.GetRecvType() 146
Método: Client[].Destiny.RecvByte() 148
Método: Client[].Destiny.RecvDword() 151
Método: Client[].Destiny.RecvFloat() 153
Método: Client[].Destiny.RecvId() 155
Método: Client[].Destiny.RecvString() 158
Método: Client[].Destiny.RecvSwitch() 160
Método: Client[].Destiny.RecvVariable() 162
Método: Client[].Destiny.RecvWord() 165
Método: Client[].Destiny.SendByte() 167
Método: Client[].Destiny.SendDword() 170
Método: Client[].Destiny.SendFloat() 172
Método: Client[].Destiny.SendString() 175

Método: Client[].Destiny.SendSwitch() 177
Método: Client[].Destiny.SendVariable() 180
Método: Client[].Destiny.SendWord() 182
Objeto: Client[].Raw 185
Método: Client[].Raw.Recv() 185
Método: Client[].Raw.Send() 187
Categoria: Métodos obsoletos 189
Método: Client[].GetRecvType() 190
Método: Client[].RecvByte() 190
Método: Client[].RecvDouble() 191
Método: Client[].RecvDword() 191
Método: Client[].RecvId() 191
Método: Client[].RecvRawData() 192
Método: Client[].RecvString() 192
Método: Client[].RecvSwitch() 192
Método: Client[].RecvVariable() 193
Método: Client[].RecvWord() 193
Método: Client[].SendByte() 193
Método: Client[].SendDouble() 194
Método: Client[].SendDword() 194
Método: Client[].SendRawData() 195
Método: Client[].SendString() 195
Método: Client[].SendSwitch() 195
Método: Client[].SendVariable() 196
Método: Client[].SendWord() 196
Categoria: Dados de conexão 196
Propriedade: Client[].LocalIP 197
Propriedade: Client[].LocalPort 198
Propriedade: Client[].RemoteIP 199
Propriedade: Client[].RemotePort 200
Objeto: Command 201
Método: Command.AddAbility() 204
Método: Command.AddActor() 205
Método: Command.AddCondition() 206
Método: Command.AddExp() 207
Método: Command.AddGold() 208
Método: Command.AddHp() 209
Método: Command.AddItem() 210
Método: Command.AddLevel() 211
Método: Command.AddMp() 212
Método: Command.AddSkill() 213
Método: Command.AddTeleportPlace() 214
Método: Command.CallCommonEvent() 215
Método: Command.CallEvent() 216
Método: Command.CallMenu() 217
Método: Command.CallSaveMenu() 217
Método: Command.ChangeSwitch() 218
Método: Command.ChangeSwitchRange() 219
Método: Command.ChangeVariable() 220
Método: Command.ChangeVariableRange() 222
Método: Command.ClearTimer() 225
Método: Command.DealDamage() 225
Método: Command.DropEquipment() 227
Método: Command.EnableEscape() 228
Método: Command.EnableMenu() 229
Método: Command.EnableSaveMenu() 229

Método: Command.EnableTeleport() 230
Método: Command.EnterHeroName() 231
Método: Command.EnterKey() 232
Método: Command.EnterLeaveVehicle() 234
Método: Command.EquipItem() 234
Método: Command.ErasePicture() 235
Método: Command.EraseScreen() 236
Método: Command.FadeOutMusic() 237
Método: Command.FixScreen() 238
Método: Command.FlashEvent() 239
Método: Command.FlashScreen() 241
Método: Command.FullRecovery() 243
Método: Command.GameOver() 243
Método: Command.GetEventId() 244
Método: Command.GetTerrainId() 245
Método: Command.GoToLabel() 246
Método: Command.GoToMemorizedPlace() 247
Método: Command.MemorizeMusic() 248
Método: Command.MemorizePlace() 249
Método: Command.MoveAll() 250
Método: Command.MoveEvent() 251
Método: Command.MovePicture() 253
Método: Command.PlayMemorizedMusic() 256
Método: Command.PlayMovie() 256
Método: Command.PlayMusic() 258
Método: Command.PlaySound() 259
Método: Command.ReleaseScreen() 260
Método: Command.RemoveActor() 261
Método: Command.RemoveCondition() 261
Método: Command.RemoveItem() 262
Método: Command.RemoveSkill() 263
Método: Command.RemoveTeleportPlace() 264
Método: Command.ReplaceLowerChip() 265
Método: Command.ReplaceUpperChip() 266
Método: Command.ReturnScreen() 267
Método: Command.SetChipset() 269
Método: Command.SetEnemyEncounter() 269
Método: Command.SetEscapePlace() 270
Método: Command.SetEventPlace() 271
Método: Command.SetHeroChar() 272
Método: Command.SetHeroDegree() 273
Método: Command.SetHeroFace() 274
Método: Command.SetHeroName() 275
Método: Command.SetHeroTransparency() 276
Método: Command.SetMessageStyle() 277
Método: Command.SetPanorama() 278
Método: Command.SetScreenTone() 280
Método: Command.SetScreenTransition() 282
Método: Command.SetSystemGraphic() 284
Método: Command.SetSystemMusic() 285
Método: Command.SetSystemSound() 286
Método: Command.SetTimer() 288
Método: Command.SetVehicleChar() 289
Método: Command.SetVehiclePlace() 290
Método: Command.SetWeather() 291
Método: Command.ShakeScreen() 292

Método: Command.ShiftScreen() 294
Método: Command.ShowBattleAnimation() 295
Método: Command.ShowFace() 296
Método: Command.ShowPicture() 298
Método: Command.ShowScreen() 300
Método: Command.StartTimer() 302
Método: Command.StopAll() 303
Método: Command.StopMusic() 303
Método: Command.StopParallelProcesses() 304
Método: Command.StopTimer() 304
Método: Command.SubtractAbility() 305
Método: Command.SubtractExp() 306
Método: Command.SubtractGold() 307
Método: Command.SubtractHp() 308
Método: Command.SubtractLevel() 309
Método: Command.SubtractMp() 310
Método: Command.SwapTwoEvents() 311
Método: Command.Teleport() 312
Método: Command.TitleScreen() 313
Método: Command.Wait() 313
Objeto: CommonEvent 314
Propriedade: CommonEvent.Count 315
Objeto: CommonEvent[] 315
Propriedade: CommonEvent[].CallMode 316
Propriedade: CommonEvent[].Id 317
Propriedade: CommonEvent[].Name 317
Propriedade: CommonEvent[].Switch 318
Objeto: Condition 319
Propriedade: Condition.Count 319
Objeto: Condition[] 320
Propriedade: Condition[].ActionLimitation 322
Propriedade: Condition[].AllyStateMessage 322
Propriedade: Condition[].AlreadyInStateMessage 322
Objeto: Condition[].Chance 323
Propriedade: Condition[].Chance.Count 323
Propriedade: Condition[].Chance[] 324
Propriedade: Condition[].Color 324
Propriedade: Condition[].EnemyStateMessage 325
Propriedade: Condition[].HalfAgility 325
Propriedade: Condition[].HalfAttack 326
Propriedade: Condition[].HalfDefense 326
Propriedade: Condition[].HalfMind 327
Propriedade: Condition[].HitChanceLimitation 327
Propriedade: Condition[].HitRateChange 328
Propriedade: Condition[].MindChanceLimitation 328
Propriedade: Condition[].Name 328
Propriedade: Condition[].NaturalHealingChance 329
Propriedade: Condition[].NaturalHealingTurns 330
Propriedade: Condition[].PercentDownPerTurnHp 330
Propriedade: Condition[].PercentDownPerTurnMp 331
Propriedade: Condition[].PointDownPerStepHp 331
Propriedade: Condition[].PointDownPerStepMp 331
Propriedade: Condition[].PointDownPerTurnHp 332
Propriedade: Condition[].PointDownPerTurnMp 332
Propriedade: Condition[].Priority 333
Propriedade: Condition[].RecoveryStateMessage 333

Propriedade: Condition[].ShockHealingChance 333
Propriedade: Condition[].StepsDownHp 334
Propriedade: Condition[].StepsDownMp 334
Propriedade: Condition[].StillInStateMessage 335
Propriedade: Condition[].Type 335
Objeto: Convert 336
Método: Convert.Angle() 336
Método: Convert.Bool() 338
Método: Convert.Byte() 339
Propriedade: Convert.DecimalComma 340
Método: Convert.Dword() 340
Método: Convert.Float() 341
Método: Convert.String() 342
Método: Convert.Word() 343
Objeto: d 344
Propriedade: d.Count 345
Método: d.ReAlloc() 345
Propriedade: d[] 346
Objeto: Destiny 347
Objeto: Destiny.Debugger 348
Método: Destiny.Debugger.Open() 348
Objeto: Destiny.Debugger.Output 349
Método: Destiny.Debugger.Output.Clear() 349
Método: Destiny.Debugger.Output.Print() 350
Método: Destiny.Debugger.Output.Show() 351
Propriedade: Destiny.Debugger.Output.Text 351
Propriedade: Destiny.DllVersionMajor 352
Propriedade: Destiny.DllVersionMinor 353
Método: Destiny.Implemented() 353
Propriedade: Destiny.Language 354
Método: Destiny.Load() 355
Método: Destiny.Save() 356
Propriedade: Destiny.VersionMajor 358
Propriedade: Destiny.VersionMinor 358
Objeto: Directory 359
Categoria: Atributos de arquivos/diretórios 359
Método: Directory.GetAttributes() 360
Método: Directory.SetAttributes() 361
Categoria: Operações com arquivos/diretórios 362
Método: Directory.CopyFile() 363
Método: Directory.CreateDir() 364
Método: Directory.DeleteFile() 364
Método: Directory.RemoveDir() 365
Método: Directory.Rename() 366
Categoria: Buscas em diretórios 367
Método: Directory.FindClose() 368
Método: Directory.FindCloseAll() 370
Método: Directory.FindFirst() 371
Propriedade: Directory.FindLevel 374
Método: Directory.FindNext() 375
Objeto: Error 378
Propriedade: Error.Count 378
Método: Error.DisableAll() 379
Método: Error.EnableAll() 379
Objeto: Error[] 380
Propriedade: Error[].Enabled 381

Propriedade: Error[].Message 382
Método: Error[].Raise() 383
Propriedade: Error[].Title 384
Objeto: Errors 385
Método: Errors.Catch() 385
Método: Errors.Disable() 386
Método: Errors.Enable() 387
Método: Errors.Halt() 388
Método: Errors.Resume() 389
Objeto: Event 390
Propriedade: Event.Count 390
Objeto: Event[] 391
Objeto: Event[].AuroraSheet 393
Método: Event[].CanMoveTo() 393
Propriedade: Event[].Charframe 394
Propriedade: Event[].Charset 395
Propriedade: Event[].DirFlags 395
Propriedade: Event[].DirLook 396
Propriedade: Event[].DirMove 397
Propriedade: Event[].Exists 399
Propriedade: Event[].FixDir 399
Propriedade: Event[].Frame 400
Propriedade: Event[].FrameSimple 401
Propriedade: Event[].Frequency 402
Propriedade: Event[].Id 402
Propriedade: Event[].JumpTime 403
Propriedade: Event[].MapId 403
Objeto: Event[].MoveRoute 404
Propriedade: Event[].MoveRoute.Active 405
Propriedade: Event[].MoveRoute.Count 405
Propriedade: Event[].MoveRoute.CurrentIndex 406
Propriedade: Event[].MoveRoute.IgnoreIfCantMove 407
Propriedade: Event[].MoveRoute.Repeat 407
Objeto: Event[].MoveRoute[] 408
Propriedade: Event[].MoveRoute[].Charframe 409
Propriedade: Event[].MoveRoute[].Charset 410
Propriedade: Event[].MoveRoute[].Command 410
Propriedade: Event[].MoveRoute[].Panpot 412
Propriedade: Event[].MoveRoute[].Sound 413
Propriedade: Event[].MoveRoute[].Switch 414
Propriedade: Event[].MoveRoute[].Tempo 415
Propriedade: Event[].MoveRoute[].Volume 415
Propriedade: Event[].Name 416
Propriedade: Event[].Offset 417
Propriedade: Event[].Phasing 418
Propriedade: Event[].ScreenX 418
Propriedade: Event[].ScreenY 419
Propriedade: Event[].Speed 419
Propriedade: Event[].StopAnimation 420
Propriedade: Event[].Transparency 420
Propriedade: Event[].X 421
Propriedade: Event[].Y 422
Objeto: f 422
Propriedade: f.Count 423
Método: f.ReAlloc() 423
Propriedade: f[] 424

Objeto: Faceset 425
Objeto: Faceset.AuroraSheet 426
Propriedade: Faceset.Filename 426
Propriedade: Faceset.Flipped 427
Propriedade: Faceset.Frame 428
Propriedade: Faceset.Position 429
Objeto: File 430
Método: File.CloseAll() 430
Propriedade: File.Count 431
Método: File.NextFree() 431
Método: File.ReAlloc() 432
Objeto: File[] 433
Método: File[].Close() 434
Método: File[].GetFilePointer() 435
Método: File[].Length() 436
Método: File[].Open() 437
Método: File[].ReadRawData() 438
Método: File[].SetFilePointer() 439
Método: File[].Truncate() 441
Método: File[].WriteRawData() 441
Objeto: Game 443
Método: Game.Load() 443
Método: Game.Quit() 444
Propriedade: Game.RM2K 445
Propriedade: Game.RM2K3 445
Método: Game.Save() 446
Método: Game.SlotExists() 447
Objeto: Item 448
Propriedade: Item.Count 448
Objeto: Item[] 449
Categoria: Geral 450
Propriedade: Item[].Agility 450
Propriedade: Item[].Attack 451
Propriedade: Item[].ConditionChance 451
Propriedade: Item[].Cost 451
Propriedade: Item[].Defense 452
Propriedade: Item[].Description 452
Método: Item[].HasAttribute() 453
Método: Item[].HasCondition() 453
Propriedade: Item[].Mind 454
Propriedade: Item[].Name 454
Propriedade: Item[].NumberOfUses 455
Propriedade: Item[].Type 455
Método: Item[].UsableByActor() 456
Objeto: Item[].Material 456
Propriedade: Item[].Material.AgilityDiff 457
Propriedade: Item[].Material.AttackDiff 457
Propriedade: Item[].Material.DefenseDiff 458
Propriedade: Item[].Material.MaxHpDiff 458
Propriedade: Item[].Material.MaxMpDiff 458
Propriedade: Item[].Material.MindDiff 459
Objeto: Item[].Medicine 459
Propriedade: Item[].Medicine.EffectiveOnlyOnDeathHeroes 460
Propriedade: Item[].Medicine.PercentRecoveryHp 460
Propriedade: Item[].Medicine.PercentRecoveryMp 460
Propriedade: Item[].Medicine.PointRecoveryHp 461

Propriedade: Item[].Medicine.PointRecoveryMp 461
Propriedade: Item[].Medicine.Scope 462
Propriedade: Item[].Medicine.UseOnFieldOnly 462
Objeto: Item[].Protection 462
Propriedade: Item[].Protection.AvoidPhysicalAttack 463
Propriedade: Item[].Protection.HalfMp 463
Propriedade: Item[].Protection.PreventCriticalHit 464
Propriedade: Item[].Protection.PreventTerrainDamage 464
Objeto: Item[].Scroll 464
Propriedade: Item[].Scroll.ShowUseMessage 465
Propriedade: Item[].Scroll.Skill 465
Objeto: Item[].Switch 466
Propriedade: Item[].Switch.AvailableOnBattle 466
Propriedade: Item[].Switch.AvailableOnField 466
Propriedade: Item[].Switch.Switch 467
Objeto: Item[].Weapon 467
Propriedade: Item[].Weapon.AttackAllEnemies 468
Propriedade: Item[].Weapon.AttackTwice 468
Propriedade: Item[].Weapon.BattleAnimation 469
Propriedade: Item[].Weapon.CriticalHitChance 469
Propriedade: Item[].Weapon.HitChance 469
Propriedade: Item[].Weapon.IgnoreMonsterEvasion 470
Propriedade: Item[].Weapon.MpCost 470
Propriedade: Item[].Weapon.PreemptiveAttack 470
Propriedade: Item[].Weapon.TwoHanded 471
Objeto: Keyboard 471
Método: Keyboard.GetKey() 472
Método: Keyboard.GetKeyState() 473
Método: Keyboard.GetKeyText() 474
Método: Keyboard.SetKeyState() 475
Objeto: Logic 476
Categoria: Operações lógicas 476
Categoria: Operações básicas 476
Método: Logic.And() 477
Método: Logic.Not() 478
Método: Logic.Or() 479
Categoria: Operações compostas 481
Método: Logic.And3() 481
Método: Logic.Imp() 483
Método: Logic.Inh() 484
Método: Logic.Nand() 485
Método: Logic.Nand3() 486
Método: Logic.Nor() 487
Método: Logic.Nor3() 488
Método: Logic.Or3() 490
Método: Logic.Xnor() 491
Método: Logic.Xnor3() 492
Método: Logic.Xor() 494
Método: Logic.Xor3() 495
Categoria: Comparações 496
Método: Logic.Above() 497
Método: Logic.AboveEqual() 498
Método: Logic.Below() 499
Método: Logic.BelowEqual() 500
Método: Logic.Equal() 501
Método: Logic.If() 502

Método: Logic.Unequal() 503
Objeto: Map 505
Categoria: Chipset do mapa 505
Propriedade: Map.Chipset 505
Objeto: Map.AuroraSheet 506
Categoria: Informações gerais do mapa 506
Propriedade: Map.Height 507
Propriedade: Map.HeroX 507
Propriedade: Map.HeroY 508
Propriedade: Map.Id 508
Propriedade: Map.NumberOfSaves 508
Propriedade: Map.ScreenX 509
Propriedade: Map.ScreenY 509
Propriedade: Map.ScrollType 510
Propriedade: Map.Width 510
Categoria: Conteúdo do mapa 511
Propriedade: Map.EventCount 511
Propriedade: Map.Lower[] 511
Propriedade: Map.Upper[] 512
Objeto: MapEvent 513
Objeto: Math 514
Categoria: Constantes 514
Propriedade: Math.E 515
Propriedade: Math.Pi 515
Categoria: Funções de potência 516
Método: Math.Exp() 516
Método: Math.Lb() 517
Método: Math.Lg() 518
Método: Math.Ln() 519
Método: Math.Log() 519
Método: Math.Power() 521
Método: Math.Sqrt() 522
Categoria: Funções de arredondamento 523
Método: Math.Decimal() 523
Método: Math.Integer() 524
Método: Math.Round() 525
Método: Math.RoundDown() 526
Método: Math.RoundUp() 527
Categoria: Outras funções 529
Método: Math.Abs() 529
Método: Math.Cmp() 530
Categoria: Funções de ângulo 531
Categoria: Funções hiperbólicas inversas 531
Método: Math.ACosH() 531
Método: Math.ACotH() 532
Método: Math.ACscH() 533
Método: Math.ASecH() 534
Método: Math.ASinH() 535
Método: Math.ATanH() 535
Categoria: Funções hiperbólicas 536
Método: Math.CosH() 537
Método: Math.CotH() 537
Método: Math.CscH() 538
Método: Math.SecH() 539
Método: Math.SinH() 540
Método: Math.TanH() 541

Categoria: Funções trigonométricas inversas 542
Método: Math.ACos() 542
Método: Math.ACot() 544
Método: Math.ACsc() 545
Método: Math.ASec() 547
Método: Math.ASin() 548
Método: Math.ATan() 550
Categoria: Funções trigonométricas 551
Método: Math.Cos() 552
Método: Math.Cot() 554
Método: Math.Csc() 555
Método: Math.Sec() 557
Método: Math.Sin() 558
Método: Math.Tan() 560
Categoria: Números aleatórios 561
Método: Math.Rand() 562
Propriedade: Math.RandSeed 563
Objeto: Mouse 563
Propriedade: Mouse.X 564
Propriedade: Mouse.Y 564
Objeto: Panorama 565
Objeto: Panorama.AuroraSheet 565
Propriedade: Panorama.Height 566
Método: Panorama.LoadFromFile() 567
Propriedade: Panorama.Name 568
Propriedade: Panorama.PanHorizontal 568
Propriedade: Panorama.PanVertical 569
Propriedade: Panorama.ScrollPosHorizontal 570
Propriedade: Panorama.ScrollPosVertical 570
Propriedade: Panorama.ScrollSpeedHorizontal 571
Propriedade: Panorama.ScrollSpeedVertical 572
Propriedade: Panorama.Used 573
Propriedade: Panorama.Width 573
Objeto: Picture 574
Propriedade: Picture.Count 574
Objeto: Picture[] 575
Propriedade: Picture[].Action 577
Propriedade: Picture[].ActionStrength 578
Propriedade: Picture[].ActionValue 578
Objeto: Picture[].AuroraSheet 579
Método: Picture[].AuroraSheet.Clone() 580
Método: Picture[].AuroraSheet.CopyPalette() 581
Método: Picture[].AuroraSheet.Create() 582
Método: Picture[].AuroraSheet.FlushPalette() 583
Método: Picture[].AuroraSheet.Free() 584
Propriedade: Picture[].AuroraSheet.Handle 585
Propriedade: Picture[].AuroraSheet.Height 585
Método: Picture[].AuroraSheet.IsEmpty() 586
Método: Picture[].AuroraSheet.LoadFromFile() 586
Objeto: Picture[].AuroraSheet.Palette 588
Propriedade: Picture[].AuroraSheet.Palette.Count 588
Propriedade: Picture[].AuroraSheet.Palette[] 589
Propriedade: Picture[].AuroraSheet.Pixel[] 590
Método: Picture[].AuroraSheet.SaveAs() 591
Propriedade: Picture[].AuroraSheet.UseMaskColor 593
Propriedade: Picture[].AuroraSheet.Width 594

Categoria: Funções de desenho 594
Método: Picture[].AuroraSheet.BltRect() 595
Método: Picture[].AuroraSheet.CopyRect() 597
Método: Picture[].AuroraSheet.DrawLine() 599
Método: Picture[].AuroraSheet.FillRect() 601
Método: Picture[].BltRect() 603
Propriedade: Picture[].Blue 605
Propriedade: Picture[].Chroma 605
Método: Picture[].CopyRect() 606
Método: Picture[].DrawLine() 608
Propriedade: Picture[].Filename 608
Método: Picture[].FillRect() 608
Método: Picture[].FlushPalette() 609
Propriedade: Picture[].Green 609
Propriedade: Picture[].Height 610
Método: Picture[].LoadFromFile() 610
Propriedade: Picture[].Magnification 611
Propriedade: Picture[].MapMove 612
Propriedade: Picture[].MapX 613
Propriedade: Picture[].MapY 613
Propriedade: Picture[].MoveX 614
Propriedade: Picture[].MoveY 615
Propriedade: Picture[].Palette[] 615
Propriedade: Picture[].Pixel[] 616
Propriedade: Picture[].Red 616
Método: Picture[].SaveAs() 617
Propriedade: Picture[].Transparency 617
Propriedade: Picture[].UseMaskColor 618
Propriedade: Picture[].Width 618
Propriedade: Picture[].X 619
Propriedade: Picture[].Y 620
Objeto: s 621
Propriedade: s.Count 621
Propriedade: s[] 622
Objeto: Screen 623
Propriedade: Screen.Blue 623
Propriedade: Screen.Chroma 624
Propriedade: Screen.Green 625
Propriedade: Screen.HighColor 625
Propriedade: Screen.Red 626
Propriedade: Screen.TrueColor 627
Objeto: Server 628
Método: Server.Accept() 628
Método: Server.Close() 630
Método: Server.Listen() 630
Propriedade: Server.State 632
Propriedade: Server.Type 633
Objeto: Skill 634
Propriedade: Skill.Count 634
Objeto: Skill[] 634
Categoria: Geral 635
Propriedade: Skill[].Description 636
Propriedade: Skill[].FailureMessage 636
Propriedade: Skill[].MpCost 636
Propriedade: Skill[].Name 637
Propriedade: Skill[].Sound 637

Propriedade: Skill[].Type 637
Propriedade: Skill[].UseMessage 638
Propriedade: Skill[].UseMessage2 638
Objeto: Skill[].Normal 639
Propriedade: Skill[].Normal.Absorption 640
Propriedade: Skill[].Normal.Agility 640
Propriedade: Skill[].Normal.Attack 641
Propriedade: Skill[].Normal.BaseEffect 641
Propriedade: Skill[].Normal.BaseSuccessRate 641
Propriedade: Skill[].Normal.BattleAnimation 642
Propriedade: Skill[].Normal.ChangeAttributeEffect 642
Propriedade: Skill[].Normal.Defense 642
Método: Skill[].Normal.HasAttribute() 643
Método: Skill[].Normal.HasCondition() 643
Propriedade: Skill[].Normal.HitChance 644
Propriedade: Skill[].Normal.HP 644
Propriedade: Skill[].Normal.IgnoreDefense 645
Propriedade: Skill[].Normal.Mind 645
Propriedade: Skill[].Normal.MindChance 646
Propriedade: Skill[].Normal.MP 646
Propriedade: Skill[].Normal.Scope 646
Propriedade: Skill[].Normal.Variance 647
Objeto: Skill[].Switch 648
Propriedade: Skill[].Switch.AvailableOnBattle 648
Propriedade: Skill[].Switch.AvailableOnField 648
Propriedade: Skill[].Switch.Switch 649
Objeto: String 649
Método: String.Chr() 650
Método: String.Compare() 651
Método: String.CompareLex() 652
Método: String.Fill() 654
Método: String.Format() 655
Método: String.Length() 656
Método: String.LTrim() 657
Método: String.MonthName() 658
Método: String.Ord() 660
Método: String.Pos() 661
Método: String.Replace() 662
Método: String.Reverse() 663
Método: String.RTrim() 664
Método: String.SubStr() 665
Método: String.ToLower() 666
Método: String.ToUpper() 667
Método: String.Trim() 668
Método: String.WeekdayName() 669
Objeto: Time 671
Propriedade: Time.Day 672
Propriedade: Time.Hour 672
Propriedade: Time.Millisecond 673
Propriedade: Time.Minute 673
Propriedade: Time.Month 674
Propriedade: Time.Second 675
Propriedade: Time.Tick 675
Propriedade: Time.Weekday 676
Propriedade: Time.Year 677
Objeto: v 677

Propriedade: v.Count 678
Propriedade: v[] 679
**Lista de grupos de constantes 680**
Grupo de constantes: (Des)ativação 683
Grupo de constantes: Página exibida no EnterHeroName 683
Grupo de constantes: Exibição dos gráficos do sistema 684
Grupo de constantes: Tipos de atributo 684
Grupo de constantes: Modos de chamada de CommonEvents 685
Grupo de constantes: Peças de equipamento 685
Grupo de constantes: Comandos de movimento 686
Grupo de constantes: Transições de imagem 687
Grupo de constantes: Direções de olhar dos eventos 688
Grupo de constantes: Valores booleanos 688
Grupo de constantes: Modos de arquivo 689
Grupo de constantes: Posições em arquivo 689
Grupo de constantes: Atributos do sistema de arquivos 690
Grupo de constantes: Tipos de dados 690
Grupo de constantes: Erros do Destiny 691
Grupo de constantes: Níveis de efeito 693
Grupo de constantes: Valores de evento 694
Grupo de constantes: Posições do face 694
Grupo de constantes: Frames de eventos 695
Grupo de constantes: Grupo inteiro em comandos de status 695
Grupo de constantes: Valores de herói 696
Grupo de constantes: Tipos de música de fundo 697
Grupo de constantes: Tipos de item 697
Grupo de constantes: Usos de item 698
Grupo de constantes: Valores de item 698
Grupo de constantes: Áreas de efeito de item 699
Grupo de constantes: Tipos de condição 699
Grupo de constantes: Restrições de condição 700
Grupo de constantes: Meses 700
Grupo de constantes: Formatos de mensagem 701
Grupo de constantes: Posições de mensagem 701
Grupo de constantes: Ações de imagem (Picture) 702
Grupo de constantes: Fontes 702
Grupo de constantes: Tipos de scroll 703
Grupo de constantes: Socket: seleção do servidor 703
Grupo de constantes: Socket: flags 704
Grupo de constantes: Status do socket 704
Grupo de constantes: Tipos de socket 705
Grupo de constantes: Caracteres especiais 705
Grupo de constantes: Outros valores 706
Grupo de constantes: Tipos de som 706
Grupo de constantes: Eventos especiais 707
Grupo de constantes: Idiomas 707
Grupo de constantes: Valores de status 708
Grupo de constantes: Operações com switches 708
Grupo de constantes: Eventos de tecla 709
Grupo de constantes: Áreas de efeito de técnicas 709
Grupo de constantes: Tipos de técnica 710
Grupo de constantes: Cores de texto 710
Grupo de constantes: Operações com variáveis 711
Grupo de constantes: Variantes de exibição da tela 712
Grupo de constantes: Variantes de limpeza da tela 713
Grupo de constantes: Códigos virtuais de tecla 714

Grupo de constantes: Tipos de valor 716
Grupo de constantes: Tipos de clima 717
Grupo de constantes: Intensidades de clima 717
Grupo de constantes: Formato de ângulo 718
Grupo de constantes: Dias da semana 718
**Lista de erros 720**
Erro desconhecido 722
Erro 1: Sintaxe 722
Erro 2: Nenhum valor informado 722
Erro 3: Identificador desconhecido 723
Erro 4: Erro de conversão 723
Erro 5: Escrita em valor somente leitura 724
Erro 6: Limites do array excedidos 724
Erro 7: Valor fora da faixa permitida 725
Erro 8: Erro no gerenciamento de memória 725
Erro 9: Valor inválido 725
Erro 10: Operação binária com valor de ponto flutuante 726
Erro 11: Operação matemática com Switch 726
Erro 12: Operação matemática com String 727
Erro 13: Erro de ponto flutuante 727
Erro 14: Número decimal longo demais 728
Erro 15: Divisão por zero 728
Erro 16: Formato de string inválido 729
Erro 17: Posição de texto inválida 729
Erro 18: Imagem não inicializada 729
Erro 19: Pixel fora da área da imagem 730
Erro 20: Imagem de origem e destino iguais com sobreposição 730
Erro 21: Entrada fora da paleta de 256 cores 731
Erro 22: Falha na inicialização do socket 731
Erro 23: Nenhum socket livre disponível 732
Erro 24: Não foi possível criar o socket 732
Erro 25: Socket não fechado 733
Erro 26: Socket não aberto 733
Erro 27: Falha na conexão 734
Erro 28: Operação inválida para este tipo de socket 734
Erro 29: Erro em operação de socket 735
Erro 30: Out-of-Band 735
Erro 31: String longa demais 735
Erro 32: O handle de arquivo não está livre 736
Erro 33: Não foi possível resolver o caminho 736
Erro 34: Sem permissão de acesso 737
Erro 35: Não foi possível abrir o arquivo 737
Erro 36: Arquivo não aberto 737
Erro 37: Acesso a arquivo impossível 738
Erro 38: Não foi possível criar a pasta 738
Erro 39: Não foi possível excluir a pasta 738
Erro 40: Não foi possível renomear o arquivo/pasta 739
Erro 41: Não foi possível copiar o arquivo 739
Erro 42: Não foi possível excluir o arquivo 739
Erro 43: Não foi possível ler os atributos 740
Erro 44: Não foi possível gravar os atributos 740
Erro 45: Busca já aberta 740
Erro 46: Não foi possível iniciar a busca 741
Erro 47: Busca não iniciada 741
Erro 48: Função do Destiny bloqueada 741
Erro 49: Execução de arquivo impossível 742

Erro 50: Parâmetros insuficientes 742
Erro 51: Não comparável 742
Erro 52: Sinal (prefixo) inválido 743
Erro 53: O valor não pode ser gravado aqui 743
Erro 54: EndIf ausente 744
Erro 55: Loop ausente 744
Erro 56: Condição ausente 745
Erro 57: Next ausente 745
Erro 58: Case ausente 745
Erro 59: EndSwitch ausente 746
Erro 60: Break não é permitido aqui 746
Erro 61: Continue não é permitido aqui 747
Erro 62: Handle inválido 747
Erro 63: Operador esperado 747
Erro 64: Acesso ao slot de memória impossível 748
Erro 65: Exceção ocorrida 748
Erro 66: Nível de operador de laço inválido 749
Erro 67: Continue não se refere a um laço 749

**Diagnóstico 750**

**MessageLink 751**

**DestinyDebugger 754**

Exibição de erros 754
A janela do DestinyDebugger 755
Saída de depuração 760

**Apêndice 762**

**Apêndice 763**

Alterações da linguagem de script 763
Protocolo utilizado nos sockets do Destiny 767
Comparação com outros patches 770
Cálculo de chips inferiores e superiores (Lower/Upper) 773

**Palavras finais 775**

**Informações legais (Impressum) 778**

PARTE 1

# Primeiros passos

Introdução
Guia rápido

# Introdução

## O que é o DestinyPatch?

O DestinyPatch é uma extensão para o RPG Maker da empresa Enterbrain, criada por Bananen-Joe. Ela **não** é uma extensão oficial da Enterbrain.

O DestinyPatch se divide em duas partes:

1. **DestinyPatcher** — o programa com o qual o `RPG_RT` (ou seja, o EXE do jogo feito no RPG Maker) pode ser editado.
2. **DestinyScript** — a linguagem de script do DestinyPatch, fornecida pela `Destiny.dll`.

## O que o DestinyPatch pode fazer?

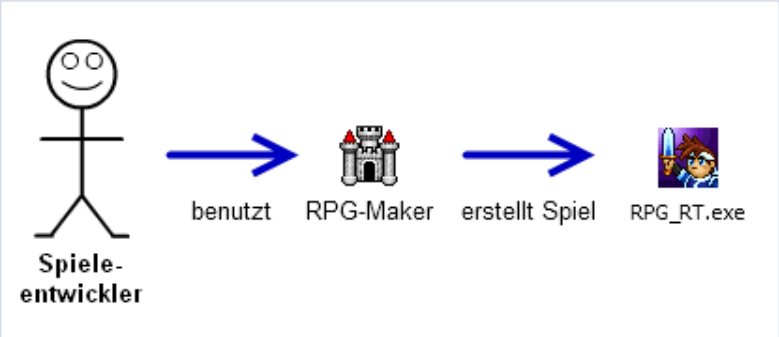
Por um lado, o DestinyPatch permite remodelar o `RPG_RT`. Assim é possível alterar, por exemplo, a configuração das teclas, o modo de janela padrão, a quantidade de imagens (pictures) e muito mais. Para isso usa-se o **DestinyPatcher**.

Por outro lado, o DestinyPatch dota o `RPG_RT` de uma linguagem de script que oferece recursos como cálculo com ponto flutuante, edição de imagens, comunicação pela internet, manipulação de arquivos, chamada de funções do RPG Maker e muito mais. Para isso usa-se a linguagem **DestinyScript** (contida na `Destiny.dll`).

## Como o DestinyPatch funciona?

Como dito, o DestinyPatch é composto por duas partes: o DestinyPatcher e a `Destiny.dll`. O DestinyPatcher é a parte usada pelo desenvolvedor do jogo para modificar o `RPG_RT` de modo que ele carregue a `Destiny.dll`. Depois disso, o `RPG_RT` usa a `Destiny.dll` para interpretar a linguagem de script.

Graficamente, o desenvolvimento de um jogo **sem** o DestinyPatch seria mais ou menos assim:




*Desenvolvimento sem o DestinyPatch*

O desenvolvedor usa o RPG Maker para criar um jogo, que depois pode ser jogado por meio do `RPG_RT`.

**Com** o DestinyPatch, o desenvolvimento fica assim:

*Desenvolvimento com o DestinyPatch*

O desenvolvedor usa o RPG Maker para criar um jogo, que depois pode ser jogado por meio do `RPG_RT` (como de costume). Além disso, porém, ele usa o DestinyPatcher para alterar o `RPG_RT` conforme as próprias necessidades. Uma dessas alterações faz com que o `RPG_RT` passe a usar a `Destiny.dll` para interpretar a linguagem de script.

## Avisos importantes sobre o DestinyPatch

- O DestinyPatch é uma extensão para o RPG Maker da Enterbrain criada por Bananen-Joe e **não** é uma extensão oficial da Enterbrain.
- O autor desenvolveu o patch com o melhor de seu conhecimento e boa-fé, mas não é possível excluir a existência de erros. Por isso, ele se isenta de quaisquer danos que possam surgir direta ou indiretamente. O autor, contudo, agradece qualquer dica útil para a correção de erros.
- O DestinyPatch inteiro é um software **gratuito**. A menos que o autor o tenha permitido expressamente, qualquer distribuição paga é ilegal.
- Este arquivo de ajuda contém, em alguns pontos, links para sites externos (como a Wikipédia). Como a internet é um meio em constante mudança (qualquer pessoa pode, por exemplo, editar artigos da Wikipédia), o autor se isenta de todo o conteúdo referenciado externamente. No momento em que este arquivo de ajuda foi criado, todos os links externos estavam corretos e apoiavam o conteúdo com explicações etc. Como o arquivo de ajuda é estático, mudanças posteriores nesses conteúdos externos não podem ser acompanhadas.

# Guia rápido

## Incluindo a linguagem de script

1. **Iniciar o DestinyPatcher** — Primeiro, abra o `DestinyPatcher` (por exemplo, pelo menu Iniciar do Windows).
2. **Abrir o RPG_RT com o DestinyPatcher** — Depois de abrir o DestinyPatcher, escolha no menu **Arquivo → Abrir** (*Datei → Öffnen*) e selecione o `RPG_RT.exe` do projeto de RPG Maker que será editado.
3. **Escolher a categoria "DestinyDLL"** — Na lista à esquerda do DestinyPatcher, selecione a entrada `DestinyDLL`.
4. **Marcar "Verwendet Destiny.dll?" (Usa a Destiny.dll?)** — Marque essa caixa. O número de versão deve ser o mais recente. A caixa "MessageLink aktiviert?" (MessageLink ativado?) também deveria, de preferência, estar marcada.
5. **Salvar o RPG_RT** — O `RPG_RT.exe` precisa ser salvo para que as alterações tenham efeito. Para isso, escolha no menu **Arquivo → Salvar** (*Datei → Speichern*).
6. **Copiar a Destiny.dll para a pasta do jogo** — Para poder usar a linguagem de script, a `Destiny.dll` correspondente precisa ser copiada para a pasta do jogo. Escolha no menu **Arquivo → Copiar Destiny.dll** (*Datei → Destiny.dll kopieren*).
7. **Fechar o DestinyPatcher** — Se você não precisar mais do DestinyPatcher, pode fechá-lo agora.
8. **Testar a linguagem de script** — No RPG Maker, a linguagem de script pode ser testada no projeto inserindo em um evento um **comentário que comece com** `$`.

Os próximos passos sobre a linguagem de script estão em Estrutura da linguagem de script.

> **Observação:** os textos da interface do DestinyPatcher aparecem no idioma configurado nas opções (por padrão, alemão). Por isso, sempre que uma opção é citada neste manual, o nome original alemão vem entre parênteses.

PARTE 2

# DestinyPatcher

DestinyPatcher

# DestinyPatcher

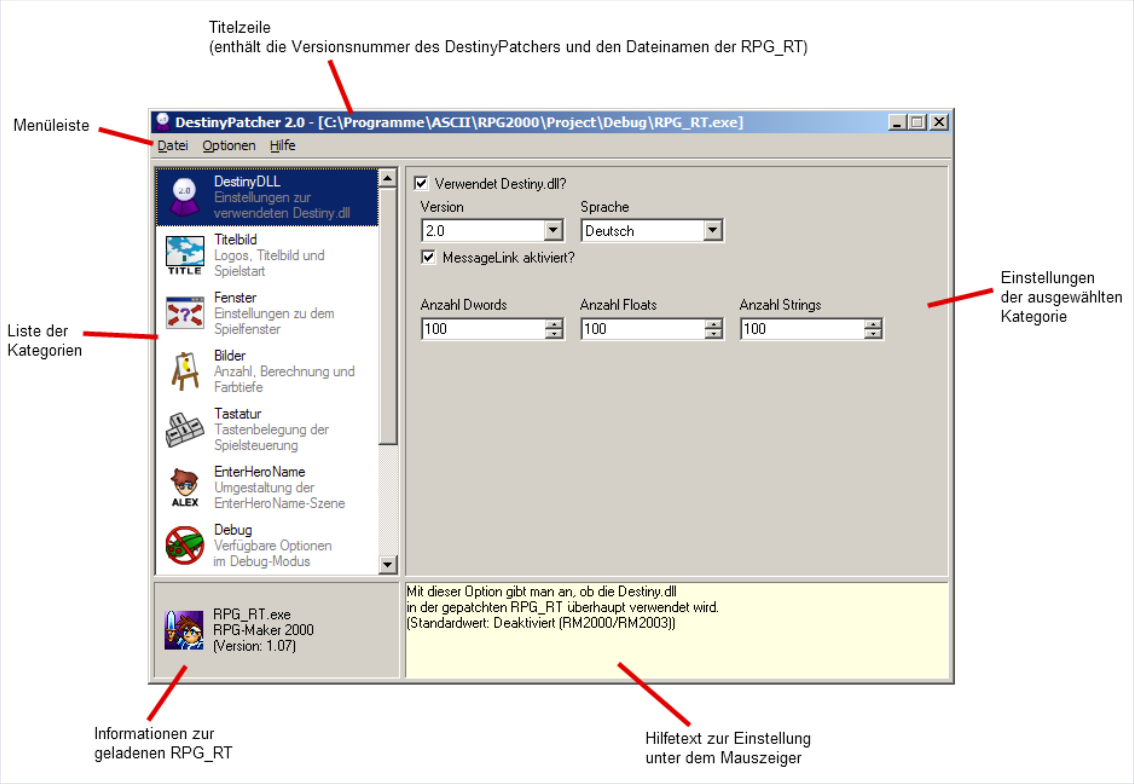
## O que é o DestinyPatcher?

O DestinyPatcher é o programa com o qual o `RPG_RT` (isto é, o EXE do jogo feito no RPG Maker) pode ser editado.

Para incluir a linguagem de script do DestinyPatch (o "DestinyScript") **uma única vez** no jogo, é preciso usar o DestinyPatcher.

## Estrutura do DestinyPatcher

Depois que um `RPG_RT` é carregado, a janela do DestinyPatcher tem aproximadamente este aspecto:



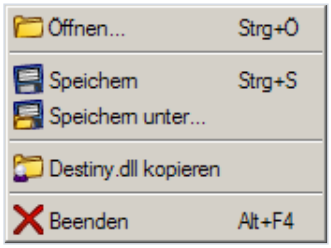

*Janela do DestinyPatcher*

À esquerda fica a lista de categorias; ao selecionar uma, suas opções aparecem à direita. As categorias estão descritas em Categorias do DestinyPatcher, e os menus em Menu do DestinyPatcher.

GUIA

# Menu do DestinyPatcher

## Menu Arquivo (*Datei*)




*Menu Arquivo*

### Abrir... (*Öffnen…*)

Seleciona um `RPG_RT` para abrir.

### Salvar (*Speichern*)

Salva o `RPG_RT` atualmente aberto, **sem** pedir confirmação.

### Salvar como (*Speichern unter*)

Salva o `RPG_RT` atualmente aberto com outro nome de arquivo. Se um arquivo existente for sobrescrito, o programa pergunta antes se isso é desejado.

### Copiar Destiny.dll (*Destiny.dll kopieren*)

Copia a `Destiny.dll` utilizada para a pasta do `RPG_RT` atualmente carregado.

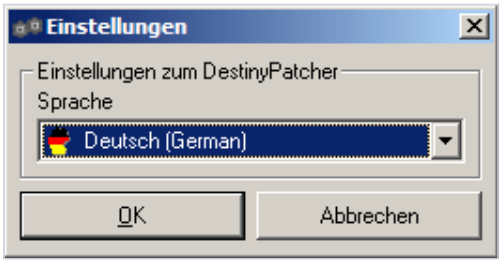
### Sair (*Beenden*)

Fecha o DestinyPatcher sem salvar. Se houver alterações, elas são descartadas **sem** pedido de confirmação.

## Menu Opções (*Optionen*)

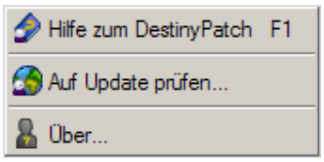
### Configurações... (*Einstellungen…*)

Abre uma janela para alterar as configurações do DestinyPatcher. Por enquanto, essa janela permite alterar apenas o idioma.




*Janela de configurações*

## Menu Ajuda (*Hilfe*)




*Menu Ajuda*

### Ajuda sobre o DestinyPatch (*Hilfe zum DestinyPatch*)

Abre esta ajuda.

### Verificar atualizações (*Auf Update prüfen*)

Abre no navegador padrão o site da `Destiny.dll` com as informações de versão mais recentes. Se você estiver usando uma versão desatualizada do DestinyPatcher ou da `Destiny.dll`, as atualizações serão oferecidas ali.

### Sobre... (*Über...*)

Abre uma janela de informações sobre o criador do DestinyPatch.

GUIA

# Categorias do DestinyPatcher

Como se vê no DestinyPatcher, as diversas opções de configuração estão organizadas em categorias:

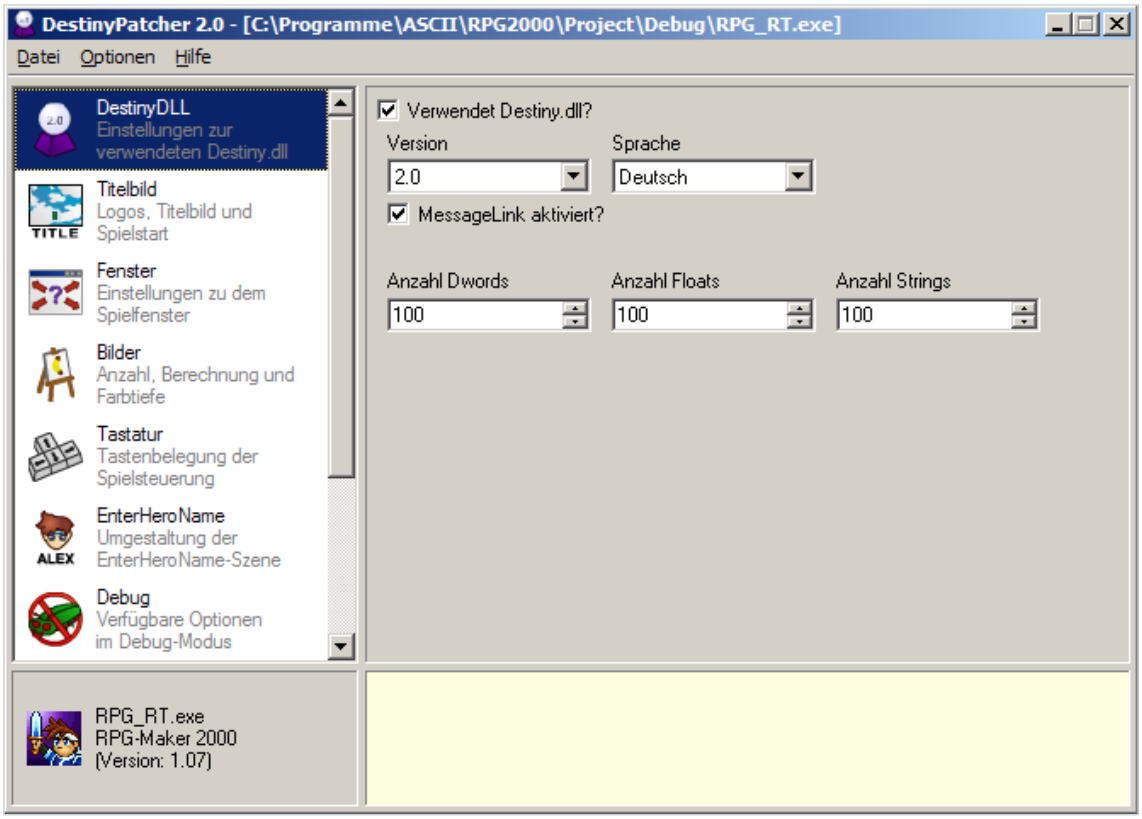
| Categoria | O que permite configurar |
|---|---|
| `DestinyDLL` | Incluir a `Destiny.dll` (e, com ela, a linguagem de script) no projeto. |
| `Titelbild` (Tela de título) | Ajustes da tela de título (por exemplo, quantidade de logos, pular a tela de título, tecla da tela de título...). |
| `Fenster` (Janela) | Ajustes da janela do jogo (por exemplo, se o jogo inicia em tela cheia). |
| `Bilder` (Imagens) | Ajustes das imagens ("Pictures"), por exemplo quantas imagens estão disponíveis no jogo. |
| `Tastatur` (Teclado) | Alterar a configuração das teclas do jogo (somente teclado). |
| `EnterHeroName` | Configurar a entrada de nomes de heróis ("EnterHeroName") e consertá-la em computadores não japoneses. |
| `Debug` | Alterar as configurações do modo de depuração do RPG Maker (por exemplo, disponibilidade e teclas). |
| `Blitting` | Alterar a ordem de desenho das diversas camadas do RPG Maker (por exemplo, BattleAnimation sobre as Pictures). |
| `Strings` | Alterar diversas strings (por exemplo, nomes de arquivos e fontes). |
| `Musik` (Música) | Ajustes da reprodução de música (por exemplo, se a `Harmony.dll` é usada). |

DESTINYPATCHER

# Categoria do Patcher: DestinyDLL

## Descrição

Esta categoria permite fazer os ajustes necessários para a linguagem de script. O ajuste fundamental é a caixa de seleção que indica se a `Destiny.dll` deve ser usada. Se ela não estiver marcada, a `Destiny.dll` não é utilizada.




*Categoria DestinyDLL*

## Estrutura

### Usa a Destiny.dll? (*Verwendet Destiny.dll?*)

Indica se o `RPG_RT` modificado usa a `Destiny.dll` (e, portanto, a linguagem de script). Se a caixa não estiver marcada, os demais ajustes desta categoria ficam indisponíveis.

> **Informação:** o DestinyPatcher também pode ser usado apenas para editar `RPG_RT`s **sem** adicionar a linguagem de script. Portanto, incluí-la não é obrigatório.

> **Informação:** depois de adicionar a linguagem de script, a `Destiny.dll` ainda precisa ser copiada para a pasta do jogo. Isso pode ser feito pelo item de menu correspondente.

### Versão (*Version*)

Indica a versão necessária da `Destiny.dll`. Essa versão precisa estar presente na pasta do jogo para que ele possa iniciar.

> **Atenção:** alguns ajustes das outras categorias não são compatíveis com a versão 1.0. Além disso, a versão 1.0 só está disponível para os `RPG_RT`s 1.0.5 e 1.0.7 do RPG Maker 2000.

### Idioma (*Sprache*)

Indica o idioma usado, ao iniciar o jogo, nas mensagens de erro e no separador decimal. Esse valor pode ser alterado durante a execução por meio de `Destiny.Language`. Somente alemão e inglês são suportados.

### MessageLink ativado? (*MessageLink aktiviert?*)

Indica se marcadores (placeholders) adicionais para caixas de mensagem ("Show Message") são liberados. Mais detalhes no capítulo sobre o MessageLink.

### Quantidade de Dwords (*Anzahl Dwords*)

Indica a quantidade de Dwords (`d[]`) disponíveis ao iniciar o jogo. Esse valor pode ser alterado durante a execução com `d.ReAlloc()`.

### Quantidade de Floats (*Anzahl Floats*)

Indica a quantidade de Floats (`f[]`) disponíveis ao iniciar o jogo. Esse valor pode ser alterado durante a execução com `f.ReAlloc()`.

### Quantidade de Strings (*Anzahl Strings*)

Indica a quantidade de Strings (`a[]`) disponíveis ao iniciar o jogo. Esse valor pode ser alterado durante a execução com `a.ReAlloc()`.

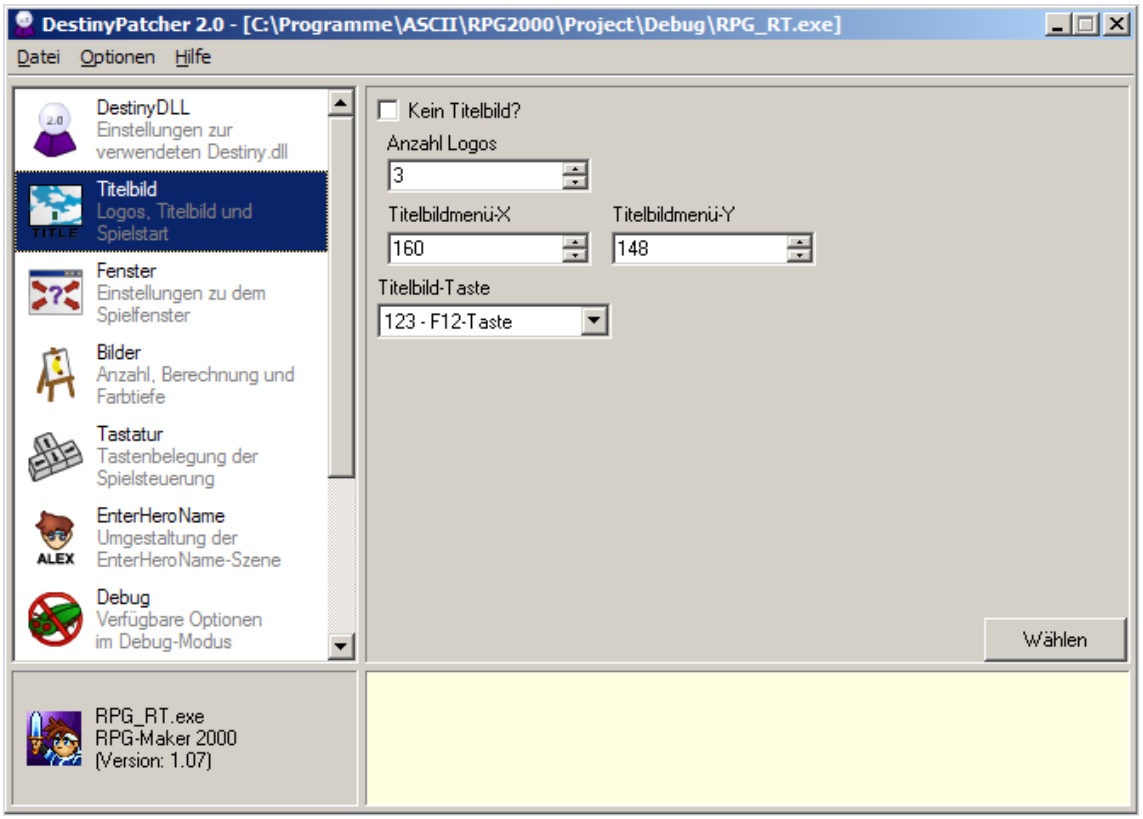
---

DESTINYPATCHER

# Categoria do Patcher: Tela de título

## Descrição

Esta categoria permite fazer os ajustes da tela de título. É possível, por exemplo, suprimir totalmente a tela de título e assim criar uma tela de título própria (por exemplo, com animação).




*Categoria Tela de título*

## Estrutura

### Sem tela de título? (*Kein Titelbild?*)

Indica se a tela de título (e, com ela, os logos) é ignorada. Se for ignorada, o jogo começa diretamente na posição inicial. As funções da tela de título podem ser reproduzidas com DestinyScript:

- para **carregar** o jogo, use `Game.Load()`;
- para **encerrar** o jogo, use `Game.Quit()`.

> **Informação:** se a tela de título for ignorada, pressionar a tecla da tela de título simplesmente **reinicia** o jogo. Se isso não for desejado, a tecla da tela de título deve ser definida como `0 - Keine` (nenhuma).

### Quantidade de logos (*Anzahl Logos*)

Indica quantos logos (= anúncios exibidos antes da tela de título) aparecem antes dela. Se o valor for 0, nenhum logo é exibido.

### Menu do título – X (*Titelbildmenü-X*)

Indica a coordenada X do menu da tela de título. Essa coordenada se refere ao **centro** do menu.

### Menu do título – Y (*Titelbildmenü-Y*)

Indica a coordenada Y do menu da tela de título. Essa coordenada se refere à **borda superior** do menu.

### Tecla da tela de título (*Titelbildtaste*)

Indica a tecla que, quando pressionada, interrompe o jogo e volta à tela de título. Se a tela de título for ignorada, pressionar essa tecla reinicia o jogo. Se isso não for desejado, o valor da tecla pode ser definido como `0 - Keine`, o que desliga por completo a função da tecla da tela de título.

### Escolher (*Wählen*)

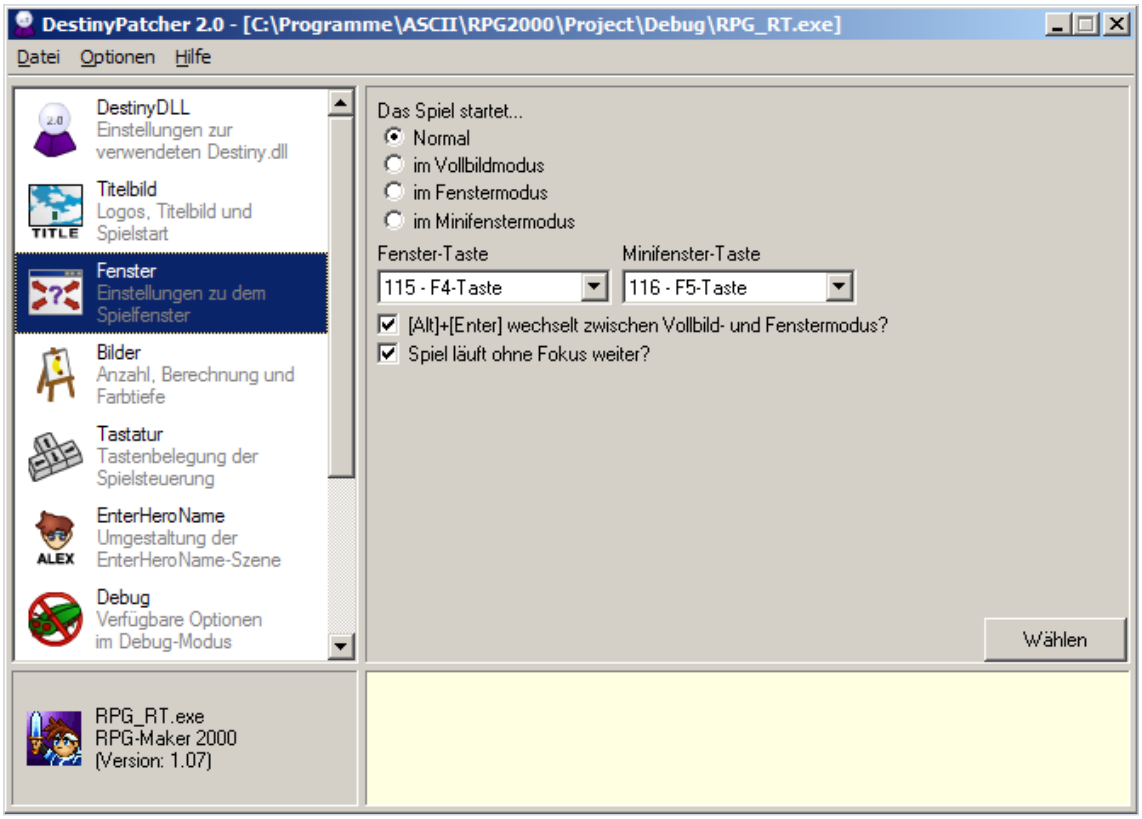
Ao clicar nesse botão e depois em uma lista de seleção de teclas (aqui, apenas a tecla da tela de título), o número da tecla pressionada é atribuído ao campo simplesmente pressionando-se a tecla no teclado.

DESTINYPATCHER

# Categoria do Patcher: Janela

## Descrição

Esta categoria permite fazer os ajustes da janela. É possível, por exemplo, definir em que modo de janela o jogo inicia e quais teclas alternam o modo de janela.




*Categoria Janela*

## Estrutura

### O jogo inicia... (*Das Spiel startet...*)

Com este ajuste é possível escolher em que modo de janela o jogo inicia:

| Opção | Significado |
|---|---|
| **Normal** (*Normal*) | O jogo inicia em tela cheia, a menos que outro modo seja indicado por parâmetro de inicialização (este é o caso normal). |
| **em tela cheia** (*im Vollbildmodus*) | O jogo sempre inicia em tela cheia (independentemente dos parâmetros de inicialização). |
| **em modo janela** (*im Fenstermodus*) | O jogo sempre inicia em modo janela (independentemente dos parâmetros de inicialização). |
| **em modo minijanela** (*im Minifenstermodus*) | O jogo sempre inicia em modo minijanela (independentemente dos parâmetros de inicialização). |

### Tecla de janela (*Fenster-Taste*)

Indica a tecla que alterna entre tela cheia e modo janela.

### Tecla de minijanela (*Minifenster-Taste*)

Indica a tecla que alterna entre modo janela e modo minijanela.

### [Alt]+[Enter] alterna entre tela cheia e modo janela? (*[Alt]+[Enter] wechselt zwischen Vollbild- und Fenstermodus?*)

Indica se a combinação de teclas [Alt]+[Enter] também alterna entre tela cheia e modo janela.

### O jogo continua rodando sem foco? (*Spiel läuft ohne Fokus weiter?*)

Indica se o jogo continua rodando quando, em modo janela, perde o foco (por exemplo, ao clicar em outro programa). Essa opção **deve** ser ativada em jogos que usam a internet/rede. Normalmente o jogo congelaria e, com isso, não conseguiria receber os pacotes de dados que chegam.

### Escolher (*Wählen*)

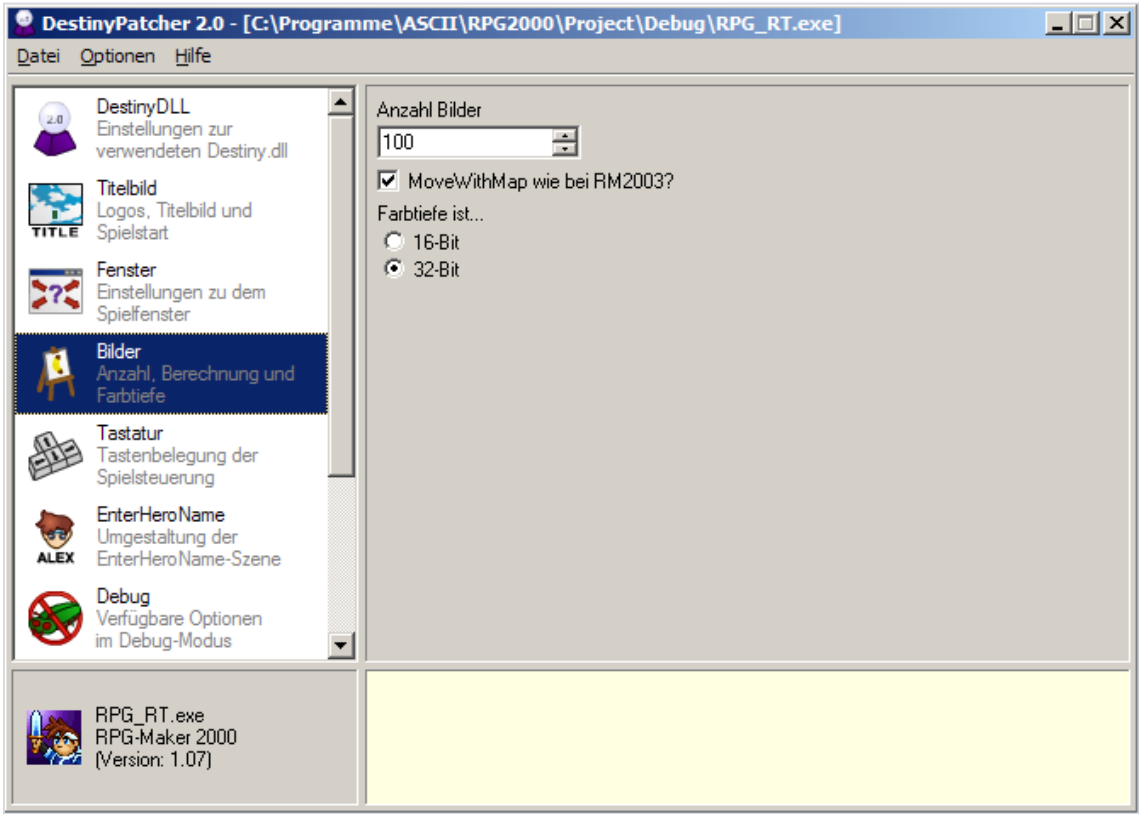
Ao clicar nesse botão e depois em uma lista de seleção de teclas, o número da tecla pressionada é atribuído ao campo simplesmente pressionando-se a tecla no teclado.

DESTINYPATCHER

# Categoria do Patcher: Imagens

## Descrição

Esta categoria permite fazer os ajustes relativos às imagens ("Pictures"). É possível, por exemplo, definir quantas imagens, no máximo, estão disponíveis no jogo.




*Categoria Imagens*

## Estrutura

### Quantidade de imagens (*Anzahl Bilder*)

Indica a quantidade máxima de imagens disponíveis no jogo.

> **Informação:** se a quantidade máxima de imagens for alterada, o próprio RPG Maker também precisa ser ajustado para que mais imagens fiquem disponíveis.

### MoveWithMap como no RM2003? (*MoveWithMap wie bei RM2003?*)

Indica se, para imagens com a propriedade MoveWithMap ativada, as coordenadas do comando MovePicture são calculadas como no RPG Maker 2003. Com essa opção ativada, `MovePicture` (com X = 1 e Y = 0) significa, por exemplo, que a imagem deve ser deslocada 1 pixel para a direita. Com a opção desativada, para obter a mesma ação seria preciso pegar as coordenadas originais do `ShowPicture` e somar 1 ao valor de X.

### A profundidade de cor é... (*Farbtiefe ist…*)

Indica qual profundidade de cor o jogo usa. Isso influencia apenas a qualidade das cores do jogo, e não a quantidade máxima de cores por imagem (256 cores).

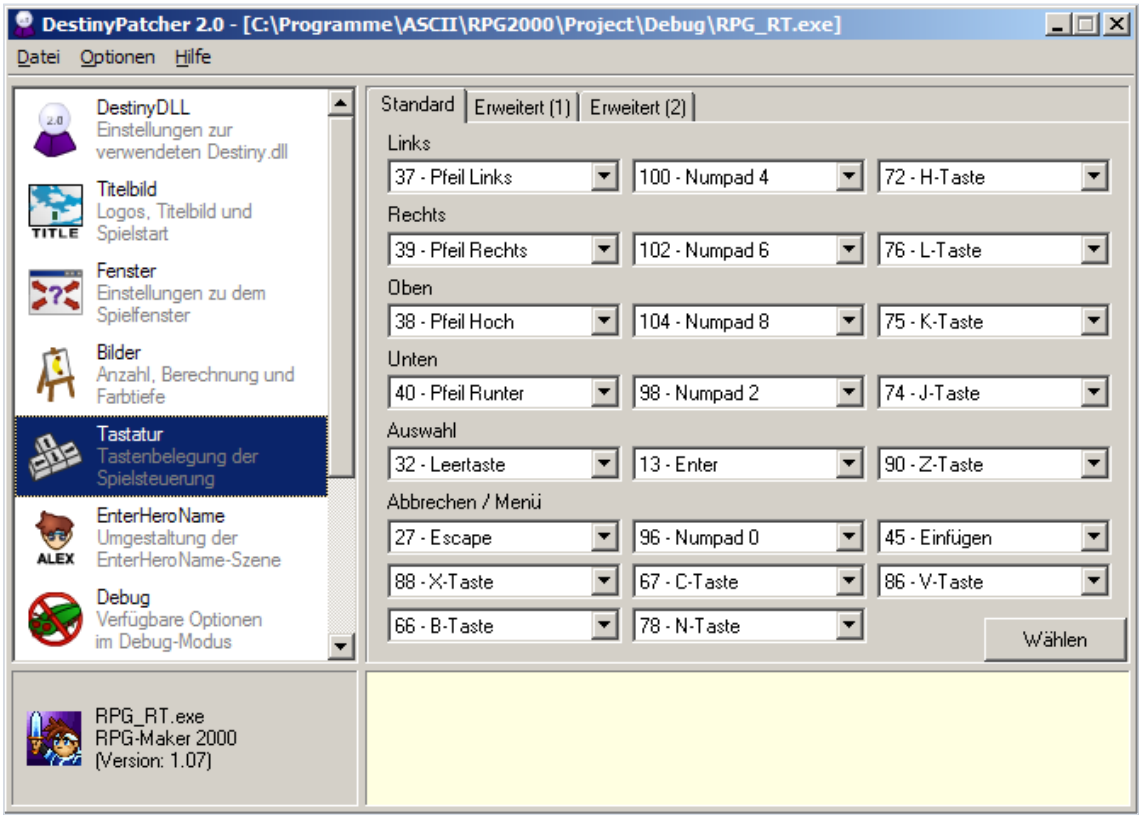
| Opção | Significado |
|---|---|
| **16 bits** | As cores são exibidas no modo 565 (32 níveis de vermelho, 64 de verde, 32 de azul). |
| **32 bits** | As cores são exibidas no modo XRGB (256 níveis de vermelho, 256 de verde, 256 de azul). |

DESTINYPATCHER

## Categoria do Patcher: Teclado

## Descrição

Esta categoria permite alterar a configuração das teclas do controle do jogo (somente teclado).




*Categoria Teclado*

## Estrutura

### Esquerda (*Links*)

Indica até três teclas possíveis para mover o personagem para a esquerda.

### Direita (*Rechts*)

Indica até três teclas possíveis para mover o personagem para a direita.

### Cima (*Oben*)

Indica até três teclas possíveis para mover o personagem para cima.

### Baixo (*Unten*)

Indica até três teclas possíveis para mover o personagem para baixo.

### Confirmar (*Auswahl*)

Indica até três teclas possíveis para confirmar uma seleção.

### Cancelar / Menu (*Abbrechen* / *Menü*)

Indica até oito teclas possíveis para abrir o menu ou cancelar uma seleção.

### Zero (0) a Nove (9) (*Null (0)* ... *Neun (9)*)

No RPG Maker 2003, cada dígito de 0 a 9 tem seu próprio campo, que indica até **duas** teclas possíveis para digitar aquele dígito. São os campos *Null (0)*, *Eins (1)*, *Zwei (2)*, *Drei (3)*, *Vier (4)*, *Fünf (5)*, *Sechs (6)*, *Sieben (7)*, *Acht (8)* e *Neun (9)*.

### Somar (+) (*Addieren (+)*)

No RPG Maker 2003, indica uma tecla que pode ser usada para somar.

### Subtrair (-) (*Subtrahieren (-)*)

No RPG Maker 2003, indica uma tecla que pode ser usada para subtrair.

### Multiplicar (*) (*Multiplizieren (*)*)

No RPG Maker 2003, indica uma tecla que pode ser usada para multiplicar.

### Dividir (/) (*Dividieren (/)*)

No RPG Maker 2003, indica uma tecla que pode ser usada para dividir.

### Separador decimal (,) (*Dezimaltrennzeichen (,)*)

No RPG Maker 2003, indica uma tecla que pode ser usada para separar casas decimais.

### Escolher (*Wählen*)

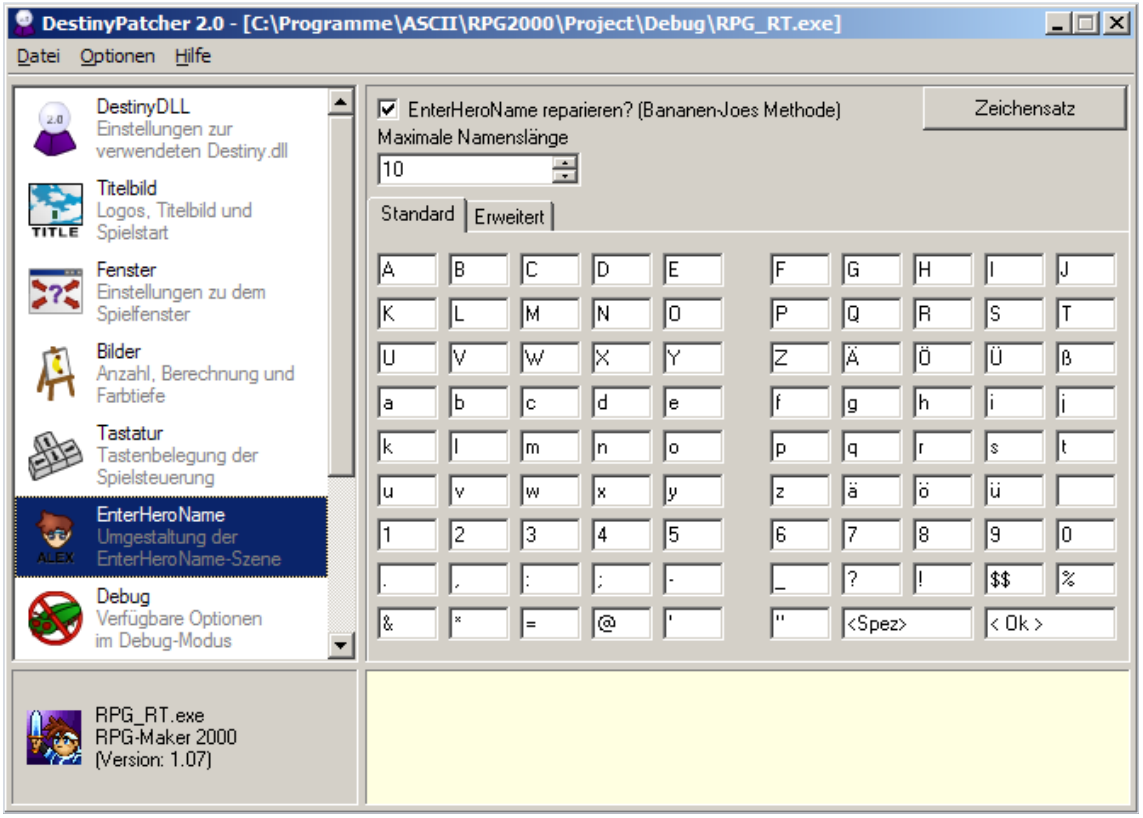
Ao clicar nesse botão e depois em uma lista de seleção de teclas, o número da tecla pressionada é atribuído ao campo simplesmente pressionando-se a tecla no teclado.

DESTINYPATCHER

# Categoria do Patcher: EnterHeroName

## Descrição

Esta categoria permite consertar a entrada de nomes de heróis em computadores não japoneses e ajustar os valores dos campos.




*Categoria EnterHeroName*

## Estrutura

### Consertar o EnterHeroName? (método de Bananen-Joe) (*EnterHeroName reparieren?*)

Com essa caixa marcada, o `EnterHeroName` volta a funcionar em computadores não japoneses. Além disso, pode-se usar um comprimento de caractere variável (1 ou 2 caracteres), o que permite, por exemplo, o uso de caracteres especiais (como `$A` para a espada).

### Conjunto de caracteres (*Zeichensatz*)

Permite escolher um conjunto de caracteres padrão:

| Opção | Significado |
|---|---|
| **RM2000 - [Don Miguel]** | Usa o conjunto de caracteres que "Don Miguel" usou na época na tradução do RPG Maker 2000. |
| **RM2000 - [ASCII]** | Usa o conjunto de caracteres que a fabricante do RPG Maker 2000 ("ASCII") usava originalmente. |
| **Conjunto alemão** (*Deutscher Zeichensatz*)* | Usa um conjunto com os caracteres mais comuns do idioma alemão. |

* Esta entrada de menu depende do idioma configurado no DestinyPatcher.

### Comprimento máximo do nome (*Maximale Namenslänge*)

Permite definir o comprimento máximo do nome de um herói na entrada de nome. Esse ajuste só está disponível se o `EnterHeroName` tiver sido consertado pelo método de Bananen-Joe (caixa acima).

### Padrão (*Standard*)

Permite definir os caracteres oferecidos na entrada de nome. "Padrão" corresponde à página **Hiragana** do RPG Maker (= primeira página).

### Estendido (*Erweitert*)

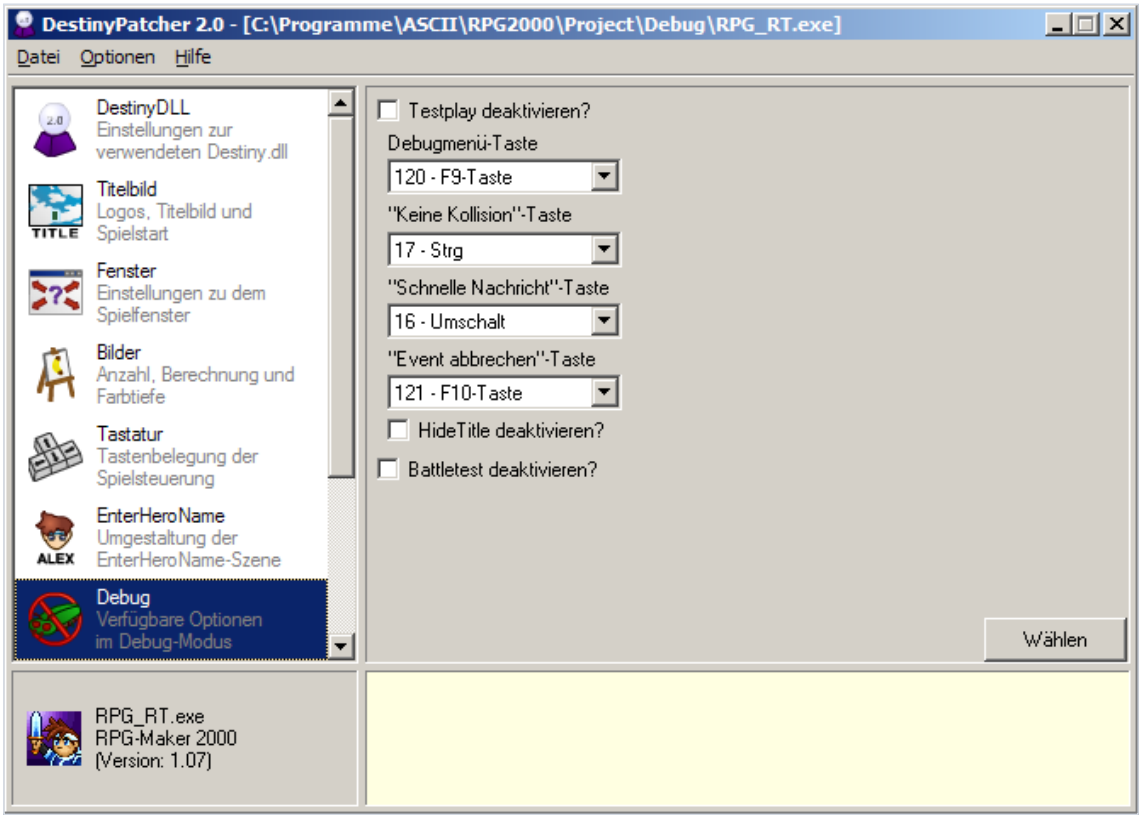
Permite definir os caracteres oferecidos na entrada de nome. "Estendido" corresponde à página **Katakana** do RPG Maker (= segunda página).

DESTINYPATCHER

## Categoria do Patcher: Debug

## Descrição

Esta categoria permite (des)ativar as diversas opções do modo de depuração do `RPG_RT`.




*Categoria Debug*

## Estrutura

### Desativar o Testplay? (*Testplay deaktivieren?*)

Com a caixa marcada, o modo de depuração padrão do `RPG_RT` não pode mais ser usado. Independentemente dos parâmetros do jogo, ele passa a iniciar sempre no modo normal.

### Tecla do menu de depuração (*Debugmenü-Taste*)

Indica a tecla que abre o menu de depuração (exibição de variáveis e switches).

### Tecla "Sem colisão" (*"Keine Kollision"-Taste*)

Indica a tecla que permite ao jogador atravessar paredes e andar sobre eventos (sem acioná-los).

### Tecla "Mensagem rápida" (*"Schnelle Nachricht"-Taste*)

Indica a tecla que faz o conteúdo de uma caixa de mensagem aparecer imediatamente (sem o atraso da formação do texto).

### Tecla "Cancelar evento" (*"Event abbrechen"-Taste*)

Indica a tecla que permite cancelar o evento atual.

### Desativar o HideTitle? (*HideTitle deaktivieren?*)

Indica se a opção "HideTitle" deve ser desativada. Com a caixa marcada, a tela de título não pode mais ser ocultada por um parâmetro de inicialização.

### Desativar o BattleTest? (*BattleTest deaktivieren?*)

Indica se o BattleTest (isto é, o combate de teste contra monstros no sistema de batalha padrão) deve ser desativado.

### Escolher (*Wählen*)

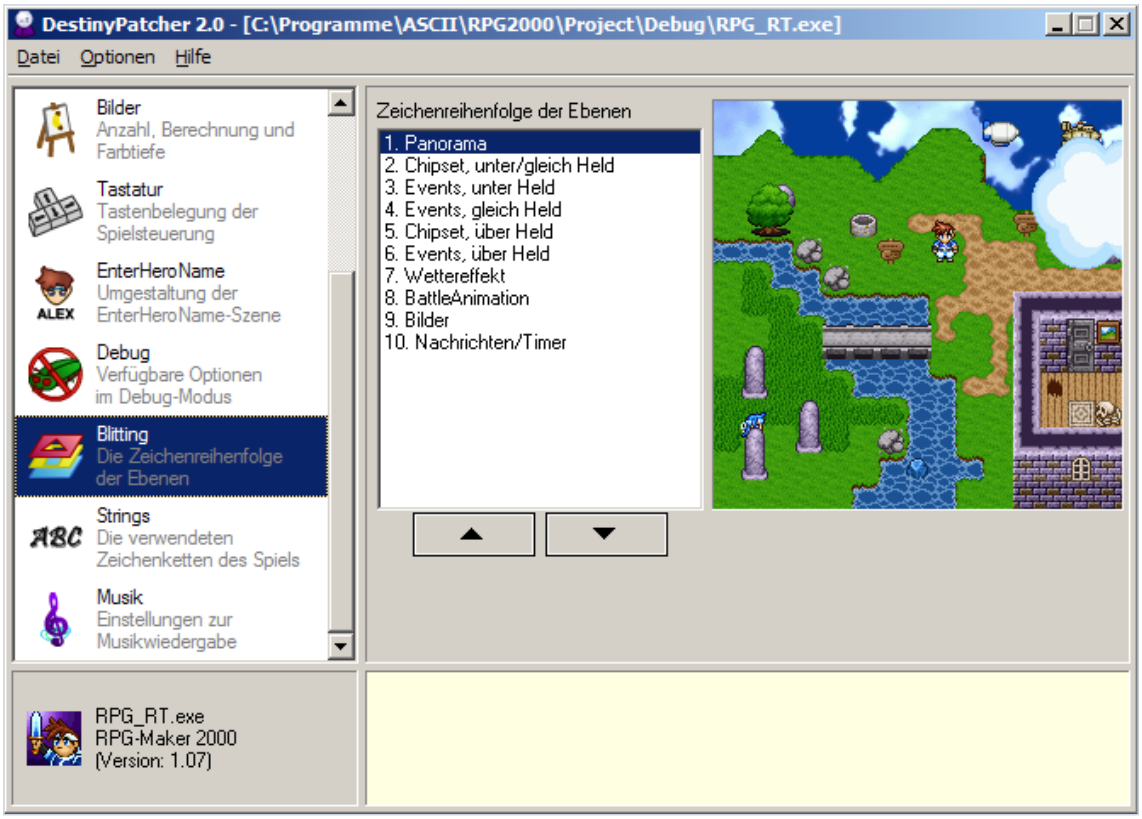
Ao clicar nesse botão e depois em uma lista de seleção de teclas, o número da tecla pressionada é atribuído ao campo simplesmente pressionando-se a tecla no teclado.

DESTINYPATCHER

# Categoria do Patcher: Blitting

## Descrição

Esta categoria permite alterar a ordem de desenho das diversas camadas do RPG Maker.




*Categoria Blitting*

## Estrutura

### Lista (*Liste*)

Esta lista mostra as diversas camadas de desenho (com a ordem atual). As entradas são, da mais ao fundo à mais à frente:

1. Panorama
2. Chipset, abaixo/no mesmo nível do herói
3. Eventos, abaixo do herói
4. Eventos, no mesmo nível do herói
5. Chipset, acima do herói
6. Eventos, acima do herói
7. Efeito de clima
8. BattleAnimation
9. Imagens (Pictures)
10. Mensagens/Timer

A ordem das camadas de desenho pode ser alterada com as setas abaixo da lista.

> **Atenção:** uma alteração na ordem de desenho vale para o jogo inteiro.

### Gráfico de exemplo (*Beispielgrafik*)

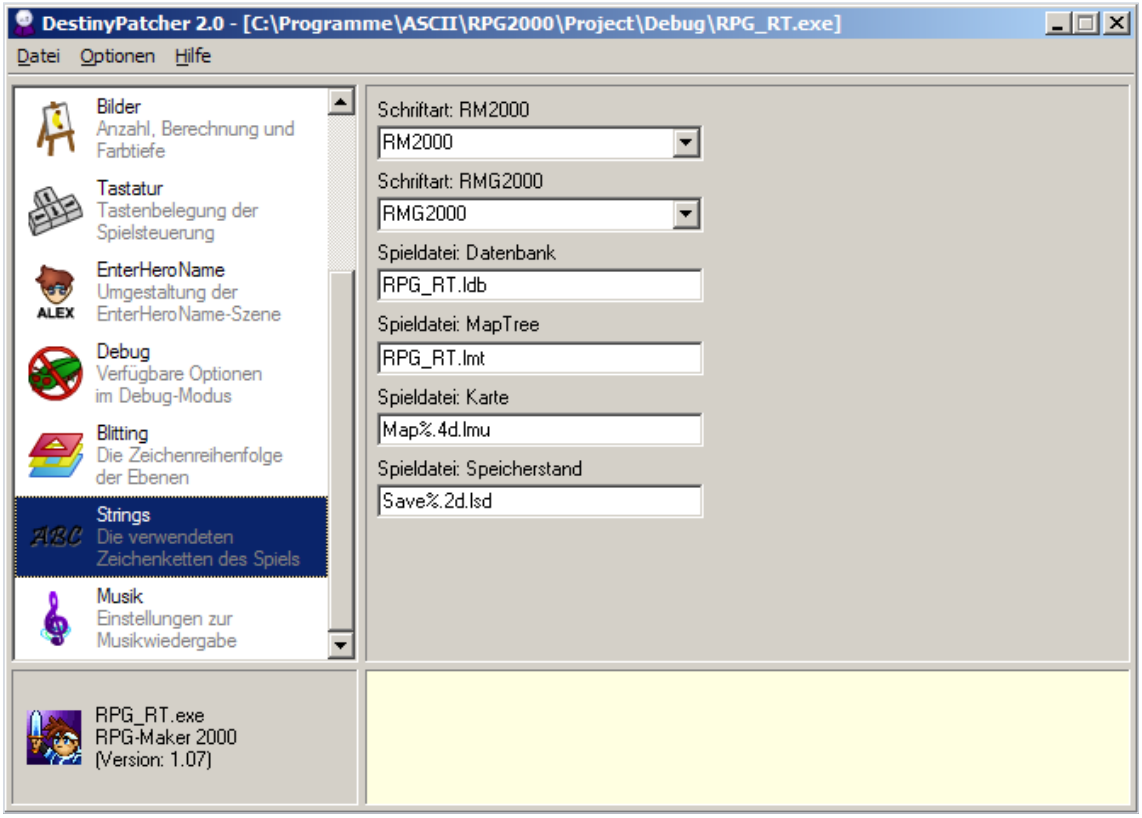
O gráfico de exemplo mostra uma prévia da ordem de desenho configurada. Por falta de espaço, porém, nem todas as camadas de desenho aparecem nesse gráfico.

DESTINYPATCHER

# Categoria do Patcher: Strings

## Descrição

Esta categoria permite alterar algumas das strings do `RPG_RT`. É possível, por exemplo, escolher fontes completamente diferentes.




*Categoria Strings*

## Estrutura

### Fonte: RM2000 (*Schriftart: RM2000*)

Nesta lista é possível escolher a fonte da opção de fonte "RM2000". A fonte escolhida precisa estar instalada no sistema de destino.

### Fonte: RMG2000 (*Schriftart: RMG2000*)

Nesta lista é possível escolher a fonte da opção de fonte "RMG2000". A fonte escolhida precisa estar instalada no sistema de destino.

### Arquivo do jogo: Banco de dados (*Spieldatei: Datenbank*)

Aqui é possível indicar o nome do arquivo do banco de dados do jogo.

> **Atenção:** se o nome do arquivo do banco de dados for alterado, o arquivo correspondente também precisa ser renomeado; caso contrário, o jogo não consegue mais iniciar.

> **Atenção:** se o nome do arquivo do banco de dados for alterado, o RPG Maker não consegue mais abrir o jogo. Para isso, o RPG Maker teria de ser ajustado manualmente também, para que o jogo possa voltar a ser editado.

### Arquivo do jogo: MapTree (*Spieldatei: MapTree*)

Aqui é possível indicar o nome do arquivo da árvore de mapas.

> **Atenção:** se o nome do arquivo da árvore de mapas for alterado, o arquivo correspondente também precisa ser renomeado; caso contrário, o jogo não consegue mais iniciar.

> **Atenção:** se o nome do arquivo da árvore de mapas for alterado, o RPG Maker não consegue mais abrir o jogo. Para isso, o RPG Maker teria de ser ajustado manualmente também.

### Arquivo do jogo: Mapa (*Spieldatei: Karte*)

Aqui é possível indicar o nome dos arquivos dos mapas do jogo. É usado um marcador para um número inteiro: `Map%4d.lmu`, por exemplo, significa que o arquivo do primeiro mapa se chamará `Map0001.lmu`. Se você digitar `Map%6d.lmu`, o arquivo do primeiro mapa passará a se chamar `Map000001.lmu`.

> **Atenção:** se o nome dos arquivos dos mapas for alterado, os arquivos correspondentes também precisam ser renomeados; caso contrário, o jogo trava ao entrar em um desses mapas.

> **Atenção:** se o nome dos arquivos dos mapas for alterado, o RPG Maker não consegue mais abrir o jogo corretamente. Para isso, o RPG Maker teria de ser ajustado manualmente também.

### Arquivo do jogo: Save (*Spieldatei: Speicherstand*)

Aqui é possível indicar o nome dos arquivos de save do jogo. Também é usado um marcador para um número inteiro: `Save%2d.lsd`, por exemplo, significa que o arquivo do primeiro save se chamará `Save01.lsd`. Se você digitar `Save%3d.lsd`, o arquivo do primeiro save passará a se chamar `Save001.lsd`.

> **Atenção:** se o nome dos arquivos de save for alterado, os arquivos correspondentes também precisam ser renomeados, caso os saves existentes devam continuar sendo usados.

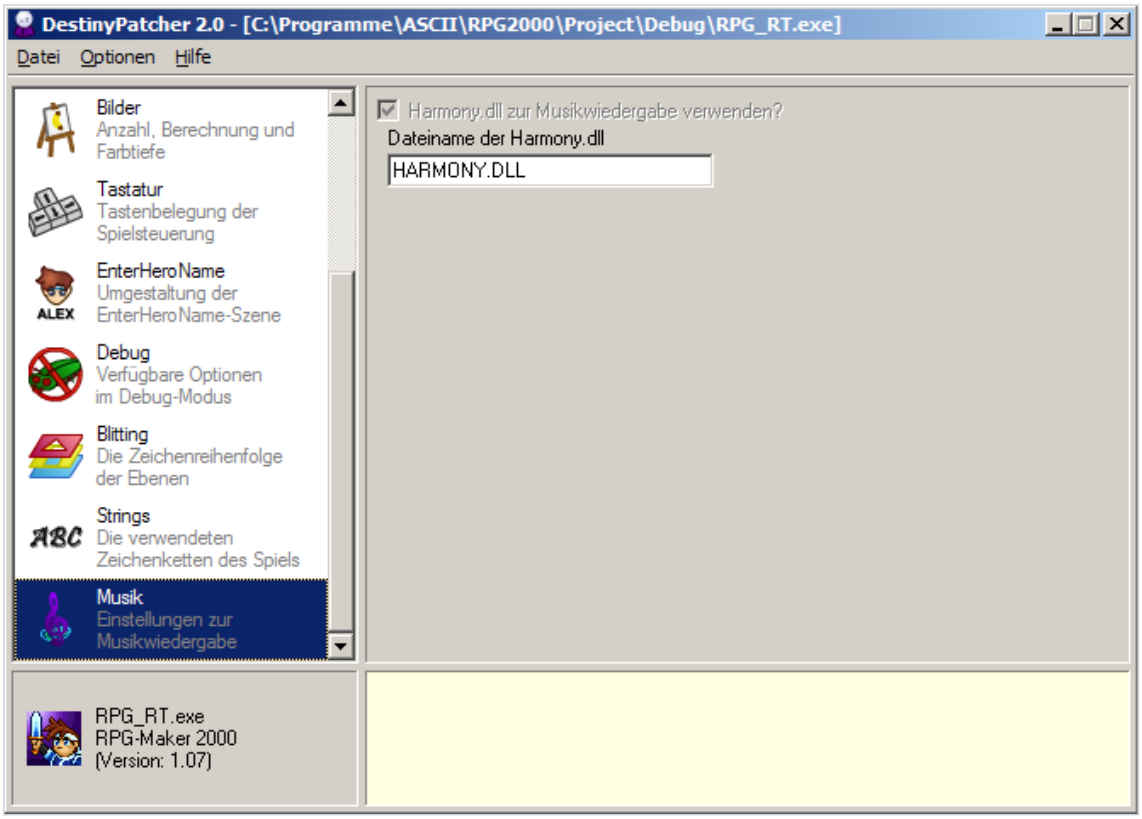
---

DESTINYPATCHER

# Categoria do Patcher: Música

## Descrição

Esta categoria permite alterar os ajustes da reprodução de música com a `Harmony.dll`.




*Categoria Música*

## Estrutura

### Usar a Harmony.dll para reproduzir música? (*Harmony.dll zur Musikwiedergabe verwenden?*)

Com a caixa marcada, a `Harmony.dll` é usada para reproduzir música.

> **Informação:** essa caixa só pode ser alterada se a versão original do `RPG_RT` **não** usar a `Harmony.dll` para reproduzir música. Isso vale apenas para poucas versões do `RPG_RT`.

### Nome do arquivo da Harmony.dll (*Dateiname der Harmony.dll*)

Aqui é possível definir outro nome de arquivo para a `Harmony.dll`. Esse ajuste só está disponível quando a `Harmony.dll` é usada para reproduzir música.

PARTE 3

# Linguagem DestinyScript

Estrutura da linguagem de script

# Estrutura da linguagem de script

## Início de um DestinyScript

O núcleo do DestinyPatch é a extensão do `RPG_RT` com uma linguagem de script chamada "DestinyScript". Para iniciar um DestinyScript, usam-se os comandos de **comentário** de um evento. Como eles servem tanto para DestinyScripts quanto para comentários "normais", todo DestinyScript começa com um **cifrão** (`$`; [Shift] + [4] → `$`). Esse símbolo foi escolhido porque lembra um S, e a palavra *Script* começa com S.

O importante é que o **primeiro caractere** do comentário do evento seja o `$`. Portanto, não pode haver nenhum espaço antes dele; caso contrário, o comentário do evento não será interpretado como DestinyScript.

> **Informação:** se um comentário de evento do RPG Maker deve ser um comentário "normal", mas começa com `$`, ele será sempre interpretado como DestinyScript (o que pode causar mensagens de erro). Nesse caso, o problema se resolve simplesmente escrevendo um espaço antes do `$`, de modo que o `$` deixe de ser o primeiro caractere do comentário do RPG Maker.

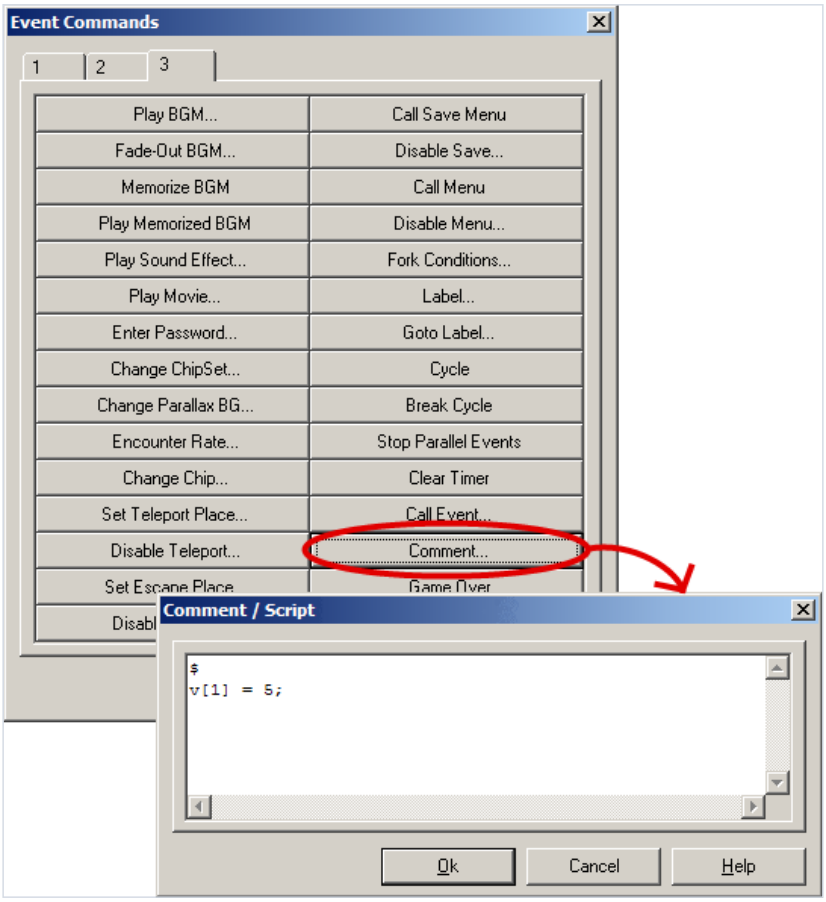
## Exemplo

No primeiro exemplo, um valor será atribuído a uma variável.

Para isso, cria-se um projeto de RPG Maker novo (e **vazio**!) — supõe-se que o banco de dados padrão seja usado. Depois de criado o projeto, o `RPG_RT` precisa ser editado com o DestinyPatcher para usar a `Destiny.dll` (versão: 2.0, idioma: alemão, MessageLink: ativado, quantidade de Dwords: 100, quantidade de Doubles: 100, quantidade de Strings: 100). Além disso, a `Destiny.dll` da versão utilizada precisa ser copiada para a pasta do jogo (item de menu **Arquivo → Copiar Destiny.dll** no menu do DestinyPatcher).

Depois de criado o projeto, adiciona-se um novo evento (ele deve estar visível e poder ser acionado). Nesse evento adiciona-se um comentário, e nele copia-se o seguinte DestinyScript:

```
$
v[1] = 5;
```




*Comentário com DestinyScript em um evento*

A imagem pode diferir conforme a versão do RPG Maker usada (idioma, edição, alterações individuais etc.).

Abaixo desse comentário insere-se agora uma mensagem que exibe o conteúdo da primeira variável (por exemplo, "A variável 1 tem o valor \v[1]").

Ao acionar esse evento no jogo, obtém-se o seguinte resultado:

*Resultado no jogo*

Como se vê, a 1ª variável contém o valor 5. Mas por quê? Todas as variáveis do RPG Maker são inicializadas com 0 ao iniciar o jogo e, como se trata de um projeto novo (e sobretudo vazio), algo diferente deve ter acontecido. A resposta é óbvia: o DestinyScript atribuiu o valor 5 à primeira variável!

## Estrutura de um DestinyScript

Olhando o DestinyScript novamente, o sentido do script vai ficando claro:

```
$
v[1] = 5;
```

Na linha 1 está o `$`, que instrui a `Destiny.dll` a interpretar o comentário do evento como DestinyScript. Na linha 2 está o comando que atribui o valor 5 à primeira variável. Um comando desse tipo sempre tem a mesma estrutura:

- **À esquerda** fica o **destino** da operação. Neste caso, é a primeira variável (daí o `v[1]` — `v` significa *variável* e `[1]` é o número da variável!).

- **No meio** fica o **operador** usado. Aqui, é a **atribuição** (representada por um único sinal de igual; [Shift] + [0] → `=`).
- **À direita** fica o que deve ser escrito no destino (= **origem**). Aqui, o número 5.

### DestinyScripts com vários comandos

Um DestinyScript pode conter vários comandos. Para separá-los usa-se o **ponto e vírgula** ([Shift] + [vírgula] → `;`). Além disso, um comando não precisa ficar em uma linha própria. A rigor, o DestinyScript inteiro poderia estar em uma única linha (o `$` inicial não precisa de linha própria!).

> **Informação:** o último comando de um DestinyScript pode terminar com ponto e vírgula, mas não é obrigatório. Recomenda-se, porém, terminar também o último comando com ponto e vírgula, para que o DestinyScript seja mais fácil de alterar (por exemplo, novo código pode ser inserido logo abaixo) e, não menos importante, para que ele pareça mais bonito (e uniforme).

Um DestinyScript com 2 comandos ficaria assim:

```
$
v[1] = 5;
v[2] = 23;
```

Esse DestinyScript atribuiria o valor 5 à 1ª variável e o valor 23 à 2ª variável.

> **Informação:** dentro de um comentário do RPG Maker pode haver vários comandos. O `$`, porém, só é necessário para **iniciar** o DestinyScript. Portanto, o `$` **não** pode ser escrito antes de cada comando!

### Comportamento em caso de erros

Se ocorrer um erro durante a execução de um DestinyScript, o DestinyScript **inteiro** é interrompido. Isso pode ser vantajoso, mas também desvantajoso. Se, por exemplo, um DestinyScript contém vários comandos e ocorre um erro no primeiro, todos os comandos seguintes não são executados. Para mudar esse comportamento, é possível colocar cada comando em um comentário separado (quando possível) ou alterar o comportamento com o objeto `Errors`. Como esse comportamento só importa quando ocorrem erros, na maioria dos casos ele pode ser ignorado.

## Neste capítulo


- Comentários
- Tipos de dados e áreas de memória
- Operadores
- Expressões de fórmula
- Elementos de estrutura

GUIA

# Comentários

## O que são comentários?

Como o DestinyScript é uma linguagem de script, às vezes é necessário comentar o código para torná-lo compreensível. Para isso, além do código de script (efetivo), é possível incluir no DestinyScript comentários que não fazem parte da execução. Esse conhecimento é necessário para entender os exemplos dos capítulos seguintes, pois neles muitos comentários são usados para descrever o código dos exemplos em DestinyScript.

## Como escrever comentários

Existem dois tipos de comentário:

- comentários de **uma linha**;
- comentários de **várias linhas**.

### Comentários de uma linha

Os comentários de uma linha são iniciados, como em C++, por duas barras seguidas (*slashes*; [Shift] + [7] → `/`). Tudo o que vem depois das duas barras é comentário e não é interpretado como código DestinyScript. O comentário termina no fim da respectiva linha.

```
$
// Esta linha é um comentário
v[1] = 5; // Aqui há um comentário depois de um comando…
```

### Comentários de várias linhas

Os comentários de várias linhas são iniciados, como em C++, por uma barra (`/`) seguida de um asterisco (sinal de multiplicação; [Shift] + [+] → `*`, ou `*` do teclado numérico) e terminam com um asterisco seguido de uma barra. Um comentário assim pode se estender por várias linhas (mas não precisa). Como ele termina com `*/`, essa combinação de caracteres não pode aparecer no texto do comentário. Comentários de várias linhas costumam ser usados para "comentar" (desativar) trechos de código.

```
$
/*
  Este
  é
  um
  comentário
  de várias linhas!
*/

/* Este é um comentário de várias linhas em uma só linha… */
v[1] = /* Comentário no meio do código */ 5;
/*
  Agora vem um código de script comentado:
  v[2] = 23; <-- não é executado, pois está…
*/
```

GUIA

# Tipos de dados e áreas de memória

## O que são tipos de dados?

Quem usa o RPG Maker conhece normalmente 2 tipos de dados:

1. **Variáveis**
2. **Switches** (em português, "interruptores" ou "chaves")

Não é possível comparar diretamente uma variável com um switch, pois eles armazenam tipos de valores completamente diferentes. No RPG Maker 2000, uma variável pode assumir valores numéricos de -999999 a +999999 (com a `Destiny.dll` há uma faixa de valores maior — veja a tabela). Estão disponíveis apenas **números inteiros** (ou seja, números com vírgula, como 1,5, não são possíveis!). Um switch, por sua vez, só pode ter um de dois valores: ligado (1) ou desligado (0).

A `Destiny.dll` conhece outros tipos de dados, que também ficam à disposição do usuário. Os adicionais mais importantes são os **Floats** (números com vírgula) e as **Strings** (textos).

## Quais tipos de dados existem?

No DestinyPatch existem os seguintes tipos de dados:

| Tipo | Tamanho | Faixa de valores | Descrição |
|---|---|---|---|
| `Dword` | 4 bytes | -2147483648 … +2147483647 | O tipo mais usado. Armazena apenas números inteiros. É o tipo usado para as variáveis. |
| `Word` | 2 bytes | -32768 … +32767 | Semelhante ao Dword, mas com faixa de valores menor. Poucos métodos o usam, mas é prático na transferência de dados quando se quer economizar tráfego. Também armazena apenas números inteiros; na prática é raramente usado. |
| `Byte` | 1 byte | 0 … +255 | Semelhante ao Dword, mas com faixa de valores muito menor. Armazena apenas valores bem pequenos e, por isso, gasta pouquíssimos recursos. É o tipo usado para os pixels de uma Picture. Também armazena apenas números inteiros. |
| `Float` | 8 bytes | -1.7E+308 … +1.7E+308; -5.0E-324 … +5.0E-324 | Pode armazenar números muito grandes e muito pequenos, mas ocupa proporcionalmente muito espaço. Do número armazenado guardam-se cerca de 15 dígitos; além do número, guarda-se um expoente (assim é possível armazenar, por exemplo, tanto 1E+100 quanto 1E-100). É o **único** tipo capaz de armazenar números de ponto flutuante. |
| `Bool` | 1 byte | 0 … +1 | Assume apenas valores booleanos (sim ou não). Costumam-se usar as constantes `true` (= 1) e `false` (= 0). É o tipo usado para os switches. |
| `String` | 4 + n bytes | 0 … +2147483647 caracteres (aprox.) | Armazena cadeias de caracteres (por exemplo, nomes, palavras ou frases inteiras). É usado para todos os textos (por exemplo, em `Actor[].Name`, o nome de um herói). |
| `Handle` | — | — | Usado para identificar gráficos. Não pode ser armazenado nem convertido. |

> **Informação:** no DestinyPatch V1.0, o tipo Float ainda se chamava **Double**.

## Áreas de memória

Para poder usar o resultado de um comando, é preciso guardá-lo em algum lugar — "armazená-lo". Para isso existem as chamadas **áreas de memória**. As áreas de memória das variáveis e dos switches já são conhecidas e

são fornecidas por padrão pelo `RPG_RT` (uma área separada para variáveis e outra para switches). A `Destiny.dll` estende o `RPG_RT` com três áreas de memória adicionais:

1. **Dword:** essa área armazena apenas números inteiros.
2. **Float:** essa área armazena números de ponto flutuante.
3. **String:** essa área armazena apenas textos (por exemplo, palavras, nomes, frases...).

Com as áreas de memória adicionais da `Destiny.dll`, é possível armazenar não só mais dados, mas também outros tipos de dados.

Para acessar cada área de memória, o DestinyPatch usa abreviações:

| Abreviação | Tipo | Origem | Descrição |
|---|---|---|---|
| `v[ ]` | Dword | RPG_RT.exe | V de **V**ariável |
| `s[ ]` | Bool | RPG_RT.exe | S de **S**witch |
| `d[ ]` | Dword | Destiny.dll | D de **D**word |
| `f[ ]` | Float | Destiny.dll | F de **F**loat (forma abreviada de *floating point number*, "número de ponto flutuante") |
| `a[ ]` | String | Destiny.dll | A de **A**NSI-String |

### Acesso às áreas de memória

Como acessar a área de memória das variáveis já foi testado no capítulo anterior. Basta escrever o nome abreviado da área de memória e, em seguida, colchetes. Dentro dos colchetes escreve-se o **índice** do elemento que se deseja acessar (por exemplo, `v[1]` para a primeira variável, `v[2]` para a segunda...). O mesmo princípio vale para as outras áreas de memória:

```
$
d[5] = 100;    // Atribui o valor 100 ao 5º Dword
v[3] = d[5];   // Atribui à 3ª variável o valor do 5º Dword
```

Ao executar esse exemplo e exibir depois o conteúdo da terceira variável, vê-se que ela tem o valor 100. No exemplo, primeiro atribui-se o valor 100 ao 5º Dword. Depois atribui-se à 3ª variável o valor do 5º Dword. Dessa forma, a 3ª variável recebeu o valor de um elemento de outra área de memória.

### Endereçamento indireto

Até agora sempre se definiu diretamente qual elemento da área de memória seria acessado (por exemplo, `v[1]` para a primeira variável). Outra possibilidade é **endereçar indiretamente**, informando entre os colchetes, em vez de um número fixo, um elemento de uma área de memória:

```
$
v[4] = 5;       // Atribui o valor 5 à 4ª variável
v[v[4]] = 17;   // Atribui 17 à variável cujo número está na 4ª variável
```

Ao executar esse script e exibir o valor da 5ª variável, o resultado é o número 17. No exemplo, acessou-se a 5ª variável de forma indireta. Para isso, primeiro atribuiu-se à 4ª variável o valor 5 (linha 2), pois depois se pretende acessar o 5º elemento. Em seguida, na linha 3, informou-se entre os colchetes, em vez de um número direto, outra variável. Na execução do DestinyScript, o valor informado entre colchetes é usado como índice. Não importa se esse valor é informado diretamente, vem de uma área de memória ou é calculado.

> **Informação:** todas as áreas de memória podem ser endereçadas indiretamente. As áreas usadas para o endereçamento indireto também podem ser endereçadas indiretamente (por exemplo, `v[v[v[1]]]`). Teoricamente não há limite para a profundidade de endereçamento. Na prática, porém, nunca se precisa de um nível mais profundo que 1.

## Conversão de tipos de dados

Se, dentro de um cálculo, os tipos de dados forem diferentes (por exemplo, porque um Float é multiplicado por um Dword), eles precisam ser ajustados ("convertidos"). Com tipos diferentes, isso normalmente acontece automaticamente (= **implicitamente**). Em alguns casos especiais, porém (por exemplo, quando a conversão é ambígua), há problemas. Assim, em um cálculo misto de Strings e números, ou de switches e números, é exibida uma mensagem de erro. Se ainda assim se quiser misturar esses tipos, é preciso converter os tipos manualmente (= **explicitamente**). Para isso usa-se o objeto `Convert`.

## Neste capítulo


- Números
- Strings


GUIA

# Números

Números são necessários em toda linguagem de programação para calcular coisas (por exemplo, a posição de uma imagem). Como, conforme a finalidade, os números nem sempre são necessários no mesmo formato, em um DestinyScript é possível escrevê-los de várias maneiras. Distinguem-se essencialmente duas coisas:

1. É um número **inteiro** ou um número **com vírgula**?
2. Usa-se o sistema **decimal** (o nosso sistema numérico normal) ou outro (da informática)?

> **Informação:** nesta documentação fala-se, na maioria dos casos, de **números de ponto flutuante** (Gleitkommazahlen). Isso não deve preocupar o usuário comum do patch — ele pode considerar os dois como equivalentes. A diferença essencial é que um número de ponto flutuante tem um expoente separado; assim, o número 1E-100, por exemplo, ocupa exatamente o mesmo espaço que 1E+100. Uma limitação dessa abordagem, porém, é que a quantidade de **dígitos significativos** é restrita (normalmente a 15; também no DestinyPatch). Um número de ponto flutuante com 15 dígitos significativos não consegue, portanto, armazenar 1234567890123456 (16 dígitos diferentes), mas consegue armazenar 123456789012345000000 (este último número é maior, porém menos preciso, pois usa apenas 15 dígitos distintos).

## Números inteiros e números com vírgula

Na maioria das linguagens de programação é comum distinguir entre inteiros e números com vírgula. Isso é necessário porque o computador calcula inteiros de maneira diferente de números com vírgula.

### Como escrever números inteiros

Como escrever um número inteiro já foi testado em exemplos anteriores. Basta escrever o número desejado como de costume:

```
$
v[1] = 12345;
```

Aqui foram usados dois números inteiros: 1 (entre os colchetes) e 12345 (como valor a atribuir). Para um inteiro, portanto, usam-se simplesmente os dígitos habituais, de 0 a 9.

### Como escrever números com vírgula

A escrita de números com vírgula não é bem a habitual. O principal motivo é que, nas linguagens de programação, é comum usar a notação inglesa. Como separador decimal usa-se, portanto, o **ponto**; o separador de milhares é

totalmente dispensado:

```
$
f[1] = 1234.5;
```

Aqui foram usados um inteiro (o 1 entre os colchetes) e um número com vírgula (1234.5 como valor a atribuir).

Outra forma de escrever um número com vírgula é escrever um **inteiro** (sem parte decimal) que seja **grande demais** para ser um inteiro. No capítulo anterior, Tipos de dados, foi descrito que um inteiro (do tipo Dword) só pode estar entre -2147483648 e +2147483647. Se você escrever um número maior (ou menor), ele é interpretado automaticamente como número com vírgula (mesmo sem parte decimal). Se depois se tentar atribuir esse número a um valor de memória do tipo Dword, ocorre um erro. Se, ao contrário, ele for atribuído a um valor de memória do tipo Float, não há erro.

```
$
f[1] = 3000000000;  // Este número é maior que o maior Dword…
```

Como terceira possibilidade, pode-se escrever um número com vírgula em **notação exponencial**. Para isso, escreve-se o número (normalmente com um dígito antes da vírgula e pelo menos um depois), seguido de um `E` e do expoente da base 10:

```
$
f[1] = 3.0E+20;     // Equivale a um 3 seguido de 20 zeros
```

#### *Limitações na escrita de números com vírgula*

Como já descrito acima na caixa de informação, a quantidade de dígitos significativos (isto é, a quantidade de dígitos que restam quando se removem todos os zeros iniciais e finais) é limitada a 15. Essa não é uma limitação específica do DestinyPatch: é o comum na maioria dos programas de computador.

#### *Particularidades dos números com vírgula*

Como internamente o computador não calcula no sistema decimal, mas no binário (zeros e uns), alguns números fáceis de representar para humanos (por exemplo, 0,1) são, para o computador, uma dízima periódica. Assim, numa multiplicação por 0,1 o computador calcula, na verdade, algo como uma multiplicação por 0,0999999999999998. Podem, portanto, ocorrer algumas imprecisões nos últimos dígitos.

### Diferenças no cálculo de inteiros e de números com vírgula

Se em um cálculo aparecem apenas inteiros (tipo Dword), o resultado também será sempre um inteiro. Se aparecem apenas números com vírgula (tipo Float), o resultado também será sempre um número com vírgula (eventualmente sem parte decimal; o tipo, porém, continua Float). Se aparecem números de vários tipos, a conversão é sempre feita para o **maior** tipo. Numa mistura de inteiros e números com vírgula, portanto, todos os inteiros são convertidos em números com vírgula antes do cálculo.

Por causa dessa diferença, alguns cálculos podem dar resultados diferentes:

```
$
f[1] = 3 / 2;       // Resultado é 1   (divisão de inteiros)
f[2] = 3.0 / 2.0;   // Resultado é 1.5 (divisão de números com vírgula)
```

Ao final do cálculo, `f[1]` seria 1 e `f[2]` seria 1,5. Numa divisão de inteiros, portanto, o resto não divisível (que formaria a parte decimal) é descartado. Para mudar esse comportamento, é preciso escrever os números diretamente como números com vírgula (como na linha 3) ou, no caso de variáveis, converter explicitamente o tipo para Float (`Convert.Float()`).

## Sistemas numéricos

Além do conhecido sistema decimal, na informática costumam existir três sistemas numéricos adicionais:

- **Sistema binário** (base 2);
- **Sistema octal** (base 8);
- **Sistema hexadecimal** (base 16).

A única diferença entre todos esses sistemas numéricos é a **base**.

> **Informação:** o conhecido sistema decimal tem base 10 (baseia-se no fato de o ser humano ter 10 dedos). Um número qualquer (por exemplo, 123,45) deve ser interpretado de modo que cada dígito seja multiplicado por uma potência da base com um expoente. O expoente cresce (ou diminui) 1 a cada posição que se afasta da vírgula: o primeiro dígito antes da vírgula tem expoente 0, o segundo 1 e assim por diante; o primeiro dígito depois da vírgula tem expoente -1, o segundo -2 e assim por diante.
>
> Aplicado ao número 123,45:
>
> $123{,}45 = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0 + 4 \cdot 10^{-1} + 5 \cdot 10^{-2}$ $123{,}45 = 1 \cdot 100 + 2 \cdot 10 + 3 \cdot 1 + 4 \cdot 0{,}1 + 5 \cdot 0{,}01$ $123{,}45 = 100 + 20 + 3 + 0{,}4 + 0{,}05$ $123{,}45 = 123{,}45$

**Sistema decimal.** Já foi descrito em detalhes acima. Sua particularidade é que ele pode ser usado para escrever números com vírgula. Isso só é possível no sistema decimal; os outros sistemas numéricos suportam apenas inteiros.

**Sistema binário.** Um número binário começa com `0b` (sem as aspas) e só pode conter os dígitos 0 e 1. Não é possível escrever números com vírgula.

```
$
v[1] = 0b1011011;    // Equivale ao número 91
```

**Sistema octal.** Um número octal começa com `0o` (sem as aspas) e só pode conter os dígitos 0, 1, 2, 3, 4, 5, 6 e 7. Não é possível escrever números com vírgula.

```
$
v[1] = 0o731;        // Equivale ao número 471
```

**Sistema hexadecimal.** Um número hexadecimal começa com `0x` (sem as aspas) e só pode conter os dígitos 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E e F (A = 10, B = 11, C = 12, D = 13, E = 14, F = 15). Tanto faz escrever em maiúsculas ou minúsculas os dígitos que são letras. Não é possível escrever números com vírgula.

```
$
v[1] = 0xF3AB;       // Equivale ao número 62379
```

GUIA

# Strings

Como descrito no capítulo sobre Tipos de dados, as **strings** (em português, "cadeias de caracteres") são basicamente apenas textos. Como, porém, em uma string podem aparecer, em teoria, todos os caracteres possíveis do computador, é preciso distingui-la sintaticamente de outros elementos da linguagem (por exemplo, números, comentários...).

> **Informação:** uma string também pode conter caracteres de comentário. Isso, porém, não é considerado comentário, pois faz parte do conteúdo da string.

Para escrever uma string no DestinyScript há essencialmente duas variantes:

1. **Sintaxe de uma linha**
2. **Sintaxe de várias linhas** — que por sua vez pode ser *simples* ou *complexa* (com marcadores/placeholders).

## Sintaxe de uma linha

É a sintaxe mais usada. O conteúdo da string é colocado entre **aspas duplas**. A string também pode conter aspas duplas, mas elas precisam ser marcadas de modo especial: em vez de uma aspa dupla, escrevem-se **duas seguidas** (as aspas duplas que ficam fora da string, é claro, continuam sendo escritas normalmente).

```
$
a[1] = "Esta é uma string";
a[2] = "String com aspas duplas ("" ) no texto";
```

Na linha 2, ao primeiro valor de memória de string é atribuído o texto `Esta é uma string`. Na linha 3, ao segundo valor de memória de string é atribuído o texto `String com aspas duplas ( " ) no texto`.

## Sintaxe de várias linhas

Para escrever diretamente uma string de várias linhas usa-se a chamada sintaxe **Heredoc** (nome emprestado da linguagem de script PHP). Essa sintaxe permite escrever no DestinyScript, de forma agradável de ler, um texto como string. Como quebra de linha é sempre usado CRLF (como é comum no Windows). O texto também termina sempre com uma quebra de linha desse tipo.

### Sintaxe Heredoc simples

Uma string Heredoc (simples) é iniciada por **três sinais de menor** (`<<<`) seguidos de uma palavra e de uma quebra de linha. A palavra é de livre escolha, mas deve começar com uma letra e só pode conter letras, números e o sublinhado. A string Heredoc termina assim que a palavra escolhida aparece no **início de uma linha**. Depois dessa palavra ainda podem vir outros comandos, que então ficam fora da string (diferentemente do PHP, em que a palavra (exceto um ponto e vírgula) deve ser o único conteúdo da linha).

```
$
a[1] = <<<PalavraQueEncerraAStringHeredoc
Isto aqui faz parte da string
isto também
e isto!
PalavraQueEncerraAStringHeredocEMaisAlgo <- …
 PalavraQueEncerraAStringHeredoc           <- …
PalavraQueEncerraAStringHeredoc;  // <- Fim da string…
```

### Sintaxe Heredoc complexa

Às vezes se quer manter partes da string dinâmicas durante a sua escrita. Para isso existe a sintaxe Heredoc **complexa**, que permite usar **marcadores** (placeholders). Nesses marcadores escreve-se mais código DestinyScript. Esse código é executado (no momento em que a string é interpretada) e o resultado é inserido na string. Para transformar uma sintaxe Heredoc simples em complexa, basta escrever um `$` logo depois da palavra escolhida, que vem após os três sinais de menor. Para encerrar a sintaxe complexa, o `$` **não** é escrito. Um marcador começa com um cifrão seguido de um par de parênteses. Dentro dos parênteses fica o código DestinyScript a ser executado.

```
$
a[1] = <<<HEREDOC_END$
Nome:  $(Actor[1].Name)
Título: $(Actor[1].Degree)
Nível: $(Actor[1].Level)
HEREDOC_END;
```

No final, o primeiro valor de memória de string conteria as informações do primeiro herói. Essas informações **não** se atualizam sozinhas; para isso, o código acima teria de ser executado novamente a cada vez (por exemplo, ao abrir um menu).

## Concatenação de strings

Para concatenar strings (juntá-las) usa-se o **operador de concatenação** (`..`). Por compatibilidade com versões anteriores também é possível usar o operador de adição (`+`), mas isso é desaconselhado.

```
$
a[1] = "abc";                   // a[1] é "abc"
a[1] = "xyz" .. a[1] .. "xyz";  // a[1] é "xyzabcxyz"
```

## Constantes de string importantes

Alguns caracteres especiais são difíceis (ou impossíveis) de representar com as sintaxes acima. Para isso foram criadas algumas constantes, que podem ser anexadas à string com o operador de concatenação:

| Nome | Descrição |
|---|---|
| `QUOTE` | Contém o caractere de **aspas duplas** |
| `CRLF` | Quebra de linha no estilo Windows |
| `LF` | Quebra de linha no estilo Linux |
| `CR` | Quebra de linha no estilo Macintosh |

GUIA

# Operadores

Para realizar cálculos, atribuições, comparações etc. com um DestinyScript, é preciso usar **operadores**. Distinguem-se os seguintes tipos de operadores:

- Operadores de atribuição
- Operadores aritméticos
- Operadores de string
- Operadores binários
- Operadores lógicos
- Operadores de comparação
- Operadores de objeto
- Sinais (positivo e negativo)

Esses operadores também têm **prioridades** diferentes (por exemplo, multiplicação e divisão antes de soma e subtração). Mais detalhes no capítulo sobre Expressões de fórmula.

GUIA

## Operadores de atribuição

Os operadores de atribuição servem para atribuir outro valor a um valor de memória. O valor de memória (= **destino**) fica à esquerda do operador e precisa ser gravável. O valor a atribuir (= **origem**; a seguir chamado *valor de origem*) fica à direita do operador e também pode ser uma expressão calculada. As atribuições também podem ser feitas no meio de uma expressão ou de forma aninhada.

Os operadores de atribuição também podem ser uma **mistura** do operador de atribuição comum com um operador de cálculo. Nesse caso, primeiro o valor de memória é combinado com o valor a atribuir e só então o resultado é atribuído. Por exemplo, `a *= b + c` deve ser entendido como `a = a * (b + c)`.

| Operador | Nome | Descrição |
|---|---|---|
| `=` | Atribuição | Atribui o valor de origem ao valor de memória, sem alterar o valor. |
| `+=` | Adição/atribuição | Soma o valor de origem ao valor de memória e atribui o resultado ao valor de memória. Esse operador também pode ser usado para concatenar strings (isso, porém, não é recomendado). |
| `-=` | Subtração/atribuição | Subtrai o valor de origem do valor de memória e atribui o resultado ao valor de memória. |
| `*=` | Multiplicação/atribuição | Multiplica o valor de origem pelo valor de memória e atribui o resultado ao valor de memória. |
| `/=` | Divisão/atribuição | Divide o valor de memória pelo valor de origem e atribui o resultado ao valor de memória. Uma divisão por 0 causa um erro. |
| `%=` | Módulo/atribuição | Divide o valor de memória pelo valor de origem e atribui ao valor de memória a parte que forma as casas decimais (para inteiros, o resto não divisível da divisão). Uma divisão por 0 causa um erro. |
| `..=` | Concatenação/atribuição | Anexa o valor de origem ao valor de memória (como string) e atribui o resultado ao valor de memória. |
| `^=` | OU exclusivo binário/atribuição | Executa uma operação binária OU exclusivo (XOR) entre o valor de origem e o valor de memória e atribui o resultado ao valor de memória. |
| `\|=` | OU binário/atribuição | Executa uma operação binária OU entre o valor de origem e o valor de memória e atribui o resultado ao valor de memória. |
| `&=` | E binário/atribuição | Executa uma operação binária E entre o valor de origem e o valor de memória e atribui o resultado ao valor de memória. |
| `<<=` | Deslocamento aritmético à esquerda/atribuição | Desloca os bits do valor de memória para a esquerda pela quantidade de bits indicada no valor de origem e atribui o resultado ao valor de memória. Os novos bits são preenchidos com 0. Um eventual sinal é mantido (desde que não ocorra estouro). |
| `>>=` | Deslocamento aritmético à direita/atribuição | Desloca os bits do valor de memória para a direita pela quantidade de bits indicada no valor de origem e atribui o resultado ao valor de memória. Os novos bits são preenchidos com o antigo bit mais alto do valor de memória. Um eventual sinal é mantido. |
| `\|\|=` | OU lógico/atribuição | Executa uma operação lógica OU entre o valor de origem e o valor de memória e atribui o resultado ao valor de memória. |
| `&&=` | E lógico/atribuição | Executa uma operação lógica E entre o valor de origem e o valor de memória e atribui o resultado ao valor de memória. |

GUIA

## Operadores aritméticos

Os operadores aritméticos servem para combinar dois números entre si. Como operandos só podem ser usados **números** (com exceção do `+` com strings). O primeiro operando fica à esquerda do operador e o segundo à direita. Ambos os operandos também podem ser expressões calculadas. Todos os operadores citados aqui também existem como operadores de atribuição.

| Operador | Nome | Descrição |
|---|---|---|
| `+` | Adição | Soma o primeiro operando (parcela) ao segundo (parcela) e devolve o resultado (soma). Esse operador também pode ser usado para concatenar strings (isso, porém, não é recomendado). |

| Operador | Nome | Descrição |
|---|---|---|
| `-` | Subtração | Subtrai o segundo operando (subtraendo) do primeiro (minuendo) e devolve o resultado (diferença). |
| `*` | Multiplicação | Multiplica o primeiro operando (fator) pelo segundo (fator) e devolve o resultado (produto). |
| `/` | Divisão | Divide o primeiro operando (dividendo) pelo segundo (divisor) e devolve o resultado (valor do quociente). Uma divisão por 0 causa um erro. |
| `%` | Módulo | Divide o primeiro operando (dividendo) pelo segundo (divisor) e devolve como resultado (módulo) a parte que forma as casas decimais (para inteiros, o resto não divisível). Uma divisão por 0 causa um erro. |

GUIA

## Operadores de string

Os operadores de string servem para combinar strings entre si. Como operandos só podem ser usados **strings e números**. O primeiro operando fica à esquerda do operador e o segundo à direita. Ambos os operandos também podem ser expressões calculadas. Todos os operadores citados aqui também existem como operadores de atribuição.

| Operador | Nome | Descrição |
|---|---|---|
| `..` | Concatenação | Anexa o segundo operando ao primeiro (como string) e devolve o resultado. |

GUIA

## Operadores binários

Os operadores binários (bit a bit) servem para combinar números inteiros por meio de operações lógicas binárias. Como operandos só podem ser usados **números inteiros** (Dword, Word, Byte). O primeiro operando fica à esquerda do operador e o segundo à direita. Ambos os operandos também podem ser expressões calculadas. Todos os operadores citados aqui também existem como operadores de atribuição.

| Operador | Nome | Descrição |
|---|---|---|
| `^` | OU exclusivo (XOR) | Executa uma operação binária OU exclusivo entre os dois operandos e devolve o resultado. |
| `\|` | OU (OR) | Executa uma operação binária OU entre os dois operandos e devolve o resultado. |
| `&` | E (AND) | Executa uma operação binária E entre os dois operandos e devolve o resultado. |
| `<<` | Deslocamento aritmético à esquerda | Desloca os bits do primeiro operando para a esquerda pela quantidade indicada no segundo operando e preenche os novos bits com 0. Um eventual sinal é mantido (desde que não ocorra estouro). |
| `>>` | Deslocamento aritmético à direita | Desloca os bits do primeiro operando para a direita pela quantidade indicada no segundo operando e preenche os novos bits com o antigo bit mais alto do primeiro operando. Um eventual sinal é mantido. |

GUIA

## Operadores lógicos

Os operadores lógicos servem para combinar valores booleanos (mas também números e strings) por meio de operações lógicas. Todos os operandos são tratados como valores booleanos:

- Se um número for diferente de 0, o valor é `true`; caso contrário, `false`.
- Se uma string não estiver vazia, o valor é `true`; caso contrário, `false`.

O primeiro operando fica à esquerda do operador e o segundo à direita. Ambos os operandos também podem ser expressões calculadas. Todos os operadores citados aqui também existem como operadores de atribuição.

| Operador | Nome | Descrição |
|---|---|---|
| `\|\|` | OU lógico (OR) | Executa uma operação lógica OU entre os dois operandos e devolve o resultado. |
| && | E lógico (AND) | Executa uma operação lógica E entre os dois operandos e devolve o resultado. |

## Exemplo

```
$
// OU
s[1] = false || false;   // false
s[2] = false || true;    // true
s[3] = true  || false;   // true
s[4] = true  || true;    // true

// E
s[5] = false && false;   // false
s[6] = false && true;    // false
s[7] = true  && false;   // false
s[8] = true  && true;    // true
```

GUIA

# Operadores de comparação

Os operadores de comparação servem para comparar valores entre si. Podem ser comparados todos os tipos de dados válidos (Dword, Word, Byte, Float, String, Bool). O resultado de uma comparação é sempre um valor **booleano**. O primeiro operando fica à esquerda do operador e o segundo à direita. Ambos os operandos também podem ser expressões calculadas.

| Operador | Nome | Descrição |
|---|---|---|
| == | Igual | Verifica se os dois operandos são iguais (true: iguais; false: diferentes). |
| != | Diferente | Verifica se os dois operandos são diferentes (true: diferentes; false: iguais). |
| >= | Maior ou igual | Verifica se o primeiro operando é maior ou igual ao segundo (true: o primeiro é maior ou igual; false: o primeiro é menor). |
| <= | Menor ou igual | Verifica se o primeiro operando é menor ou igual ao segundo (true: o primeiro é menor ou igual; false: o primeiro é maior). |
| > | Maior que | Verifica se o primeiro operando é maior que o segundo (true: o primeiro é maior; false: o primeiro é menor ou igual). |
| < | Menor que | Verifica se o primeiro operando é menor que o segundo (true: o primeiro é menor; false: o primeiro é maior ou igual). |

## Exemplo

```
$
s[1] = 3 == 3;  // true
s[2] = 3 != 3;  // false
s[3] = 5 >= 4;  // true
s[4] = 5 <= 4;  // false
s[6] = 1 >  2;  // false
s[7] = 2 <  5;  // true
```

GUIA

# Operadores de objeto

Os operadores de objeto servem para acessar determinadas propriedades/métodos de objetos e, se necessário, informar parâmetros.

> **Informação:** mais detalhes sobre objetos estão na Lista de objetos.

| Operador | Nome | Descrição |
|---|---|---|
| `.` | Propriedade/método subordinado | Acessa uma propriedade/método subordinado. O valor subordinado pode ser, ele próprio, um objeto. |
| `(` *parâmetro 1*, ..., *parâmetro n* `)` | Chamada | Chama um método. Os parâmetros (se houver) são informados entre os parênteses. Vários parâmetros são separados por vírgulas. |
| `[` *parâmetro 1*, ..., *parâmetro n* `]` | Elemento subordinado | Acessa um elemento subordinado de uma propriedade/de um objeto. Os parâmetros (é sempre necessário pelo menos um) são informados entre os colchetes. |

## Exemplo

```
$

// Acesso a uma propriedade
v[1] = Time.Tick;                      // Atual…

// Acesso a um método subordinado (e chamada)
v[2] = Keyboard.GetKeyState(VK_LEFT);  // Consulta…

// Acesso a um elemento subordinado de uma propriedade
v[3] = v[2];                           // Acesso…

// Acesso a um objeto subordinado
v[4] = Actor[1].Default.Level;         // Padrão…
```

GUIA

# Sinais (positivo e negativo)

Os sinais (e "pós-sinais") servem para alterar um valor antes de uma operação. Existem quatro sinais "clássicos", que alteram o valor seguinte mas deixam **inalterado** o valor de memória eventualmente usado. Além disso, existem dois sinais (pré/pós) que, se for o caso, alteram o valor seguinte, mas **sempre** alteram o valor de memória obrigatoriamente usado.

Os sinais "clássicos" podem ser combinados com os sinais que alteram valores, mas os sinais "clássicos" não podem ser combinados entre si.

| Operador | Nome | Descrição |
|---|---|---|
| `+` | Mais (identidade) | Se um mais vem antes de um número e já há um operador em uso, esse mais é tratado como sinal. O número não é alterado; a vantagem desse sinal é que números podem ser copiados diretamente para o DestinyScript sem precisar remover um eventual sinal. Pode ser usado com todos os tipos de dados numéricos. |
| `-` | Menos | Se um menos vem antes de um número e já há um operador em uso, esse menos é tratado como sinal. O valor do número é invertido (mais vira menos e menos vira mais; para inteiros, trata-se do complemento de dois). Pode ser usado com todos os tipos de dados numéricos. |

| Operador | Nome | Descrição |
|---|---|---|
| `~` | NÃO binário (NOT) | Usado para negar o valor de um inteiro (complemento de um): cada bit 1 vira bit 0 e cada bit 0 vira bit 1. Só pode ser usado com tipos de dados inteiros. |
| `!` | NÃO lógico (NOT) | Usado para inverter o valor lógico de uma expressão: true vira false e false vira true. Pode ser usado com todos os tipos de dados; verifica-se se o valor é diferente de 0 e o resultado é invertido (exceção nas strings: verifica-se se a string não está vazia; em strings, esse sinal não pode ser usado com valores informados diretamente). |
| `++` | Incremento | Só pode ser usado com números e pode vir antes ou depois do valor. O valor precisa ser gravável. Se o incremento vem **antes** do valor, o valor de memória é aumentado em 1 e depois devolvido como resultado. Se vem **depois**, o valor de memória é aumentado em 1 e o valor original de memória é devolvido como resultado. |
| `--` | Decremento | Só pode ser usado com números e pode vir antes ou depois do valor. O valor precisa ser gravável. Se o decremento vem **antes** do valor, o valor de memória é reduzido em 1 e depois devolvido como resultado. Se vem **depois**, o valor de memória é reduzido em 1 e o valor original de memória é devolvido como resultado. |

## Exemplo

```
$

// Mais
v[1] = +3;          // 3

// Menos
v[2] = -3;          // -3
v[3] = -v[2];       // 3

// NÃO binário
v[4] = ~5;          // -6

// NÃO lógico
s[1] = !false;      // true
s[2] = !true;       // false
s[3] = !3;          // false
s[4] = !0;          // true

// Incremento
v[10] = 100;
v[11] = v[10]++;    // 100 em v[11], 101 em v[10]
v[12] = ++v[10];    // 102 em v[12], 102 em v[10]

// Decremento
v[10] = 100;
v[13] = v[10]--;    // 100 em v[13], 99 em v[10]
v[14] = --v[10];    // 98 em v[14],  98 em v[10]
```

GUIA

# Expressões de fórmula

Depois de conhecer os tipos de dados, a escrita de números e strings, os operadores e os comentários, já se sabe quase tudo o que é preciso para usar expressões de fórmula no DestinyScript e, assim, escrever um script linear (que roda de cima para baixo). O que falta são as **prioridades dos operadores** e o uso de **parênteses**.

## Prioridades dos operadores

Como se sabe, na álgebra matemática vale: multiplicação e divisão antes de soma e subtração. Alguns também conhecem a regra da álgebra booleana: E antes de OU. Essas regras também valem no DestinyScript, mas precisam ser ampliadas (por causa da maior quantidade de operadores). Por isso os operadores têm certas **prioridades**, que indicam a ordem em que as operações são executadas. Quando operadores de mesma prioridade aparecem lado a lado, a operação da esquerda é executada primeiro.

### Lista de prioridades

Na tabela a seguir, vale: quanto **maior** a prioridade, mais cedo a operação é executada (por exemplo, "vezes" tem prioridade maior que "mais"):

| Operador | Nome | Prioridade |
|---|---|---|
| `=` (e combinações) | Operador de atribuição | 0 |
| `\|\|` | OU lógico (OR) | 1 |
| `&&` | E lógico (AND) | 2 |
| `==` | Igual | 3 |
| `!=` | Diferente | 3 |
| `>` | Maior que | 3 |
| `>=` | Maior ou igual | 3 |
| `<` | Menor que | 3 |
| `<=` | Menor ou igual | 3 |
| `..` | Concatenação | 4 |
| `+` | Adição | 5 |
| `-` | Subtração | 5 |
| `*` | Multiplicação | 6 |
| `/` | Divisão | 6 |
| `%` | Módulo | 6 |
| `<<` | Deslocamento (aritmético) à esquerda | 7 |
| `>>` | Deslocamento aritmético à direita | 7 |
| `\|` | OU binário (OR) | 7 |
| `^` | OU exclusivo binário (XOR) | 7 |
| `&` | E binário (AND) | 8 |
| `+` | Sinal de mais (identidade) | 9 |
| `-` | Sinal de menos | 9 |
| `!` | Sinal de NÃO lógico (NOT) | 9 |

| Operador | Nome | Prioridade |
|---|---|---|
| `~` | Sinal de NÃO binário (NOT) | 9 |
| `++` | Incremento | 10 |
| `--` | Decremento | 10 |
| `.` | Propriedade/método subordinado | 11 |
| `( parâmetro 1, ..., parâmetro n )` | Chamada | 11 |
| `[ parâmetro 1, ..., parâmetro n ]` | Elemento subordinado | 11 |

### Exemplo

```
$
v[1] = 1 + 2 * 3;    // =  7
v[2] = 4 | 7 & 1;    // =  5
v[3] = ~++v[2];      // = -7
```

## Uso de parênteses

Às vezes se quer executar operações em uma ordem diferente da prescrita pelas prioridades. Nesse caso, a parte correspondente deve ser escrita entre **parênteses** `( )`. A parte da expressão de fórmula que está entre parênteses é calculada separadamente.

> **Informação:** um sinal antes de um parêntese refere-se apenas ao resultado do cálculo entre parênteses.

### Exemplo

```
$
v[1] = 3 * (2 + 5);   // =  21
v[2] = 3 * -(2 + 5);  // = -21
```

GUIA

# Elementos de estrutura

Até este ponto, todos os DestinyScripts rodavam de forma **linear**. Se agora se quer executar algumas partes do código apenas sob condição, ou de forma repetida, é preciso recorrer aos **elementos de estrutura**. Quem já trabalhou com outras linguagens de programação, como C++, Visual Basic ou PHP, leva vantagem, pois a sintaxe básica foi emprestada dessas linguagens.

Basicamente, o DestinyScript distingue os seguintes tipos de elementos de estrutura:

- **Ramificações** — execução condicional de trechos de código, sem repetição.
- **Laços de repetição** — execução condicional de trechos de código, com repetição.
- **Comandos de salto** — interrupção do fluxo do programa.

GUIA

## Ramificações

As ramificações são os elementos de estrutura mais usados. Servem para que apenas um determinado trecho de código (dentre uma quantidade definida de trechos) seja executado. O fluxo linear do programa é, portanto, "ramificado".

No DestinyScript existem as seguintes ramificações:

- **Ramificações simples** (`if … elseif … else … endif`)
- **Ramificações múltiplas** (`switch … case … default … endswitch`)

GUIA

# Ramificação simples (If)

## if ... endif

O `if` (do inglês "se") é o elemento de estrutura mais usado. Depois do `if`, escreve-se entre parênteses uma expressão de fórmula que possa ser resolvida em um valor. Se essa expressão for **verdadeira** (ou seja, números diferentes de 0 e strings não vazias), o código que está abaixo é executado. O bloco assim iniciado termina com um `endif` (escrito junto).

```
$
if(v[1] >= 3)
  // Código condicional
  v[2] = 7;
endif;
```

Na linha 2 inicia-se um bloco If, que termina na linha 5. O código das linhas 3 e 4 só é executado se a expressão de fórmula (depois do `if`) for verdadeira. Aqui, a expressão verifica se o valor da primeira variável é maior ou igual a 3:

- Se a primeira variável for maior ou igual a 3 (por exemplo, 17), o valor da segunda variável é definido como 7.
- Se a primeira variável for menor que 3 (por exemplo, 1), o valor da segunda variável não é alterado.

## if ... else ... endif

Às vezes se quer indicar exatamente dois códigos possíveis:

- um código executado quando uma condição é satisfeita;
- um código executado quando **nenhuma** condição é satisfeita.

Para isso existe o elemento de estrutura adicional `else` (do inglês "caso contrário"). Depois do `else` não há condição; o código que está ali é sempre executado quando nenhuma das condições anteriores do bloco If for satisfeita. O `else` precisa ser sempre a **última** seção do bloco If antes do `endif`, e um bloco If só pode conter uma seção Else.

```
$
if(v[1] >= 3)
  // Código condicional
  v[2] = 7;
else
  // Código condicional
  v[2] = 5;
endif;
```

Na linha 5 inicia-se a seção Else. O código da seção If (linhas 3 e 4) é executado se a condição indicada (primeira variável maior ou igual a 3) for verdadeira. O código da seção Else (linhas 6 e 7) é executado se as condições acima do bloco If (aqui, primeira variável maior ou igual a 3) **não** forem verdadeiras:

- Se a primeira variável for maior ou igual a 3 (por exemplo, 17), o valor da segunda variável é definido como 7.
- Se a primeira variável for menor que 3 (por exemplo, 1), o valor da segunda variável é definido como 5.

## if ... elseif ... else ... endif

Como já se insinuou acima, é possível indicar várias seções condicionais em um bloco If. Para isso existe o elemento de estrutura adicional `elseif` (escrito junto). A sintaxe do ElseIf é igual à do If, com a diferença de que, dentro de um bloco If, podem aparecer **várias** seções ElseIf. As seções ElseIf precisam ficar depois da seção If e antes da seção Else (se houver). Se um bloco If contém várias seções (If, ElseIf e/ou Else), **apenas uma** delas é executada. Se não houver seção Else, pode até acontecer de nenhuma seção ser executada.

```
$
if(v[1] >= 3)
  // Código condicional
  v[2] = 7;
elseif(v[1] <= -5)
  // Código condicional
  v[2] = 1;
elseif(v[1] <= -2)
  // Código condicional
  v[2] = 4;
elseif(v[1] <= -8)
  // Código condicional (nunca é executado)
  v[2] = 3;
else
  // Código condicional
  v[2] = 5;
endif;
```

Este exemplo foi complementado com três seções ElseIf:

1. Na linha 5 verifica-se se a primeira variável é menor ou igual a -5. Se for, o valor da segunda variável é definido como 1 (linhas 6 e 7).
2. Na linha 8 verifica-se se a primeira variável é menor ou igual a -2. Se for, o valor da segunda variável é definido como 4 (linhas 9 e 10).
3. Na linha 11 verifica-se se a primeira variável é menor ou igual a -8. Essa condição **nunca** pode ser satisfeita (mesmo que a primeira variável seja menor que -8), pois nas linhas 5 e 8 já se verificou se a primeira variável é menor ou igual a -5 ou a -2. Se for o caso, o respectivo código condicional é executado e, em seguida, o bloco If é abandonado. A condição de a primeira variável ser menor ou igual a -8 não pode, portanto, ser satisfeita nesse ponto.

A seção Else (linha 14) só é executada quando as condições acima do bloco If não se aplicam. Isso só acontece quando a primeira variável está na faixa de -1 a +2.

## Várias condições em uma seção

Também é possível informar várias condições na expressão de fórmula de uma seção (com condição; ou seja, If e/ou ElseIf). Para isso, normalmente se combinam várias comparações com os operadores lógicos:

- `&&` (E) é usado quando várias condições devem ser satisfeitas;
- `||` (OU) é usado quando uma (ou mais) entre várias condições deve ser satisfeita;
- `!` (NÃO; como sinal) é usado quando a condição seguinte deve ser invertida (ou seja, a condição **não** deve ser satisfeita).

As condições também podem ser agrupadas entre parênteses, como descrito no capítulo sobre Expressões de fórmula.

```
$
if(!(v[1] == 3))
  // Só é executado se v[1] NÃO for 3
endif;
```

```
if(v[1] >= 3 && v[1] <= 5)
  // Só é executado se v[1] for maior ou igual a 3 E menor ou igual a 5
endif;

if(v[1] == 3 || v[1] == 4)
  // Só é executado se v[1] for igual a 3 OU igual a 4
endif;

if((v[1] == 3 || v[1] == 4) && !(v[2] == 5))
  // Só é executado se v[1] for 3 OU 4, e v[2] NÃO for 5
endif;
```

## Blocos If aninhados

Em princípio, todos os elementos de estrutura podem ser **aninhados**. Isso também vale para blocos If, de modo que alguns blocos If só são executados se um bloco If de nível superior for satisfeito:

```
$
if(v[1] == 3)
  v[2]++;
  if(v[2] == 10)
    v[3]++;
    v[2] = 0;
  endif;
endif;
```

O bloco If das linhas 4 a 7 só é executado se a seção superior do bloco If (linhas 2 a 8) for executada.

GUIA

# Ramificação múltipla (Switch)

## switch ... case ... endswitch

O `switch` (do inglês "chavear") serve para saltar, conforme um valor, para uma determinada seção do bloco Switch. A passagem entre as seções é **contínua** (ou seja, ao fim de uma seção, por padrão, o conteúdo da seção seguinte é executado). Depois do `switch`, escreve-se entre parênteses uma expressão de fórmula que possa ser resolvida em um valor. Esse valor precisa ser um inteiro, um número com vírgula, uma string ou um bool.

Dentro do bloco Switch vêm então uma ou mais seções `case` (do inglês "caso"), que indicam pontos de entrada dentro do bloco. Depois do `case` vem um **valor fixo** (não uma expressão de fórmula!) seguido de dois-pontos. Se a expressão de fórmula comparada (depois do `switch`) for igual ao valor (depois do `case`), o código dessa seção é executado. Se nenhuma seção corresponder à expressão comparada, nenhuma das seções do bloco Switch é executada e o código continua depois do bloco Switch. O bloco Switch termina com um `endswitch` (escrito junto).

```
$
v[2] = 0;
switch(v[1])
  case 0:
    // Código condicional (se v[1] for igual a 0)
    v[2]++;
  case 1:
    // Código condicional (se v[1] for igual a 1 ou se…)
    v[2]++;
endswitch;
```

Na linha 3 inicia-se um bloco Switch, que termina na linha 10. O código das linhas 4 a 6 só é executado se a expressão de fórmula (depois do `switch`) for igual ao número 0. O código das linhas 7 a 9 só é executado se a expressão for igual ao número 1 **ou** se a seção anterior (linhas 4 a 7) tiver sido executada.

- Se a primeira variável for 0, o valor da segunda é definido como 2 (por dois incrementos).
- Se a primeira variável for 1, o valor da segunda é definido como 1 (por um incremento).
- Se a primeira variável não for nem 0 nem 1, nada acontece.

### Saindo de um bloco Switch

Na maioria dos casos não se quer que, depois da seção de cima, as seções seguintes também sejam executadas. Normalmente se quer que haja seções bem definidas dentro do bloco Switch para cada valor (ou grupo de valores). Para isso é preciso **sair do bloco Switch antes do fim**. Existe para isso o elemento de estrutura `break` (do inglês "interromper"). Se houver um `break` em uma linha dentro do bloco Switch, o bloco é abandonado naquele ponto. Se abaixo desse `break` houver outras seções Case, elas naturalmente ainda podem ser alcançadas, pois o bloco Switch não termina no `break`; apenas, durante a execução, o bloco é abandonado no ponto do `break`. Se um `break` for escrito na última linha de cada seção, cada seção só é executada para os valores definidos.

```
$
v[2] = 0;
switch(v[1])
  case 0:
    // Código condicional (se v[1] for igual a 0)
    v[2]++;
    break;
  case 1:
    // Código condicional (se v[1] for igual a 1)
    v[2]++;
    break;
endswitch;
```

Ao contrário do exemplo anterior, aqui o bloco Switch é abandonado no fim da primeira seção (linhas 4 a 7; para v[1] igual a 0). Assim, não é executada em seguida, como no exemplo anterior, a seção seguinte (linhas 9 a 12; para v[1] igual a 1).

### Vários casos para uma seção

Às vezes uma determinada seção (que termina com `break`) deve valer para vários valores. Basta combinar os dois comportamentos (sair de uma seção com `break` e continuar para a seção seguinte sem `break`):

```
$
switch(v[1])
  case 0: case 1:
    // Código condicional (se v[1] for igual a 0 ou a 1)
    v[2] = 5;
    break;
  case 2: case 3: case 4: case 5:
    // Código condicional (se v[1] for igual a 2, 3, 4 ou 5)
    v[2] = 17;
    break;
endswitch;
```

## switch ... case ... default ... endswitch

Outro caso de uso frequente é quando se quer que uma determinada seção seja executada **se nenhuma** das seções Case definidas se aplicar. Para isso existe a seção `default` (do inglês "caso padrão"), que precisa ser a **última** seção dentro do bloco Switch e só pode aparecer uma vez por bloco Switch.

```
$
switch(v[1])
  case 0:
    // Código condicional (se v[1] for igual a 0)
    v[2] = 11;
```

```
    break;
  case 1:
    // Código condicional (se v[1] for igual a 1)
    v[2] = 31;
    break;
  default:
    // Código condicional (se v[1] não for nem 0 nem 1)
    v[2] = 17;
endswitch;
```

Neste exemplo, a seção Default só é executada se todas as outras seções Case do bloco Switch não se aplicarem.

- Se a primeira variável for 0, o valor da segunda é definido como 11.
- Se a primeira variável for 1, o valor da segunda é definido como 31.
- Em todos os outros casos (primeira variável menor que 0 ou maior que 1), o valor da segunda é definido como 17.

## Semelhanças com as ramificações simples (if ... elseif ... else ... endif)

Se um `break` for usado no fim de cada seção Case, um bloco Switch desse tipo também pode ser escrito como bloco If (e vice-versa).

```
$
// Exemplo como Switch
switch(v[1])
  case 0:
    // Código condicional (se v[1] for igual a 0)
    v[2] = 11;
    break;
  case 1: case 2:
    // Código condicional (se v[1] for igual a 1 ou a 2)
    v[2] = 31;
    break;
  default:
    // Código condicional (se v[1] não for 0, 1 nem 2)
    v[2] = 17;
endswitch;

// Exemplo como If
if(v[1] == 0)
  // Código condicional (se v[1] for igual a 0)
  v[2] = 11;
elseif(v[1] == 1 || v[1] == 2)
  // Código condicional (se v[1] for igual a 1 ou a 2)
  v[2] = 31;
else
  // Código condicional (se v[1] não for 0, 1 nem 2)
  v[2] = 17;
endif;
```

O bloco Switch tem as seguintes **vantagens** em relação ao bloco If:

- O código é mais legível.
- A execução do código é um pouco mais rápida, pois o valor depois do `switch` não precisa ser avaliado a cada vez.
- É possível (se desejado) fazer várias seções "escorrerem" umas para as outras (omitindo o `break`).

Em compensação, o bloco Switch tem as seguintes **desvantagens** em relação ao bloco If:

- Só é possível comparar **um** valor com vários outros valores. Comparações mais complexas, como "todos os valores maiores que 3", não são possíveis.

- Se você esquecer um `break`, podem ocorrer erros no fluxo do programa, pois omitir o `break` não é um erro de sintaxe.

GUIA

# Laços de repetição

Os laços (loops) são elementos de estrutura usados para executar um determinado trecho de código de forma **repetida** ("em laço"). O fluxo linear do programa ganha, portanto, um retorno.

No DestinyScript existem os seguintes laços:

- **Laços com teste no início ou no fim** (`do while/until … loop` e `do … loop while/until`)
- **Laços de contagem** (`for … next`)

GUIA

# Laços com teste no início ou no fim

## do while/until ... loop

Nos laços **com teste no início** (kopfgesteuerte Schleifen), o controle fica na "cabeça" (isto é, no começo do laço). No DestinyScript, um laço com teste no início é iniciado por um `do` (do inglês "faça"), seguido de um `while` (do inglês "enquanto") ou de um `until` (do inglês "até que"). Depois do `while` ou `until` escreve-se, entre parênteses, uma expressão de fórmula que possa ser resolvida em um valor. Para que o laço seja iniciado e/ou repetido, esse valor precisa ser **verdadeiro** no caso do `while` e **falso** no caso do `until`. Se a condição se aplicar (verdadeira no `while`, falsa no `until`), o código abaixo é executado. O bloco assim iniciado termina com um `loop` (do inglês "laço").

### do while ... loop

```
$
v[1] = 10000;
v[2] = 1;
do while(v[1] > 1)
  v[2]++;
  v[1] /= 10;
loop;
```

Neste exemplo, a primeira variável é definida como 10000 e a segunda como 1 (linhas 2 e 3). Em seguida vem um laço com teste no início, que é executado enquanto o valor da primeira variável for maior que 1 (linhas 4 a 7). O código dentro do laço (linhas 5 e 6) é executado várias vezes (aqui, quatro vezes).

### do until ... loop

```
$
v[1] = 10000;
v[2] = 1;
do until(v[1] > 1)
  v[2]++;
  v[1] /= 10;
loop;
```

Este exemplo equivale ao anterior, com a diferença de que o `while` foi substituído por um `until`. Aqui o laço **não é executado nenhuma vez**, porque a condição (a primeira variável não pode ser maior que 1) não é satisfeita. Um laço controlado por `until` nada mais é do que um laço controlado por `while` cuja condição foi simplesmente invertida por um NÃO lógico (`!`).

## do ... loop while/until

Ao contrário dos laços com teste no início, nos laços **com teste no fim** (fußgesteuerte Schleifen) a condição fica no "pé" (isto é, no final do laço). A diferença entre eles é apenas que os laços com teste no fim são percorridos **pelo menos uma vez** (pois a condição só é verificada no final). Nos laços com teste no início, a condição é verificada já na primeira passagem, de modo que pode acontecer de o conteúdo do laço não ser executado nenhuma vez (veja o exemplo anterior).

### do ... loop while

```
$
v[1] = 0;
v[2] = 1;
do
  v[2] *= 2;
  v[1]++;
loop while(v[1] == 10);
```

Neste exemplo, no início a primeira variável é definida como 0 e a segunda como 1 (linhas 2 e 3). O laço com teste no fim (linhas 4 a 7) é percorrido pelo menos uma vez, pois a condição (a primeira variável precisa ser 10) só é verificada no final do laço. Dentro do laço (linhas 5 e 6), primeiro o conteúdo da segunda variável é dobrado e depois o conteúdo da primeira é aumentado em 1. Como, após a primeira passagem, a primeira variável tem o valor 1 (e não 10), o laço não é repetido, mas abandonado (e o código seguinte, se houver, é executado).

### do ... loop until

```
$
v[1] = 0;
v[2] = 1;
do
  v[2] *= 2;
  v[1]++;
loop until(v[1] == 10);
```

Este exemplo equivale ao anterior, com a diferença de que o `while` foi substituído por um `until`. Ao contrário do exemplo anterior, este laço é percorrido **dez vezes** ao todo, pois a condição (a primeira variável não pode ser 10) só deixa de ser satisfeita depois de dez passagens (quando a primeira variável chega a dez).

## Saindo antes do laço (break)

Se um laço deve ser abandonado imediatamente (sem verificar a condição novamente), pode-se usar o elemento de estrutura `break` (do inglês "interromper"). Se ele for usado dentro do laço, o laço é abandonado naquele ponto. O código do laço que eventualmente esteja depois do `break` não é executado. O `break` também pode estar "encapsulado" em outros elementos de estrutura (por exemplo, dentro de um `if`).

```
$
v[1] = 0;
v[2] = 0;
do while(v[1] < 10)
  v[1]++;
  if(v[1] == 3)
    break;
  endif;
  v[2]++;
loop;
```

Neste exemplo é executado um laço que, aparentemente, seria percorrido dez vezes. A primeira variável (inicialmente 0) é aumentada em 1 a cada passagem, e a condição do laço diz que ela precisa ser menor que 10.

Na verdade, porém, o laço só é executado **três vezes**, pois dentro dele verifica-se se a primeira variável tem o valor 3 (linha 6) e, se for o caso, o laço é abandonado (linha 7). Como essa condição já é satisfeita após três passagens, o laço é abandonado nesse ponto. No fim do código, a primeira variável é 3 e a segunda é 2 (o código depois do `break` não foi mais executado ao sair do laço).

## Repetindo o laço antes do fim (continue)

Se um laço deve ser repetido imediatamente (e a condição verificada novamente), pode-se usar o elemento de estrutura `continue` (do inglês "continuar"). Se ele for usado dentro do laço, a condição do laço é verificada de novo e, se ela se aplicar, a execução prossegue na primeira linha do laço. Se a condição não se aplicar, o laço é abandonado. Em qualquer caso, o código que estiver depois do `continue` não é mais executado nessa passagem.

```
$
v[1] = 0;
v[2] = 0;
do while(v[1] < 10)
  v[1]++;
  if(v[1] == 3)
    continue;
  endif;
  v[2]++;
loop;
```

Este exemplo é parecido com o anterior, com a diferença de que se usa um `continue` (em vez de um `break`). Aqui o laço é realmente percorrido **dez vezes**. Na terceira passagem, porém, o código depois do `continue` (isto é, o aumento da segunda variável em 1) não é executado. No fim do código, a primeira variável é 10 e a segunda é 9.

GUIA

# Laços de contagem (For)

## for ... next

Os laços de contagem têm a tarefa de repetir um determinado código com base em uma **contagem**. São usados, por exemplo, para percorrer todos os eventos etc. (que são identificados por um número). No DestinyScript, um laço de contagem é iniciado por um `for` (do inglês "para") seguido de um par de parênteses. Dentro desse par de parênteses é preciso informar **três áreas**, separadas por ponto e vírgula:

1. **Área de inicialização** — executada antes de entrar no laço. Normalmente serve para definir uma variável (de contagem) com um valor inicial. O conteúdo dessa área pode ser omitido (então nada acontece antes da entrada).
2. **Área de condição** — aqui se informa uma expressão de fórmula que possa ser resolvida em um valor. Esse valor é verificado antes de cada passagem do laço; se for verdadeiro, o laço é iniciado/repetido; caso contrário, é abandonado. O conteúdo dessa área pode ser omitido (então a condição do laço está sempre satisfeita).
3. **Área de contagem** — contém uma expressão de fórmula que é executada depois de uma passagem do laço, antes da verificação da condição. Normalmente serve para aumentar ou diminuir a variável (de contagem). O conteúdo dessa área pode ser omitido (então nada acontece entre as passagens).

O bloco assim iniciado termina com um `next` (do inglês "próximo").

```
$
// Esvaziar a primeira string
a[1] = "";

// Executar um laço com dez execuções
for(v[1] = 0; v[1] < 10; v[1]++)
```

```
  a[1] ..= v[1] .. ";";
next;
```

Neste exemplo, no início a primeira string é esvaziada (linha 3). Em seguida vem um laço de contagem, que conta com a primeira variável de 0 até 10, exclusive (linhas 6 a 8). O código dentro do laço (linha 7) é executado dez vezes. No final, a primeira string conteria o texto `0;1;2;3;4;5;6;7;8;9;`.

O laço funciona assim:

1. Define a primeira variável como 0 (área de inicialização).
2. Verifica se a primeira variável é menor que 10 (área de condição).
3. Como a condição se aplica, o código interno é executado.
4. Depois de executado o código interno, a primeira variável é aumentada em 1 (área de contagem).
5. Em seguida verifica-se de novo se a primeira variável é menor que 10 (área de condição). Se sim, continua-se a partir do ponto 3; caso contrário, o laço de contagem termina aqui.

## Os laços de contagem mais usados

Existem muitas formas de laços de contagem, mas as seguintes são as mais usadas.

### *Laço crescente (começa em 0)*

Neste laço, a contagem começa em 0 (área de inicialização). A variável de contagem precisa ser **menor** que a quantidade de elementos (área de condição). Entre as passagens, a variável de contagem é aumentada em 1 (área de contagem).

```
$
for(v[1] = 0; v[1] < Client.Count; v[1]++)
  Client[v[1]].Close();
next;
```

Neste exemplo é executado um laço de contagem sobre todos os elementos do objeto `Client`.

> **Informação:** todos os objetos com índice que descrevem objetos da `Destiny.dll` começam com o índice **0**.

### *Laço crescente (começa em 1)*

Neste laço, a contagem começa em 1 (área de inicialização). A variável de contagem precisa ser **menor ou igual** à quantidade de elementos (área de condição). Entre as passagens, a variável de contagem é aumentada em 1 (área de contagem).

```
$
a[1] = "";
for(v[1] = 1; v[1] <= Actor.Count; v[1]++)
  a[1] ..= Actor[v[1]].Name .. ";";
next;
```

Neste exemplo é executado um laço de contagem sobre todos os elementos do objeto `Actor`.

> **Informação:** todos os objetos com índice que descrevem objetos do `RPG_RT` começam com o índice **1**. (A exceção é o objeto `MapEvent`, pois ele oferece uma visão diferente dos eventos.)

> **Observação da tradução:** no manual original o código desse exemplo usa `v[1] < Actor.Count`, mas o texto explica que a condição deve ser "menor ou igual"; aqui o código foi corrigido para `<=`, de acordo com a explicação.

### *Laço decrescente (termina em 0)*

Neste laço, a contagem começa com a quantidade de elementos menos 1 (área de inicialização). A variável de contagem precisa ser **maior ou igual a 0** (área de condição). Entre as passagens, a variável de contagem é diminuída em 1 (área de contagem).

```
$
for(v[1] = Client.Count - 1; v[1] >= 0; v[1]--)
  Client[v[1]].Close();
next;
```

Neste exemplo é executado um laço de contagem sobre todos os elementos do objeto `Client`.

### *Laço decrescente (termina em 1)*

Neste laço, a contagem começa com a quantidade de elementos (área de inicialização). A variável de contagem precisa ser **maior ou igual a 1** (área de condição). Entre as passagens, a variável de contagem é diminuída em 1 (área de contagem).

```
$
a[1] = "";
for(v[1] = Actor.Count; v[1] >= 1; v[1]--)
  a[1] ..= Actor[v[1]].Name .. ";";
next;
```

Neste exemplo é executado um laço de contagem sobre todos os elementos do objeto `Actor`.

### *Laço infinito*

Às vezes se quer um laço que rode "para sempre". Esses laços são chamados de **laços infinitos**; normalmente eles têm condições complexas demais para serem informadas na área de condição. Para isso usa-se o laço de contagem e simplesmente se omitem as três áreas.

```
$
for(;;)  // Laço infinito
  if(Keyboard.GetKeyState(VK_F10) != 0)  // Verifica se F10 está pressionada…
    Destiny.Debugger.Open();             // Se sim, abre o depurador…
  endif;
  pause;                                 // Deixa o resto do jogo funcionar…
next;
```

Este exemplo é executado "para sempre" (isto é, até o jogo ser encerrado ou o mapa atual ser abandonado). Dentro do laço, verifica-se com o objeto `Keyboard` se a tecla F10 está pressionada (linha 3). Se estiver, abre-se o DestinyDebugger (linha 4). Na linha 6 aparece o elemento de estrutura ainda desconhecido `pause`. Ele é descrito no próximo capítulo e é necessário aqui para que o jogo não congele (afinal, o laço infinito nunca é abandonado no código).

## Saindo antes do laço de contagem (break)

Se um laço de contagem deve ser abandonado imediatamente (sem verificar a condição novamente e sem executar a área de contagem), pode-se usar o elemento de estrutura `break` (do inglês "interromper"). Se ele for usado dentro do laço de contagem, o laço é abandonado naquele ponto. O código do laço que eventualmente esteja depois do `break` não é executado. O `break` também pode estar "encapsulado" em outros elementos de estrutura (por exemplo, dentro de um `if`).

```
$
v[2] = 0;
for(v[1] = 0; v[1] < 10; v[1]++)
  if(v[1] == 3)
```

```
    break;
  endif;
  v[2]++;
next;
```

Neste exemplo é executado um laço de contagem que, aparentemente, seria percorrido dez vezes. A primeira variável é definida como 0 na área de inicialização e aumentada em 1 após cada passagem. A condição do laço diz que a primeira variável precisa ser menor que 10. Na verdade, porém, o laço de contagem só é executado **quatro vezes**, pois dentro dele verifica-se se a primeira variável tem o valor 3 (linha 4) e, se for o caso, o laço é abandonado (linha 5). Como essa condição já é satisfeita após quatro passagens, o laço é abandonado nesse ponto. No fim do código, a primeira variável é 3 (depois de sair do laço por `break`, o valor da primeira variável não foi mais aumentado) e a segunda é 2 (o código depois do `break` não foi mais executado).

## Repetindo o laço de contagem antes do fim (continue)

Se um laço de contagem deve ser repetido imediatamente (com a área de contagem executada e a condição verificada novamente), pode-se usar o elemento de estrutura `continue` (do inglês "continuar"). Se ele for usado dentro do laço de contagem, primeiro é executada a área de contagem. Em seguida a condição do laço é verificada de novo e, se ela se aplicar, a execução prossegue na primeira linha do laço. Se a condição não se aplicar, o laço de contagem é abandonado. Em qualquer caso, o código que estiver depois do `continue` não é mais executado nessa passagem.

```
$
v[2] = 0;
for(v[1] = 0; v[1] < 10; v[1]++)
  if(v[1] == 3)
    continue;
  endif;
  v[2]++;
next;
```

Este exemplo é parecido com o anterior, com a diferença de que se usa um `continue` (em vez de um `break`). Aqui o laço de contagem é realmente percorrido **dez vezes**. Na quarta passagem (com `v[1]` igual a 3), porém, o código depois do `continue` (isto é, o aumento da segunda variável em 1) não é executado. No fim do código, a primeira variável é 10 e a segunda é 9.

> **Observação da tradução:** no manual original o código desse exemplo traz `break` (uma cópia do exemplo anterior), embora o texto descreva o uso de `continue`; aqui o código foi corrigido para `continue`.

GUIA

# Comandos de salto

Os comandos de salto são elementos de estrutura usados para **pular** um determinado trecho de código. Em princípio, o fluxo do programa, que de outro modo é linear, é alterado de modo que o código continue em outro ponto.

No DestinyScript existem os seguintes tipos de comandos de salto:

- **Comandos de salto para laços e ramificações** (`break` e `continue`)
- **Outros comandos de salto** (`exit` e `pause`)

GUIA

# Comandos de salto para laços e ramificações

## Funções básicas

Os comandos de salto para laços/ramificações (`break` e `continue`) já foram descritos nos capítulos anteriores, mas são mais poderosos do que os exemplos até agora indicaram.

Resumindo, a funcionalidade básica desses elementos de estrutura é:

- Em **laços**, os comandos de salto servem para abandonar o laço (`break`) ou repeti-lo (`continue`).
- Em **ramificações** (apenas `switch`), o `break` serve para abandonar a ramificação. O `continue` se refere ao laço de nível superior (não ao `switch`).

## Uso em laços aninhados

O uso desses elementos de estrutura só se torna problemático quando eles são usados dentro de código "aninhado" (por exemplo, um laço dentro de outro laço). Se, por exemplo, se quiser abandonar o laço **externo** por meio de um `break` no laço **interno**, isso não é possível com a sintaxe normal. Para usar `break` e `continue` em relação a laços de nível superior, escreve-se depois do elemento de estrutura, entre parênteses, um número inteiro maior ou igual a 1:

- **1** refere-se ao laço/switch atual.
- **2** refere-se ao laço/switch que envolve o laço/switch atual.
- ...

Note-se que laços e `switch` são contados da mesma forma. Se, dentro de um `switch`, se quiser abandonar/repetir um laço de nível superior, é preciso usar `break(2)` ou `continue(2)`.

> **Informação:** se depois de `break` ou `continue` não houver número entre parênteses, isso deve ser entendido como `break(1)` ou `continue(1)`, respectivamente.

> **Informação:** se um `continue` se refere a um `switch`, é disparado um erro.

### Exemplo

```
$
for(v[1] = 0; v[1] < 10; v[1]++)
  for(v[2] = 0; v[2] < 10; v[2]++)
    if(v[2] == 5)
      break(2);
    endif;
  next;
next;
```

No final, a primeira variável seria 0 e a segunda seria 5.

GUIA

# Outros comandos de salto

No DestinyScript, além dos comandos de salto para laços, existem outros dois comandos de salto:

| Comando | Descrição |
|---|---|
| `exit` | Encerra o script atual |
| `pause` | Pausa o script atual |

## Encerrando o script atual (exit)

Com o elemento de estrutura `exit` (do inglês "sair") o script em execução é encerrado. Todos os comandos que eventualmente estejam depois dele deixam de ser executados. Isso é útil, por exemplo, para verificar uma série de critérios de exclusão e, se necessário, não deixar o script continuar.

### Exemplo

```
$
// Obtém o carimbo de tempo atual em milissegundos
v[1] = Time.Tick;

// Executa um laço por 5 segundos
do
  v[2] = Time.Tick;
  if(Keyboard.GetKeyState(VK_ESCAPE) != 0)
    // Encerra o script atual se Esc for pressionada
    a[1] = "Cancelado";
    exit;
  endif;
loop while(v[2] - v[1] < 5000);

// Registra em a[1] o tempo de espera usado
a[1] = "Tempo esperado: " .. (v[2] - v[1]) / 1000;
```

Neste script é executado um laço que deve rodar pelo menos 5 segundos (= 5000 milissegundos) (linhas 6 a 13). Se, porém, a tecla Esc for pressionada (linha 8), o script atual é abandonado (linha 11). O código que está abaixo (essencialmente a linha 16) não é executado nesse caso.

## Pausando o script atual (pause)

Com o elemento de estrutura `pause` (do inglês "pausa") o script em execução é brevemente pausado. A utilidade técnica está em que, durante um ciclo de espera (como no exemplo acima), o jogo **não congela**. Normalmente um script roda até o fim sem que o `RPG_RT` possa voltar a trabalhar (por exemplo, processar eventos, exibir gráficos etc.). Com o uso do `pause`, isso passa a ser possível.

> **Informação:** o uso do `pause` é especialmente útil combinado com o objeto `Command`, pois esse objeto pode executar tempos de espera do EventScript. Com o uso de `pause` no script, esse tempo de espera é de fato cumprido.

### Exemplo

```
$
// Obtém o carimbo de tempo atual em milissegundos
v[1] = Time.Tick;

// Executa um laço por 5 segundos
do
  v[2] = Time.Tick;
  if(Keyboard.GetKeyState(VK_ESCAPE) != 0)
    // Encerra o script atual se Esc for pressionada
    a[1] = "Cancelado";
```

```
    exit;
  endif;
  pause;
loop while(v[2] - v[1] < 5000);

// Registra em a[1] o tempo de espera usado
a[1] = "Tempo esperado: " .. (v[2] - v[1]) / 1000;
```

Este exemplo equivale ao anterior, com a diferença de que o `pause` foi inserido na linha 13. Graças ao `pause`, o jogo deixa de parecer travado e outros eventos em execução (por exemplo, processos paralelos) continuam sendo executados.

PARTE 4

# Referência

Lista de objetos

Lista de grupos de constantes

Lista de erros

# Lista de objetos

## Descrição

Os objetos contêm a funcionalidade do DestinyScript. Eles servem para agrupar propriedades e métodos que pertencem juntos. Por exemplo, o objeto `Keyboard` é usado para acessar o teclado.

## Sintaxe

A sintaxe de uma chamada de objeto é sempre a mesma (ela já foi mencionada nos operadores de objeto):

```
$
// ATENÇÃO: este não é um script executável, pois…

// Chamada de um método com dois parâmetros
// O valor de retorno é guardado em v[1]
v[1] = Objeto.Metodo(Parametro1, Parametro2);

// Chamada de um método com um parâmetro
// O valor de retorno não é guardado (por exemplo…
Objeto.Metodo2(Parametro1);

// Consulta de uma propriedade de um objeto
// (o valor é guardado em v[2])
v[2] = Objeto.Propriedade;

// Consulta de uma propriedade de um objeto indexado…
// (o valor é guardado em v[3])
v[3] = Objeto[Indice].Propriedade;
```

> **Informação:** maiúsculas e minúsculas no nome de um objeto não importam: `OBJETO.PROPRIEDADE`, `objeto.propriedade` ou até `oBjEtO.PrOpRiEdAdE` produzem o mesmo efeito.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `a` | Objeto | O objeto a serve para dar acesso às strings (cadeias de caracteres, "ANSI-Strings") da Destiny.dll. |
| `Actor` | Objeto | O objeto Actor serve para reunir todas as propriedades dos heróis. |
| `Attribute` | Objeto | O objeto Attribute serve para tornar acessíveis todos os atributos de armaduras e técnicas do banco de dados do jogo. |
| `Client` | Objeto | O objeto Client serve para estabelecer conexões com outros computadores pela internet ou pela rede. |
| `Command` | Objeto | O objeto Command serve para tornar acessíveis os comandos de evento do RPG_RT. |
| `CommonEvent` | Objeto | O objeto CommonEvent serve para tornar acessíveis todos os CommonEvents do banco de dados do jogo. |
| `Condition` | Objeto | O objeto Condition serve para tornar acessíveis todas as condições dos heróis e inimigos do banco de dados do jogo. |
| `Convert` | Objeto | Com o objeto Convert é possível converter tipos de dado em outros tipos de dado. |
| `d` | Objeto | O objeto d serve para tornar acessíveis os Dwords (números inteiros, como variáveis) da Destiny.dll. |
| `Destiny` | Objeto | Com o objeto Destiny é possível acessar a própria Destiny.dll. |

| Nome | Tipo | Resumo |
|---|---|---|
| `Directory` | Objeto | Com o objeto Directory é possível editar estruturas de pastas e arquivos, além de buscar em pastas. |
| `Error` | Objeto | Com o objeto Error é possível (des)ativar mensagens de erro individuais. |
| `Errors` | Objeto | Com o objeto Errors é possível controlar o tratamento de erros. |
| `Event` | Objeto | O objeto Event serve para tornar acessíveis os eventos do mapa. |
| `f` | Objeto | O objeto f serve para tornar acessíveis os Floats (números com vírgula) da Destiny.dll. |
| `Faceset` | Objeto | Com o objeto Faceset é possível obter informações sobre o Faceset carregado para a exibição de mensagens, bem como alterar o AuroraSheet do face a ser exibido. |
| `File` | Objeto | Com o objeto File é possível ler/gravar arquivos. |
| `Game` | Objeto | Com o objeto Game é possível acessar o próprio jogo atual (praticamente o RPG_RT). |
| `Item` | Objeto | O objeto Item serve para reunir todas as propriedades de um item (objeto, equipamento etc.). |
| `Keyboard` | Objeto | Com o objeto Keyboard é possível consultar e definir teclas. |
| `Logic` | Objeto | Com o objeto Logic é possível realizar operações lógicas e comparações. |
| `Map` | Objeto | O objeto Map serve para acessar o mapa atual do jogo. |
| `MapEvent` | Objeto | O objeto MapEvent serve para tornar acessíveis os eventos do mapa. |
| `Math` | Objeto | Com o objeto Math é possível realizar diversas operações matemáticas (trigonometria, logaritmação, radiciação, arredondamento...). |
| `Mouse` | Objeto | O objeto Mouse serve para descobrir ou alterar as propriedades atuais do mouse (como, por exemplo, a posição do ponteiro do mouse). |
| `Panorama` | Objeto | O objeto Panorama serve para tornar acessíveis as propriedades do panorama (imagem de fundo) do mapa. |
| `Picture` | Objeto | O objeto Picture serve para dar acesso às Pictures do jogo. |
| `s` | Objeto | O objeto s serve para tornar acessíveis os switches (valores booleanos, "sim"/"não") do RPG_RT. |
| `Screen` | Objeto | O objeto Screen serve para disponibilizar a tela atual do jogo. |
| `Server` | Objeto | O objeto Server serve para aceitar conexões de entrada pela internet ou pela rede. |
| `Skill` | Objeto | O objeto Skill serve para reunir todas as propriedades de uma técnica. |
| `String` | Objeto | O objeto String serve para reunir todos os métodos relacionados a Strings. |
| `Time` | Objeto | O objeto Time serve para descobrir a hora/data atual do computador. |
| `v` | Objeto | O objeto v serve para tornar acessíveis as variáveis (números inteiros) do RPG_RT. |

OBJETO

# Objeto: a

## Sintaxe

```
a
```

## Descrição

O objeto `a` serve para dar acesso às strings (cadeias de caracteres, "ANSI-Strings") da `Destiny.dll`. As strings do objeto `a` são gerenciadas pela `Destiny.dll`.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Count` | Propriedade | Devolve a quantidade de strings no objeto a. |
| `ReAlloc()` | Método | Com o método ReAlloc é possível aumentar/diminuir a quantidade de strings no objeto a. |
| `a[]` | Propriedade | Permite o acesso a uma determinada string do objeto a. |

## Veja também

`d`, `f`, `v`, `s`

PROPRIEDADE

# Propriedade: a.Count

## Sintaxe

```
a.Count
```

## Descrição

Devolve a quantidade de strings no objeto `a`. Pode ser usada para fazer um laço sobre todas as strings.

> **Informação:** A quantidade de strings disponível ao iniciar o jogo é definida pelo DestinyPatcher.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** >= 0

## Exemplo

```
$
// Definir todas as strings como "Bolo"
for(v[1] = 1; v[1] <= a.Count; v[1]++)
  a[v[1]] = "Kuchen";  // Isso mesmo!
next;
```

## Veja também

`a`, `a[]`, `a.ReAlloc()`

MÉTODO

# Método: a.ReAlloc()

## Sintaxe

```
a.ReAlloc(Quantidade)
```

## Descrição

Com o método ReAlloc é possível aumentar/diminuir a quantidade de strings no objeto `a`.

> **Informação:** Se a quantidade de strings informada for grande demais, é lançado o erro `DESTINYERROR_MEMORY`. O conteúdo original do objeto `a` é mantido.

> **Informação:** Ao aumentar a quantidade de strings, as novas strings são definidas como "".

## Lista de parâmetros

### Parâmetro: `Quantidade`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 0
- **Descrição:** A nova quantidade de strings no objeto `a`. Esse valor precisa ser maior/igual a 1. Se o número for grande demais, a chamada deste método falha.

## Exemplo

```
$
// Aumentar em 1 a quantidade máxima de strings
a.ReAlloc(a.Count + 1);
// Atribuir à nova string o valor "Pizza" (não é à toa
a[a.Count] = "Pizza";
```

## Erros possíveis

`DESTINYERROR_VALUE`, `DESTINYERROR_MEMORY`

## Veja também

`a`, `a[]`, `a.Count`

PROPRIEDADE

# Propriedade: a[]

## Sintaxe

```
a[Índice]
```

## Descrição

Permite o acesso a uma determinada string do objeto `a`.

## Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** não

## Lista de parâmetros

### Parâmetro: `Índice`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... a.Count
- **Descrição:** O índice da string. Precisa estar na faixa de 1 ... `a.Count`.

## Exemplo

```
$
// Definir o valor da primeira string como "Suco"
a[1] = "Saft";
```

## Erros possíveis

`DESTINYERROR_ARRAYBOUND`

## Veja também

`a`, `a.Count`, `a.ReAlloc()`

OBJETO

# Objeto: Actor

## Sintaxe

```
Actor
```

## Descrição

O objeto `Actor` serve para reunir todas as propriedades dos heróis. Elas podem ser acessadas através do operador de array.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Count` | Propriedade | Com a propriedade Count é possível saber a quantidade de heróis nesse array. |
| `Actor[]` | Objeto | O objeto Actor[] serve para reunir todas as propriedades de um herói. |

PROPRIEDADE

# Propriedade: Actor.Count

## Sintaxe

```
Actor.Count
```

## Descrição

Com a propriedade Count é possível saber a quantidade de heróis nesse array.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

## Exemplo

No exemplo a seguir, partimos do princípio de que o jogo acabou de ser iniciado e que o banco de dados padrão do jogo está sendo usado.

```
$
// Descobrir a quantidade de heróis e guardar em v[1]
v[1] = Actor.Count;                    // 8
// Executar um laço sobre todos os heróis e concatenar seus nomes
a[1] = "";
for(v[2] = 1; v[2] < Actor.Count; v[2]++)
  a[1] ..= Actor[v[2]].Name;
next;
```

## Veja também

`Actor`, `Actor[]`, `Actor[].Name`

OBJETO

# Objeto: Actor[]

## Sintaxe

```
Actor[Herói-Id]
```

## Descrição

O objeto `Actor[]` serve para reunir todas as propriedades de um herói. Ele é identificado pela Herói-Id.

## Lista de parâmetros

### Parâmetro: `Herói-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... Actor.Count
- **Descrição:** A Id do herói

## Erros possíveis

`DESTINYERROR_ARRAYBOUND`

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Equipamento` | Categoria | Com as propriedades desta categoria é possível ler/gravar o equipamento atual de um herói. |
| `Aparência` | Categoria | As propriedades desta categoria afetam a aparência de um herói. |
| `Default` | Objeto | Este subobjeto reúne todos os valores padrão dos atributos de um herói, que podem ser alterados durante o andamento do jogo. |
| `LevelAttribute` | Objeto | Contém uma lista dos valores de status de todos os níveis de um herói. |
| `LevelAttribute[]` | Objeto | O objeto LevelAttribute reúne os valores de status padrão de um herói em um determinado nível. |
| `Nomes` | Categoria | Com as propriedades desta categoria é possível editar o nome e o título de um herói. |
| `Outras propriedades` | Categoria | Esta categoria contém todas as demais propriedades do herói, que podem ser configuradas pelo banco de dados do jogo. |
| `AttributeEffect` | Objeto | Contém todas as efetividades dos atributos para um herói. |
| `Condition` | Objeto | Contém uma lista de todas as condições de um herói. |
| `ConditionEffect` | Objeto | Contém todas as efetividades das condições para um herói. |
| `Skill` | Objeto | Contém uma lista de todas as técnicas de um herói. |

| Nome | Tipo | Resumo |
| --- | --- | --- |
| `Valores de status` | Categoria | Com as propriedades desta categoria é possível editar os valores de status atuais de um herói. |

CATEGORIA

## Categoria: Equipamento

Com as propriedades desta categoria é possível ler/gravar o equipamento atual de um herói. Os valores dessas propriedades correspondem sempre à Item-Id do banco de dados do jogo. Se uma propriedade estiver definida como 0, isso significa que não há nenhum item equipado naquela área.

### Itens contidos

| Nome | Tipo | Resumo |
| --- | --- | --- |
| `Accessory` | Propriedade | Com a propriedade Accessory é possível ler/gravar a Item-Id do acessório atualmente equipado. |
| `Armor` | Propriedade | Com a propriedade Armor é possível ler/gravar a Item-Id da armadura atualmente equipada. |
| `Helmet` | Propriedade | Com a propriedade Helmet é possível ler/gravar a Item-Id do elmo atualmente equipado. |
| `Shield` | Propriedade | Com a propriedade Shield é possível ler/gravar a Item-Id do escudo atualmente equipado. |
| `Weapon` | Propriedade | Com a propriedade Weapon é possível ler/gravar a Item-Id da arma atualmente equipada. |

PROPRIEDADE

## Propriedade: Actor[].Accessory

### Sintaxe

```
Actor[Herói-Id].Accessory
```

### Descrição

Com a propriedade Accessory é possível ler/gravar a Item-Id do acessório atualmente equipado. Se essa propriedade for igual a 0, isso significa que nenhum acessório está equipado. Esse valor precisa ser uma Item-Id válida, caso contrário um erro do Destiny é lançado.

### Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

### Exemplo

```
$
// Obter a Item-Id atual do acessório do primeiro herói
v[1] = Actor[1].Accessory;
// Se essa propriedade não for 0, obter a agilidade do item
if(v[1] != 0)
  v[2] = Item[v[1]].Agility;
endif;
```

```
// Remover o acessório do primeiro herói (o item em si não é afetado)
Actor[1].Accessory = 0;
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Actor[]`, `Item[].Protection`, `Item[].Agility`

PROPRIEDADE

# Propriedade: Actor[].Armor

## Sintaxe

```
Actor[Herói-Id].Armor
```

## Descrição

Com a propriedade Armor é possível ler/gravar a Item-Id da armadura atualmente equipada. Se essa propriedade for igual a 0, isso significa que nenhuma armadura está equipada. Esse valor precisa ser uma Item-Id válida, caso contrário um erro do Destiny é lançado.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

## Exemplo

```
$
// Obter a Item-Id atual da armadura do primeiro herói
v[1] = Actor[1].Armor;
// Se essa propriedade não for 0, obter a defesa do item
if(v[1] != 0)
  v[2] = Item[v[1]].Defense;
endif;
// Remover a armadura do primeiro herói (o item em si não é afetado)
Actor[1].Armor = 0;
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Actor[]`, `Item[].Protection`, `Item[].Defense`

PROPRIEDADE

# Propriedade: Actor[].Helmet

## Sintaxe

```
Actor[Herói-Id].Helmet
```

## Descrição

Com a propriedade Helmet é possível ler/gravar a Item-Id do elmo atualmente equipado. Se essa propriedade for igual a 0, isso significa que nenhum elmo está equipado. Esse valor precisa ser uma Item-Id válida, caso contrário um erro do Destiny é lançado.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

## Exemplo

```
$
// Obter a Item-Id atual do elmo do primeiro herói
v[1] = Actor[1].Helmet;
// Se essa propriedade não for 0, obter a inteligência do item
if(v[1] != 0)
  v[2] = Item[v[1]].Mind;
endif;
// Remover o elmo do primeiro herói (o item em si não é afetado)
Actor[1].Helmet = 0;
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Actor[]`, `Item[].Protection`, `Item[].Mind`

PROPRIEDADE

# Propriedade: Actor[].Shield

## Sintaxe

```
Actor[Herói-Id].Shield
```

## Descrição

Com a propriedade Shield é possível ler/gravar a Item-Id do escudo atualmente equipado. Se essa propriedade for igual a 0, isso significa que nenhum escudo está equipado. Esse valor precisa ser uma Item-Id válida, caso

contrário um erro do Destiny é lançado.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

## Exemplo

```
$
// Obter a Item-Id atual do escudo do primeiro herói
v[1] = Actor[1].Shield;
// Se essa propriedade não for 0, obter a defesa do item
if(v[1] != 0)
  v[2] = Item[v[1]].Defense;
endif;
// Remover o escudo do primeiro herói (o item em si não é afetado)
Actor[1].Shield = 0;
```

## Observações adicionais

Se o herói for um lutador de duas armas, aqui é equipada uma segunda arma.

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Actor[]`, `Item[].Protection`, `Item[].Defense`

PROPRIEDADE

# Propriedade: Actor[].Weapon

## Sintaxe

```
Actor[Herói-Id].Weapon
```

## Descrição

Com a propriedade Weapon é possível ler/gravar a Item-Id da arma atualmente equipada. Se essa propriedade for igual a 0, isso significa que nenhuma arma está equipada. Esse valor precisa ser uma Item-Id válida, caso contrário um erro do Destiny é lançado.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

## Exemplo

```
$
// Obter a Item-Id atual da arma do primeiro herói
v[1] = Actor[1].Weapon;
// Se essa propriedade não for 0, obter o ataque do item
if(v[1] != 0)
  v[2] = Item[v[1]].Attack;
endif;
// Desarmar o primeiro herói (o item em si não é afetado)
Actor[1].Weapon = 0;
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Actor[]`, `Item[].Weapon`, `Item[].Attack`

CATEGORIA

# Categoria: Aparência

As propriedades desta categoria afetam a aparência de um herói.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Charframe` | Propriedade | Com a propriedade Charframe é possível ler/gravar o índice do Frame usado. |
| `Charset` | Propriedade | Com a propriedade Charset é possível ler/gravar o nome do arquivo (sem caminho/extensão e relativo à pasta de Charsets). |
| `Faceframe` | Propriedade | Com a propriedade Faceframe é possível ler/gravar o índice do Frame usado. |
| `Faceset` | Propriedade | Com a propriedade Faceset é possível ler/gravar o nome do arquivo (sem caminho/extensão e relativo à pasta de Facesets). |
| `Transparency` | Propriedade | Com a propriedade Transparency é possível ler/gravar o valor de transparência do char. |

PROPRIEDADE

# Propriedade: Actor[].Charframe

## Sintaxe

```
Actor[Herói-Id].Charframe
```

## Descrição

Com a propriedade Charframe é possível ler/gravar o índice do `Frame` usado. Junto com a propriedade `Charset`, isso indica o char atualmente usado. Esse valor deve estar na faixa de 0 ... 7 (veja a figura).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não
- **Faixa de valores:** 0 ... 7

## Exemplo

```
$
// Obter o Charframe atual do primeiro herói em v[1]
v[1] = Actor[1].Charframe;
// Definir o Charframe atual do primeiro herói como 0
Actor[1].Charframe = 0;
// Definir o Charframe atual do primeiro herói como o padrão
Actor[1].Charframe = Actor[1].Default.Charframe;
```

## Erros possíveis

`DESTINYERROR_CONVERT`

## Veja também

`Actor[].Charset`, `Actor[].Default.Charframe`

PROPRIEDADE

# Propriedade: Actor[].Charset

## Sintaxe

```
Actor[Herói-Id].Charset
```

## Descrição

Com a propriedade Charset é possível ler/gravar o nome do arquivo (sem caminho/extensão e relativo à pasta de Charsets). Junto com a propriedade `Charframe`, isso indica o char atualmente usado.

## Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** não

## Exemplo

```
$
// Obter o nome do arquivo Charset atual do primeiro herói
a[1] = Actor[1].Charset;
// Definir o nome do arquivo Charset atual do primeiro herói como "CHARA1"
Actor[1].Charaset = "CHARA1";
// Definir o nome do arquivo Charset atual do primeiro herói como o padrão
Actor[1].Charset = Actor[1].Default.Charset;
```

## Erros possíveis

`DESTINYERROR_CONVERT`

## Veja também

`Actor[].Charframe`, `Actor[].Default.Charset`

PROPRIEDADE

# Propriedade: Actor[].Faceframe

## Sintaxe

```
Actor[Herói-Id].Faceframe
```

## Descrição

Com a propriedade Faceframe é possível ler/gravar o índice do `Frame` usado. Junto com a propriedade `Faceset`, isso indica o face atualmente usado. Esse valor deve estar na faixa de 0 ... 15 (veja a figura).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não
- **Faixa de valores:** 0 ... 15

## Exemplo

```
$
// Obter o Faceframe atual do primeiro herói em v[1]
v[1] = Actor[1].Faceframe;
// Definir o Faceframe atual do primeiro herói como 0
Actor[1].Faceframe = 0;
// Definir o Faceframe atual do primeiro herói como o padrão
Actor[1].Faceframe = Actor[1].Default.Faceframe;
// Preparar o rosto do primeiro herói para a próxima exibição
Faceset.Filename = Actor[1].Faceset;
Faceset.Frame = Actor[1].Faceframe;
```

## Erros possíveis

`DESTINYERROR_CONVERT`

## Veja também

`Actor[].Faceset`, `Actor[].Default.Faceframe`, `Faceset.Filename`, `Faceset.Frame`

PROPRIEDADE

## Propriedade: Actor[].Faceset

### Sintaxe

```
Actor[Herói-Id].Faceset
```

### Descrição

Com a propriedade Faceset é possível ler/gravar o nome do arquivo (sem caminho/extensão e relativo à pasta de Facesets). Junto com a propriedade `Faceframe`, isso indica o face atualmente usado.

### Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** não

### Exemplo

```
$
// Obter o nome do arquivo Faceset atual do primeiro herói
a[1] = Actor[1].Faceset;
// Definir o nome do arquivo Faceset atual do primeiro herói como "CHARA1"
Actor[1].Faceset = "CHARA1";
// Definir o nome do arquivo Faceset atual do primeiro herói como o padrão
Actor[1].Faceset = Actor[1].Default.Faceset;
// Preparar o rosto do primeiro herói para a próxima exibição
Faceset.Filename = Actor[1].Faceset;
Faceset.Frame = Actor[1].Faceframe;
```

### Erros possíveis

`DESTINYERROR_CONVERT`

### Veja também

`Actor[].Faceframe`, `Actor[].Default.Faceset`, `Faceset.Filename`, `Faceset.Frame`

PROPRIEDADE

## Propriedade: Actor[].Transparency

### Sintaxe

```
Actor[Herói-Id].Transparency
```

### Descrição

Com a propriedade Transparency é possível ler/gravar o valor de transparência do char. Esse valor deve estar na faixa de 0 ... 8, sendo que 0 significa não transparente e 4 significa meio-transparente. Se, nas propriedades do

herói, a caixa Transparente estiver marcada, esse valor é 3. Com valores maiores que 8 ocorrem erros de cálculo divertidos (ou seja, o herói fica com uma aparência bem estranha; veja a figura).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não
- **Faixa de valores:** 0 ... 8

## Exemplo

```
$
// Obter a transparência atual do primeiro herói
v[1] = Actor[1].Transparency;
// Aumentar em 1 a transparência do primeiro herói
if(Actor[1].Transparency < 8)
  Actor[1].Transparency++;
// Definir a transparência do primeiro herói como o padrão
Actor[1].Transparency = Actor[1].Default.Transpar
```

## Erros possíveis

`DESTINYERROR_CONVERT`

## Veja também

`Actor[].Default.Transparency`

OBJETO

# Objeto: Actor[].Default

## Sintaxe

```
Actor[Herói-Id].Default
```

## Descrição

Este subobjeto reúne todos os valores padrão dos atributos de um herói, que podem ser alterados durante o andamento do jogo.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Accessory` | Propriedade | Com a propriedade Accessory é possível ler a Item-Id do acessório padrão de um herói. |
| `Armor` | Propriedade | Com a propriedade Armor é possível ler a Item-Id da armadura padrão de um herói. |
| `Charframe` | Propriedade | Com a propriedade Charframe é possível ler o índice padrão do Frame usado no char de um herói. |
| `Charset` | Propriedade | Com a propriedade Charset é possível ler o nome de arquivo padrão (sem caminho/extensão e relativo à pasta de Charsets) do Charset de um herói. |

| Nome | Tipo | Resumo |
|---|---|---|
| `Degree` | Propriedade | Com a propriedade Degree é possível ler o título padrão de um herói. |
| `Faceframe` | Propriedade | Com a propriedade Faceframe é possível ler o índice padrão do Frame usado no face de um herói. |
| `Faceset` | Propriedade | Com a propriedade Faceset é possível ler o nome de arquivo padrão (sem caminho/extensão e relativo à pasta de Facesets) do Faceset de um herói. |
| `Helmet` | Propriedade | Com a propriedade Helmet é possível ler a Item-Id do elmo padrão de um herói. |
| `Level` | Propriedade | Com a propriedade Level é possível ler o nível padrão de um herói. |
| `Name` | Propriedade | Com a propriedade Name é possível ler o nome padrão de um herói. |
| `Shield` | Propriedade | Com a propriedade Shield é possível ler a Item-Id do escudo padrão de um herói. |
| `Transparency` | Propriedade | Com a propriedade Transparency é possível ler a transparência padrão de um herói. |
| `Weapon` | Propriedade | Com a propriedade Weapon é possível ler a Item-Id da arma padrão de um herói. |

PROPRIEDADE

# Propriedade: Actor[].Default.Accessory

## Sintaxe

```
Actor[Herói-Id].Default.Accessory
```

## Descrição

Com a propriedade Accessory é possível ler a Item-Id do acessório padrão de um herói. Se um herói não tiver acessório por padrão, esse valor é igual a 0. O acessório de um herói pode mudar durante o jogo. Para acessar a Item-Id do acessório atual de um herói, use a propriedade `Accessory` do objeto `Actor[]`.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Actor[].Default.Armor

## Sintaxe

```
Actor[Herói-Id].Default.Armor
```

## Descrição

Com a propriedade Armor é possível ler a Item-Id da armadura padrão de um herói. Se um herói não tiver armadura por padrão, esse valor é igual a 0. A armadura de um herói pode mudar durante o jogo. Para acessar a Item-Id da armadura atual de um herói, use a propriedade `Armor` do objeto `Actor[]`.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Actor[].Default.Charframe

## Sintaxe

```
Actor[Herói-Id].Default.Charframe
```

## Descrição

Com a propriedade Charframe é possível ler o índice padrão do `Frame` usado no char de um herói. Junto com a propriedade `Charset`, isso forma o char padrão de um herói. O char de um herói pode mudar durante o jogo. Para acessar o Charframe atual, use a propriedade `Charframe` do objeto `Actor[]`. O valor dessa propriedade fica na faixa de 0 ... 7 (veja a figura).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Actor[].Default.Charset

## Sintaxe

```
Actor[Herói-Id].Default.Charset
```

## Descrição

Com a propriedade Charset é possível ler o nome de arquivo padrão (sem caminho/extensão e relativo à pasta de Charsets) do Charset de um herói. Junto com a propriedade `Charframe`, isso forma o char padrão de um herói. O char de um herói pode mudar durante o jogo. Para acessar o Charset atual, use a propriedade `Charset` do objeto `Actor[]`.

## Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Actor[].Default.Degree

### Sintaxe

```
Actor[Herói-Id].Default.Degree
```

### Descrição

Com a propriedade Degree é possível ler o título padrão de um herói. O título de um herói pode mudar durante o jogo. Para acessar o título atual, use a propriedade `Degree` do objeto `Actor[]`.

### Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Actor[].Default.Faceframe

### Sintaxe

```
Actor[Herói-Id].Default.Faceframe
```

### Descrição

Com a propriedade Faceframe é possível ler o índice padrão do `Frame` usado no face de um herói. Junto com a propriedade `Faceset`, isso forma o face padrão de um herói. O face de um herói pode mudar durante o jogo. Para acessar o Faceframe atual, use a propriedade `Faceframe` do objeto `Actor[]`. O valor dessa propriedade fica na faixa de 0 ... 15 (veja a figura).

### Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Actor[].Default.Faceset

### Sintaxe

```
Actor[Herói-Id].Default.Faceset
```

## Descrição

Com a propriedade Faceset é possível ler o nome de arquivo padrão (sem caminho/extensão e relativo à pasta de Facesets) do Faceset de um herói. Junto com a propriedade `Faceframe`, isso forma o face padrão de um herói. O face de um herói pode mudar durante o jogo. Para acessar o Faceset atual, use a propriedade `Faceset` do objeto `Actor[]`.

## Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Actor[].Default.Helmet

## Sintaxe

```
Actor[Herói-Id].Default.Helmet
```

## Descrição

Com a propriedade Helmet é possível ler a Item-Id do elmo padrão de um herói. Se um herói não tiver elmo por padrão, esse valor é igual a 0. O elmo de um herói pode mudar durante o jogo. Para acessar a Item-Id do elmo atual de um herói, use a propriedade `Helmet` do objeto `Actor[]`.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Actor[].Default.Level

## Sintaxe

```
Actor[Herói-Id].Default.Level
```

## Descrição

Com a propriedade Level é possível ler o nível padrão de um herói. Isso corresponde ao nível que o herói tem quando entra no grupo pela primeira vez. O nível de um herói pode mudar durante o jogo. Para acessar o nível atual de um herói, use a propriedade `Level` do objeto `Actor[]`.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Actor[].Default.Name

### Sintaxe

```
Actor[Herói-Id].Default.Name
```

### Descrição

Com a propriedade Name é possível ler o nome padrão de um herói. O nome de um herói pode mudar durante o jogo. Para acessar o nome atual, use a propriedade `Name` do objeto `Actor[]`.

### Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Actor[].Default.Shield

### Sintaxe

```
Actor[Herói-Id].Default.Shield
```

### Descrição

Com a propriedade Shield é possível ler a Item-Id do escudo padrão de um herói. Se um herói não tiver escudo por padrão, esse valor é igual a 0. O escudo de um herói pode mudar durante o jogo. Para acessar a Item-Id do escudo atual de um herói, use a propriedade `Shield` do objeto `Actor[]`.

### Observações adicionais

Se o herói for um lutador de duas armas, aqui é equipada uma segunda arma.

### Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Actor[].Default.Transparency

### Sintaxe

```
Actor[Herói-Id].Default.Transparency
```

## Descrição

Com a propriedade Transparency é possível ler a transparência padrão de um herói. A transparência de um herói pode mudar durante o jogo. Para acessar a transparência atual, use a propriedade `Transparency` do objeto `Actor[]`.

## Observações adicionais

Nas configurações padrão do banco de dados do jogo, só é possível marcar uma caixa para a transparência. Se a caixa estiver marcada, o valor dessa propriedade é 3; caso contrário, 0.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Actor[].Default.Weapon

## Sintaxe

```
Actor[Herói-Id].Default.Weapon
```

## Descrição

Com a propriedade Weapon é possível ler a Item-Id da arma padrão de um herói. Se um herói não tiver arma por padrão, esse valor é igual a 0. A arma de um herói pode mudar durante o jogo. Para acessar a Item-Id da arma atual de um herói, use a propriedade `Weapon` do objeto `Actor[]`.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

OBJETO

# Objeto: Actor[].LevelAttribute

## Sintaxe

```
Actor[Herói-Id].LevelAttribute
```

## Descrição

Contém uma lista dos valores de status de todos os níveis de um herói.

## Itens contidos

| Nome | Tipo | Resumo |
| --- | --- | --- |
| `Count` | Propriedade | Devolve o nível máximo de um herói. |

PROPRIEDADE

# Propriedade: Actor[].LevelAttribute.Count

## Sintaxe

```
Actor[Herói-Id].LevelAttribute.Count
```

## Descrição

Devolve o nível máximo de um herói.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

OBJETO

# Objeto: Actor[].LevelAttribute[]

## Sintaxe

```
Actor[Herói-Id].LevelAttribute[Level]
```

## Descrição

O objeto LevelAttribute reúne os valores de status padrão de um herói em um determinado nível.

## Lista de parâmetros

### Parâmetro: `Level`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... Actor[].LevelAttribute.Count
- **Descrição:** Indica o nível do herói. Precisa estar na faixa de 1 ... `Actor[].LevelAttribute.Count`.

## Erros possíveis

`DESTINYERROR_ARRAYBOUND`

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Agility` | Propriedade | A propriedade Agility indica os pontos de agilidade de um herói em um determinado nível. |
| `Attack` | Propriedade | A propriedade Attack indica os pontos de ataque de um herói em um determinado nível. |
| `Defense` | Propriedade | A propriedade Defense indica os pontos de defesa de um herói em um determinado nível. |
| `LearnableSkill` | Objeto | Contém a lista das técnicas que o herói pode aprender no respectivo nível. |
| `LearnableSkill[]` | Propriedade | A propriedade LearnableSkill indica as Skill-Ids das técnicas que um herói pode aprender em um determinado nível. |
| `MaxHP` | Propriedade | A propriedade MaxHP indica os pontos de vida máximos (HP = Health Points) de um herói em um determinado nível. |
| `MaxMP` | Propriedade | A propriedade MaxMP indica os pontos de magia máximos (MP = Magic Points) de um herói em um determinado nível. |
| `Mind` | Propriedade | A propriedade Mind indica os pontos de inteligência de um herói em um determinado nível. |

PROPRIEDADE

# Propriedade: Actor[].LevelAttribute[].Agility

## Sintaxe

```
Actor[Herói-Id].LevelAttribute[Level].Agility
```

## Descrição

A propriedade Agility indica os pontos de agilidade de um herói em um determinado nível. Os pontos de agilidade reais podem, porém, variar desse valor, já que os valores de status ainda podem ser alterados por outros fatores (como condições, equipamento etc.). Para obter os pontos de agilidade reais atuais, use o método `ChangeVariable` do objeto `Command`.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Actor[].LevelAttribute[].Attack

## Sintaxe

```
Actor[Herói-Id].LevelAttribute[Level].Attack
```

## Descrição

A propriedade Attack indica os pontos de ataque de um herói em um determinado nível. Os pontos de ataque reais podem, porém, variar desse valor, já que os valores de status ainda podem ser alterados por outros fatores (como

condições, equipamento etc.). Para obter os pontos de ataque reais atuais, use o método `ChangeVariable` do objeto `Command`.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Actor[].LevelAttribute[].Defense

## Sintaxe

```
Actor[Herói-Id].LevelAttribute[Level].Defense
```

## Descrição

A propriedade Defense indica os pontos de defesa de um herói em um determinado nível. Os pontos de defesa reais podem, porém, variar desse valor, já que os valores de status ainda podem ser alterados por outros fatores (como condições, equipamento etc.). Para obter os pontos de defesa reais atuais, use o método `ChangeVariable` do objeto `Command`.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

OBJETO

# Objeto: Actor[].LevelAttribute[].LearnableSkill

## Sintaxe

```
Actor[Herói-Id].LevelAttribute[Level].LearnableSkill
```

## Descrição

Contém a lista das técnicas que o herói pode aprender no respectivo nível.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Count` | Propriedade | A propriedade Count indica a quantidade de técnicas que um herói pode aprender em um determinado nível. |

PROPRIEDADE

## Propriedade: Actor[].LevelAttribute[].LearnableSkill.Count

### Sintaxe

```
Actor[Herói-Id].LevelAttribute[Level].LearnableSkill.Count
```

### Descrição

A propriedade Count indica a quantidade de técnicas que um herói pode aprender em um determinado nível. Essa propriedade é usada junto com a propriedade `LearnableSkill[]` para descobrir as técnicas que um herói aprende a partir de um determinado nível.

### Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

### Exemplo

```
$
// Montar uma string informando quantas técnicas o herói
// aprendeu ao atingir o nível atual
a[1] = Actor[1].Name .. " hat im aktuellen Level
```

PROPRIEDADE

## Propriedade: Actor[].LevelAttribute[].LearnableSkill[]

### Sintaxe

```
Actor[Actor-
Id].LevelAttribute[Level].LearnableSkill[LearnableSkill-
Index]
```

### Descrição

A propriedade LearnableSkill indica as Skill-Ids das técnicas que um herói pode aprender em um determinado nível.

### Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

### Lista de parâmetros

#### Parâmetro: `Habilidade-aprendível-Índice`

- **Tipo de dado:** `dword`

- **Opcional:** não
- **Faixa de valores:** 0 bis (Actor[].LevelAttribute[].LearnableSkill.Count - 1)
- **Descrição:** Indica o índice da técnica a aprender. Precisa estar na faixa de 0 até ( `Actor[].LevelAttribute[].LearnableSkill.Count` - 1).

## Exemplo

```
$
// Montar uma lista de todas as técnicas que o herói pode aprender no próximo nível
if(Actor[1].Level < Actor[1].MaxLevel)
  a[1] = Actor[1].Name .. " kann im nächsten Leve
  if(Actor[1].LevelAttribute[Actor[1].Level].Lear
    a[1] ..= "- keine -" .. CRLF;
  else
    for(v[1] = 0; v[1] < Actor[1].LevelAttribute[
      a[1] ..= Skill[Actor[1].LevelAttribute[Acto
    next;
  endif;
else
  a[1] = Actor[1].Name .. " kann schon Alles!";
endif;
```

## Erros possíveis

`DESTINYERROR_ARRAYBOUND`

PROPRIEDADE

# Propriedade: Actor[].LevelAttribute[].MaxHP

## Sintaxe

```
Actor[Herói-Id].LevelAttribute[Level].MaxHP
```

## Descrição

A propriedade MaxHP indica os pontos de vida máximos (HP = Health Points) de um herói em um determinado nível. Os pontos de vida máximos reais podem, porém, variar desse valor, já que os valores de status ainda podem ser alterados por outros fatores (como condições, equipamento etc.). Para obter os pontos de vida máximos reais atuais, use o método `ChangeVariable` do objeto `Command`.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Actor[].LevelAttribute[].MaxMP

### Sintaxe

```
Actor[Herói-Id].LevelAttribute[Level].MaxMP
```

### Descrição

A propriedade MaxMP indica os pontos de magia máximos (MP = Magic Points) de um herói em um determinado nível. Os pontos de magia máximos reais podem, porém, variar desse valor, já que os valores de status ainda podem ser alterados por outros fatores (como condições, equipamento etc.). Para obter os pontos de magia máximos reais atuais, use o método `ChangeVariable` do objeto `Command`.

### Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Actor[].LevelAttribute[].Mind

### Sintaxe

```
Actor[Herói-Id].LevelAttribute[Level].Mind
```

### Descrição

A propriedade Mind indica os pontos de inteligência de um herói em um determinado nível. Os pontos de inteligência reais podem, porém, variar desse valor, já que os valores de status ainda podem ser alterados por outros fatores (como condições, equipamento etc.). Para obter os pontos de inteligência reais atuais, use o método `ChangeVariable` do objeto `Command`.

### Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

CATEGORIA

## Categoria: Nomes

Com as propriedades desta categoria é possível editar o nome e o título de um herói.

### Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Degree` | Propriedade | Com a propriedade Degree é possível ler/gravar o título de um herói. |

| Nome | Tipo | Resumo |
|---|---|---|
| `Name` | Propriedade | Com a propriedade Name é possível ler/gravar o nome de um herói. |

PROPRIEDADE

# Propriedade: Actor[].Degree

## Sintaxe

```
Actor[Herói-Id].Degree
```

## Descrição

Com a propriedade Degree é possível ler/gravar o título de um herói. Se essa propriedade for definida como `String.Chr(1)`, o título padrão passa a ser usado por referência. Para usar o título padrão sem referência, pode-se usar a propriedade `Default.Degree`.

## Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** não

## Exemplo

No exemplo a seguir, partimos do princípio de que o jogo acabou de ser iniciado e que o banco de dados padrão do jogo está sendo usado.

```
$
// Ler o título do primeiro herói e guardar em a[1]
a[1] = Actor[1].Degree;                    // "…"
// Definir o título do primeiro herói como "Iniciante"
Actor[1].Name = "Anfänger";
// Definir o título do primeiro herói como o título padrão, por referência
Actor[1].Degree = String.Chr(1);
// Definir o título padrão do primeiro herói como "Especialista"
Actor[1].Default.Degree = "Experte";
// Ler o título do primeiro herói e guardar em a[2]
a[2] = Actor[1].Degree;                    // "…"
// Definir o título do primeiro herói como o título padrão atual
Actor[1].Degree = Actor[1].Default.Degree;   // "…"
// Restaurar o título padrão ao seu valor anterior
Actor[1].Default.Degree = a[1];            // "…"
// Ler o título do primeiro herói e guardar em a[3]
a[3] = Actor[1].Degree;                    // "…"
```

## Erros possíveis

`DESTINYERROR_CONVERT`

## Veja também

`Actor[]`, `Actor[].Default.Degree`, `Actor[].Name`, `String.Chr()`

PROPRIEDADE

# Propriedade: Actor[].Name

## Sintaxe

```
Actor[Herói-Id].Name
```

## Descrição

Com a propriedade Name é possível ler/gravar o nome de um herói. Para atribuir ao herói o seu nome padrão, pode-se usar o conteúdo da propriedade `Default.Name`.

## Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** não

## Exemplo

No exemplo a seguir, partimos do princípio de que o jogo acabou de ser iniciado e que o banco de dados padrão do jogo está sendo usado.

```
$
// Ler o nome do primeiro herói e guardar em a[1]
a[1] = Actor[1].Name;                    // "Alex"
// Definir o nome do primeiro herói como "Bernd"
Actor[1].Name = "Bernd";                 // "Bernd"
// Definir o nome do primeiro herói como o nome padrão
Actor[1].Name = Actor[1].Default.Name;   // "Alex"
// Ler o nome do primeiro herói e guardar em a[2]
a[2] = Actor[1].Name;                    // "Alex"
```

## Observações adicionais

Se essa propriedade for definida como `String.Chr(1)`, o nome padrão também passa a ser usado. Essa, porém, é uma particularidade do RPG Maker.

## Erros possíveis

`DESTINYERROR_CONVERT`

## Veja também

`Actor[]`, `Actor[].Default.Name`, `Actor[].Degree`, `String.Chr()`

CATEGORIA

# Categoria: Outras propriedades

Esta categoria contém todas as demais propriedades do herói, que podem ser configuradas pelo banco de dados do jogo.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `AttributeEffect[]` | Propriedade | Com a propriedade AttributeEffect é possível ler o quanto um atributo de arma/magia afeta o herói. |
| `Condition[]` | Propriedade | Com a propriedade Condition é possível ler a Condition-Id de uma condição específica de um herói. |
| `ConditionEffect[]` | Propriedade | Com a propriedade ConditionEffect é possível ler o quanto uma condição afeta o herói. |
| `ControlledByAI` | Propriedade | Com a propriedade ControlledByAI é possível ler se o herói é controlado pelo computador no sistema de batalha padrão (AI = Artificial Intelligence, isto é, Inteligência Artificial). |
| `CriticalHitChance` | Propriedade | Com a propriedade CriticalHitChance é possível ler a chance de acerto crítico. |
| `FixedEquipment` | Propriedade | Com a propriedade FixedEquipment é possível ler se o equipamento do herói não pode ser alterado pelo menu padrão (True = o equipamento não pode ser alterado, False = o equipamento pode ser alterado). |
| `HasCondition()` | Método | Com o método HasCondition é possível verificar se um herói atualmente tem uma determinada condição. |
| `HasSkill()` | Método | Com o método HasSkill é possível verificar se um herói tem uma determinada técnica. |
| `MaxLevel` | Propriedade | Com a propriedade MaxLevel é possível ler o nível máximo do herói. |
| `Skill[]` | Propriedade | Com a propriedade Skill é possível ler a Skill-Id de uma técnica específica de um herói. |
| `StrongDefense` | Propriedade | Com a propriedade StrongDefense é possível ler se o herói tem uma defesa especialmente forte. |
| `TwoSwordStyle` | Propriedade | Com a propriedade TwoSwordStyle é possível ler se o herói é equipado com duas armas (True) ou com uma arma e um escudo (False). |
| `UnarmedBattleAnimation` | Propriedade | Com a propriedade UnarmedBattleAnimation é possível ler a BattleAnimation-Id exibida no sistema de batalha padrão quando o herói ataca sem estar portando uma arma. |
| `UseSkillName` | Propriedade | Com a propriedade UseSkillName é possível ler o texto "usar técnica" do herói. |

PROPRIEDADE

# Propriedade: Actor[].AttributeEffect[]

## Sintaxe

```
Actor[Herói-Id].AttributeEffect[Atributo-Id]
```

## Descrição

Com a propriedade AttributeEffect é possível ler o quanto um atributo de arma/magia afeta o herói. Isso é representado no banco de dados do jogo pelas letras de A a E, sendo que A corresponde ao número 0 (o atributo tem efeito acima da média) e E corresponde ao número 4 (o atributo tem efeito muito fraco ou nenhum). Para identificação, podem ser usadas as constantes `Effect`.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

## Lista de parâmetros

### Parâmetro: `Atributo-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** Indica a Attribute-Id do atributo de arma/magia cujo valor deve ser obtido. Se esse valor não for uma Attribute-Id válida, um erro do Destiny é lançado.

## Exemplo

```
$
// Verificar se o primeiro atributo de ataque (normalmente
if(Math.Rand(1, 100) <= Attribute[1].Rate[Actor[1
  // O atributo teve efeito; retirar do primeiro herói
  Actor[1].HP -= 10;
  // Verificar se o primeiro herói ficou sem pontos de vida
  if(Actor[1].HP <= 0)
    // Se o primeiro herói estiver morto, encerrar o jogo
    Command.GameOver();
  endif;
endif;
```

## Erros possíveis

`DESTINYERROR_ARRAYBOUND`

## Veja também

`Math.Rand()`, `Attribute[].Rate[]`, `Actor[].HP`, `Command.GameOver()`

PROPRIEDADE

# Propriedade: Actor[].Condition[]

## Sintaxe

```
Actor[Herói-Id].Condition[Condição-Índice]
```

## Descrição

Com a propriedade Condition é possível ler a Condition-Id de uma condição específica de um herói. Para verificar se o herói tem uma determinada condição, recomenda-se usar o método `HasCondition`. Para adicionar uma condição, recomenda-se usar o método `AddCondition` do objeto `Command`.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

## Lista de parâmetros

### Parâmetro: `Condição-Índice`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... (Actor[].Condition.Count - 1)
- **Descrição:** Indica o índice da condição. Precisa estar na faixa de 0 a (`Actor[].Condition.Count` - 1). Esse valor não corresponde à Condition-Id.

## Exemplo

```
$
// Montar uma lista de todas as condições do primeiro herói
a[1] = "";
for(v[1] = 0; v[1] < Actor[1].Condition.Count; v[
  a[1] ..= Condition[Actor[1].Condition[v[1]]].Na
next;
```

## Erros possíveis

`DESTINYERROR_ARRAYBOUND`

## Veja também

`Actor[].Condition.Count`, `Condition[].Name`, `Actor[].HasCondition()`

PROPRIEDADE

# Propriedade: Actor[].ConditionEffect[]

## Sintaxe

```
Actor[Herói-Id].ConditionEffect[Condição-Id]
```

## Descrição

Com a propriedade ConditionEffect é possível ler o quanto uma condição afeta o herói. Isso é representado no banco de dados do jogo pelas letras de A a E, sendo que A corresponde ao número 0 (a condição tem efeito acima da média) e E corresponde ao número 4 (a condição tem efeito muito fraco ou nenhum). Para identificação, podem ser usadas as constantes `Effect`.

## Propriedades

- **Tipo de dado:** `dword`

- **Somente leitura:** sim

## Lista de parâmetros

### Parâmetro: `Condição-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** Indica a Condition-Id da condição cujo valor deve ser obtido. Se esse valor não for uma Condition-Id válida, um erro do Destiny é lançado.

## Exemplo

No exemplo a seguir, são ignoradas peças de equipamento vestidas que poderiam impedir a aquisição de determinadas condições.

```
$
// Verificar se o primeiro herói já tem a condição 2 (nor
if(!Actor[1].HasCondition(2))
  // Se necessário, aplicar a condição 2 no primeiro herói (
  if(Math.Rand(1, 100) <= Condition[2].Chance[Act
    Command.AddCondition(1, 2);  // Herói com Herói-Id
  endif;
endif;
```

## Erros possíveis

`DESTINYERROR_ARRAYBOUND`

## Veja também

`Actor[].HasCondition()`, `Math.Rand()`, `Condition[].Chance[]`, `Command.AddCondition()`, `Actor[].Condition.Count`, `Actor[].Condition[]`

PROPRIEDADE

# Propriedade: Actor[].ControlledByAI

## Sintaxe

```
Actor[Herói-Id].ControlledByAI
```

## Descrição

Com a propriedade ControlledByAI é possível ler se o herói é controlado pelo computador no sistema de batalha padrão (AI = Artificial Intelligence, isto é, Inteligência Artificial). (True = controlado por IA, False = controlado pelo jogador).

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

## Exemplo

```
$
// Verificar se o primeiro herói é controlado pela IA durante o combate
if(Actor[1].ControlledByAI)
  a[1] = Actor[1].Name .. " ist ein willenloser S
else
  a[1] = Actor[1].Name .. " ist Mitglied der libe
endif;
```

## Veja também

`Actor[].Name`

PROPRIEDADE

# Propriedade: Actor[].CriticalHitChance

## Sintaxe

```
Actor[Herói-Id].CriticalHitChance
```

## Descrição

Com a propriedade CriticalHitChance é possível ler a chance de acerto crítico. Esse valor pode ficar entre 0 e 100.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** 0 ... 100

## Exemplo

```
$
// Obter a CriticalHitChance do primeiro herói
v[1] = Actor[1].CriticalHitChance;
// Gerar um número aleatório entre 1 e 100 e
v[2] = Math.Rand(1, 100);
// Em a[1], guardar uma mensagem de acerto crítico ou uma mensagem normal
if(v[2] <= v[1])
  a[1] = "Dir gelingt ein besonders kritischer Tr
else
  a[1] = "Dir gelingt ein gewöhnlicher Treffer.";
endif;
```

## Veja também

`Actor[]`, `Math.Rand()`

PROPRIEDADE

# Propriedade: Actor[].FixedEquipment

## Sintaxe

```
Actor[Herói-Id].FixedEquipment
```

## Descrição

Com a propriedade FixedEquipment é possível ler se o equipamento do herói não pode ser alterado pelo menu padrão (True = o equipamento não pode ser alterado, False = o equipamento pode ser alterado).

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

## Exemplo

```
$
// Verificar se o primeiro herói pode trocar seu equipamento
if(Actor[1].FixedEquipment)
  a[1] = Actor[1].Name .. " trägt einen ganzkörpe
else
  a[1] = Actor[1].Name .. " besitzt die besten Ko
endif;
```

## Veja também

`Actor[].Name`

MÉTODO

# Método: Actor[].HasCondition()

## Sintaxe

```
Actor[Herói-Id].HasCondition(Condição-Id)
```

## Descrição

Com o método HasCondition é possível verificar se um herói atualmente tem uma determinada condição.

## Lista de parâmetros

### Parâmetro: `Condição-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não

- **Descrição:** Indica a Condition-Id da condição a verificar. Se esse valor não for uma Condition-Id válida, um erro do Destiny é lançado.

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** Devolve True (se o herói tiver a condição) ou False (se ele não tiver a condição).

## Exemplo

```
$
// Verificar se o primeiro herói tem a segunda condição
if(Actor[1].HasCondition(2))
  a[1] = Actor[1].Name .. " ist verseucht mit """
else
  a[1] = Actor[1].Name .. " hat zum Glück kein ""
endif;
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Actor[].Name`, `Condition[].Name`, `Actor[].Condition.Count`, `Actor[].Condition[]`

MÉTODO
# Método: Actor[].HasSkill()

## Sintaxe

```
Actor[Herói-Id].HasSkill(Habilidade-Id)
```

## Descrição

Com o método HasSkill é possível verificar se um herói tem uma determinada técnica.

## Lista de parâmetros

### Parâmetro: `Habilidade-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** Indica a Skill-Id da técnica a verificar. Se esse valor não for uma Skill-Id válida, um erro do Destiny é lançado.

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** Devolve True (se o herói tiver a técnica) ou False (se ele não tiver a técnica).

## Exemplo

```
$
// Verificar se o primeiro herói tem a primeira técnica
if(Actor[1].HasSkill(1))
  a[1] = Actor[1].Name .. " verfügt über die """
else
  a[1] = Actor[1].Name .. " verfügt nicht über di
endif;
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Actor[].Name`, `Skill[].Name`, `Actor[].Skill.Count`, `Actor[].Skill[]`

PROPRIEDADE

# Propriedade: Actor[].MaxLevel

## Sintaxe

```
Actor[Herói-Id].MaxLevel
```

## Descrição

Com a propriedade MaxLevel é possível ler o nível máximo do herói. Esse valor pode ficar na faixa de 1...50 (RPG Maker 2000) ou 1...100 (RPG Maker 2003), dependendo da versão do Maker.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** 1 ... 50 (RPG-Maker 2000) oder 1 ... 100 (RPG-Maker 2003)

## Exemplo

```
$
// Os níveis que o primeiro herói ainda precisa alcançar
v[1] = Actor[1].MaxLevel - Actor[1].Level;
// A quantidade de níveis que o primeiro herói já obteve
v[2] = Actor[1].Level - Actor[1].Default.Level;
// Os níveis restantes e os já conquistados
a[1] = "Du bist bereits " .. v[2] .. " Level gest
```

## Veja também

`Actor[].Level`, `Actor[].Default.Level`

PROPRIEDADE

# Propriedade: Actor[].Skill[]

## Sintaxe

```
Actor[Herói-Id].Skill[Habilidade-Índice]
```

## Descrição

Com a propriedade Skill é possível ler a Skill-Id de uma técnica específica de um herói. Para verificar se o herói tem uma determinada técnica, recomenda-se usar o método `HasSkill`. Para adicionar uma técnica, recomenda-se usar o método `AddSkill` do objeto `Command`.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

## Lista de parâmetros

### Parâmetro: `Habilidade-Índice`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... (Actor[].Skill.Count - 1)
- **Descrição:** Indica o índice da técnica. Esse valor não corresponde à Skill-Id e precisa estar na faixa de 0 a (Actor[].Skill.Count - 1).

## Exemplo

```
$
// Montar uma lista de todas as técnicas do primeiro herói
a[1] = "";
for(v[1] = 0; v[1] < Actor[1].Skill.Count; v[1]++
  a[1] ..= Skill[Actor[1].Skill[v[1]]].Name .. CR
next;
```

## Erros possíveis

`DESTINYERROR_ARRAYBOUND`

## Veja também

`Actor[].Skill.Count`, `Skill[].Name`, `Actor[].HasSkill()`

PROPRIEDADE

# Propriedade: Actor[].StrongDefense

## Sintaxe

```
Actor[Herói-Id].StrongDefense
```

## Descrição

Com a propriedade StrongDefense é possível ler se o herói tem uma defesa especialmente forte. Isso significa que os pontos de defesa do herói são dobrados (True = pontos de defesa dobrados, False = pontos de defesa normais).

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

## Exemplo

```
$
// Verificar se o primeiro herói tem uma defesa especialmente forte
if(Actor[1].StrongDefense)
  a[1] = Actor[1].Name .. " hat sowohl 'ne harte
else
  a[1] = Actor[1].Name .. " ist die reinste Mimos
endif;
```

## Veja também

`Actor[].Name`

PROPRIEDADE

# Propriedade: Actor[].TwoSwordStyle

## Sintaxe

```
Actor[Herói-Id].TwoSwordStyle
```

## Descrição

Com a propriedade TwoSwordStyle é possível ler se o herói é equipado com duas armas (True) ou com uma arma e um escudo (False).

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

## Exemplo

```
$
// Verificar se o primeiro herói luta com duas armas
if(Actor[1].TwoSwordStyle)
  a[1] = Actor[1].Name .. " kämpft mit zwei Waffe
else
  a[1] = Actor[1].Name .. " kämpft ganz klassisch
endif;
```

## Veja também

`Actor[].Name`

PROPRIEDADE

# Propriedade: Actor[].UnarmedBattleAnimation

## Sintaxe

```
Actor[Herói-Id].UnarmedBattleAnimation
```

## Descrição

Com a propriedade UnarmedBattleAnimation é possível ler a BattleAnimation-Id exibida no sistema de batalha padrão quando o herói ataca sem estar portando uma arma.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

## Exemplo

O exemplo a seguir mostra a BattleAnimation do ataque do primeiro herói contra o evento atual.

```
$
// Verificar se o primeiro herói está equipado com uma arma
if(Actor[1].Weapon != 0)
  // O primeiro herói está equipado com uma arma
  // -> exibir a animação do ataque com a respectiva arma
  Command.ShowBattleAnimation(Item[Actor[1].Weapo
else
  // O primeiro herói NÃO está equipado com uma arma
  // -> exibir a animação do ataque desarmado
  Command.ShowBattleAnimation(Actor[1].UnarmedBat
endif;
```

## Veja também

`Command.ShowBattleAnimation()`, `Actor[].Weapon`, `Item[].Weapon.BattleAnimation`, `THIS`

PROPRIEDADE

# Propriedade: Actor[].UseSkillName

## Sintaxe

```
Actor[Herói-Id].UseSkillName
```

## Descrição

Com a propriedade UseSkillName é possível ler o texto "usar técnica" do herói. Se nenhum texto especial for definido para o herói, é devolvido o texto padrão do vocabulário do jogo.

## Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** sim

## Exemplo

```
$
// Obter o texto "usar técnica" do primeiro herói
a[1] = Actor[1].UseSkillName;
```

OBJETO

# Objeto: Actor[].AttributeEffect

## Sintaxe

```
Actor[Herói-Id].AttributeEffect
```

## Descrição

Contém todas as efetividades dos atributos para um herói.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Count` | Propriedade | Devolve a quantidade de atributos (do jogo). |

PROPRIEDADE

# Propriedade: Actor[].AttributeEffect.Count

## Sintaxe

```
Actor[Herói-Id].AttributeEffect.Count
```

## Descrição

Devolve a quantidade de atributos (do jogo). Isso deve ser usado apenas junto com `Actor[].AttributeEffect[]`. Para obter a quantidade normal de atributos do jogo, recomenda-se usar a propriedade `Count` do objeto `Attribute` .

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

## Veja também

`Actor[].AttributeEffect[]`, `Attribute`, `Attribute.Count`

OBJETO

# Objeto: Actor[].Condition

## Sintaxe

```
Actor[Herói-Id].Condition
```

## Descrição

Contém uma lista de todas as condições de um herói.

## Itens contidos

| Nome | Tipo | Resumo |
| --- | --- | --- |
| `Count` | Propriedade | Com a propriedade Count é possível ler a quantidade de condições de um herói. |

PROPRIEDADE

# Propriedade: Actor[].Condition.Count

## Sintaxe

```
Actor[Herói-Id].Condition.Count
```

## Descrição

Com a propriedade Count é possível ler a quantidade de condições de um herói.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

## Exemplo

```
$
// Montar uma lista de todas as condições do primeiro herói
a[1] = "";
for(v[1] = 0; v[1] < Actor[1].Condition.Count; v[
  a[1] ..= Condition[Actor[1].Condition[v[1]]].Na
next;
```

## Veja também

`Actor[].Condition[]`, `Condition[].Name`, `Actor[].HasCondition()`

OBJETO

# Objeto: Actor[].ConditionEffect

## Sintaxe

```
Actor[Herói-Id].ConditionEffect
```

## Descrição

Contém todas as efetividades das condições para um herói.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Count` | Propriedade | Devolve a quantidade de condições (do jogo). |

PROPRIEDADE

# Propriedade: Actor[].ConditionEffect.Count

## Sintaxe

```
Actor[Herói-Id].ConditionEffect.Count
```

## Descrição

Devolve a quantidade de condições (do jogo). Isso deve ser usado apenas junto com `Actor[].ConditionEffect[]`. Para obter a quantidade normal de condições do jogo, recomenda-se usar a propriedade `Count` do objeto `Condition`.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

## Veja também

`Actor[].ConditionEffect[]`, `Condition`, `Condition.Count`

OBJETO

# Objeto: Actor[].Skill

## Sintaxe

```
Actor[Herói-Id].Skill
```

## Descrição

Contém uma lista de todas as técnicas de um herói.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Count` | Propriedade | Com a propriedade Count é possível ler a quantidade de técnicas de um herói. |

PROPRIEDADE

# Propriedade: Actor[].Skill.Count

## Sintaxe

```
Actor[Herói-Id].Skill.Count
```

## Descrição

Com a propriedade Count é possível ler a quantidade de técnicas de um herói. Isso é usado junto com a propriedade `Skill[]`. Para adicionar uma técnica, recomenda-se usar o método `AddSkill` do objeto `Command`.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

## Exemplo

```
$
// Montar uma lista de todas as técnicas do primeiro herói
a[1] = "";
for(v[1] = 0; v[1] < Actor[1].Skill.Count; v[1]++
  a[1] ..= Skill[Actor[1].Skill[v[1]]].Name .. CR
next;
```

## Veja também

`Actor[].Skill[]`, `Skill[].Name`, `Actor[].HasSkill()`

CATEGORIA

# Categoria: Valores de status

Com as propriedades desta categoria é possível editar os valores de status atuais de um herói. Para obter os diversos valores de status atuais (por exemplo, pontos de ataque etc.), recomenda-se usar o método `ChangeVariable` do objeto `Command`. O motivo é que os valores de status atuais dependem de:

Os valores de status do nível atual do herói (veja `Actor[].LevelAttribute`); as diferenças dos valores de status em relação ao nível atual (veja as propriedades desta categoria); o equipamento atual (veja a categoria Equipamento e o objeto `Item`); as condições atuais do herói (veja `Actor[].Condition` e o objeto `Condition`); e, no caso dos pontos de defesa, a propriedade `StrongDefense` deste objeto.

Usar o método ChangeVariable garante que o algoritmo do RPG Maker seja usado (e, com isso, que todas as propriedades que alteram os valores de status entrem no cálculo).

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `AgilityDiff` | Propriedade | Com a propriedade AgilityDiff é possível ler/gravar a diferença dos pontos de agilidade de um herói (em relação aos pontos de agilidade normais do seu nível atual). |
| `AttackDiff` | Propriedade | Com a propriedade AttackDiff é possível ler/gravar a diferença dos pontos de ataque de um herói (em relação aos pontos de ataque normais do seu nível atual). |
| `DefenseDiff` | Propriedade | Com a propriedade DefenseDiff é possível ler/gravar a diferença dos pontos de defesa de um herói (em relação aos pontos de defesa normais do seu nível atual). |
| `EXP` | Propriedade | Com a propriedade EXP é possível ler/gravar os pontos de experiência (EXP = Experience) atuais de um herói. |
| `HP` | Propriedade | Com a propriedade HP (HP = Health Points) é possível ler/gravar os pontos de vida de um herói. |
| `Level` | Propriedade | Com a propriedade Level é possível ler/gravar o nível de um herói. |
| `MaxHPDiff` | Propriedade | Com a propriedade MaxHPDiff é possível ler/gravar a diferença dos pontos de vida máximos (HP = Health Points) de um herói (em relação aos pontos de vida máximos normais do seu nível atual). |
| `MaxMPDiff` | Propriedade | Com a propriedade MaxMPDiff é possível ler/gravar a diferença dos pontos de magia máximos (MP = Magic Points) de um herói (em relação aos pontos de magia máximos normais do seu nível atual). |
| `MindDiff` | Propriedade | Com a propriedade MindDiff é possível ler/gravar a diferença dos pontos de inteligência de um herói (em relação aos pontos de inteligência normais do seu nível atual). |
| `MP` | Propriedade | Com a propriedade MP (MP = Magic Points) é possível ler/gravar os pontos de magia de um herói. |

PROPRIEDADE

# Propriedade: Actor[].AgilityDiff

## Sintaxe

```
Actor[Herói-Id].AgilityDiff
```

## Descrição

Com a propriedade AgilityDiff é possível ler/gravar a diferença dos pontos de agilidade de um herói (em relação aos pontos de agilidade normais do seu nível atual). Esse valor não corresponde aos pontos de agilidade atuais. Para obter os pontos de agilidade atuais, pode-se usar o método `ChangeVariable` do objeto `Command` (veja o exemplo).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

## Exemplo

```
$
// Obter a diferença dos pontos de agilidade do primeiro herói
v[1] = Actor[1].AgilityDiff;
// Aumentar em 1 os pontos de agilidade do primeiro herói
Actor[1].AgilityDiff++;
// Obter os pontos de agilidade do primeiro herói
Command.ChangeVariable(2, OPERATION_SET, VALUETYP
// Obter os pontos de agilidade "normais" do primeiro herói
v[3] = Actor[1].LevelAttribute[Actor[1].Level].Ag
```

A diferença entre os pontos de agilidade (em v[2]) e os pontos de agilidade "normais" (em v[3]) é que, nos pontos de agilidade "normais", as alterações causadas pelo equipamento ou pelas condições não são consideradas.

## Veja também

`Actor[]`, `Command.ChangeVariable()`, `Actor[].LevelAttribute[].Agility`, `Actor[].Level`

PROPRIEDADE

# Propriedade: Actor[].AttackDiff

## Sintaxe

```
Actor[Herói-Id].AttackDiff
```

## Descrição

Com a propriedade AttackDiff é possível ler/gravar a diferença dos pontos de ataque de um herói (em relação aos pontos de ataque normais do seu nível atual). Esse valor não corresponde aos pontos de ataque atuais. Para obter os pontos de ataque atuais, pode-se usar o método `ChangeVariable` do objeto `Command` (veja o exemplo).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

## Exemplo

```
$
// Obter a diferença dos pontos de ataque do primeiro herói
v[1] = Actor[1].AttackDiff;
// Aumentar em 1 os pontos de ataque do primeiro herói
Actor[1].AttackDiff++;
// Obter os pontos de ataque do primeiro herói
Command.ChangeVariable(2, OPERATION_SET, VALUETYP
// Obter os pontos de ataque "normais" do primeiro herói
v[3] = Actor[1].LevelAttribute[Actor[1].Level].At
```

A diferença entre os pontos de ataque (em v[2]) e os pontos de ataque "normais" (em v[3]) é que, nos pontos de ataque "normais", as alterações causadas pelo equipamento ou pelas condições não são consideradas.

## Veja também

`Actor[]`, `Command.ChangeVariable()`, `Actor[].LevelAttribute[].Attack`, `Actor[].Level`

PROPRIEDADE

# Propriedade: Actor[].DefenseDiff

## Sintaxe

```
Actor[Herói-Id].DefenseDiff
```

## Descrição

Com a propriedade DefenseDiff é possível ler/gravar a diferença dos pontos de defesa de um herói (em relação aos pontos de defesa normais do seu nível atual). Esse valor não corresponde aos pontos de defesa atuais. Para obter os pontos de defesa atuais, pode-se usar o método `ChangeVariable` do objeto `Command` (veja o exemplo).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

## Exemplo

```
$
// Obter a diferença dos pontos de defesa do primeiro herói
v[1] = Actor[1].DefenseDiff;
// Aumentar em 1 os pontos de defesa do primeiro herói
Actor[1].DefenseDiff++;
// Obter os pontos de defesa do primeiro herói
Command.ChangeVariable(2, OPERATION_SET, VALUETYP
// Obter os pontos de defesa "normais" do primeiro herói
v[3] = Actor[1].LevelAttribute[Actor[1].Level].De
```

A diferença entre os pontos de defesa (em v[2]) e os pontos de defesa "normais" (em v[3]) é que, nos pontos de defesa "normais", as alterações causadas pelo equipamento, pelas condições ou pela propriedade StrongDefense não são consideradas.

## Veja também

`Actor[]`, `Command.ChangeVariable()`, `Actor[].LevelAttribute[].Defense`, `Actor[].Level`

PROPRIEDADE

# Propriedade: Actor[].EXP

## Sintaxe

```
Actor[Herói-Id].EXP
```

## Descrição

Com a propriedade EXP é possível ler/gravar os pontos de experiência (EXP = Experience) atuais de um herói. Alterar esse valor não ajusta o nível automaticamente. Isso só acontece na próxima vez em que os pontos de experiência forem ajustados por uma função do `RPG_RT` (por exemplo, depois de um combate vencido). Para ajustar o nível diretamente ao alterar os pontos de experiência, recomenda-se usar o método `AddExp` do objeto `Command`.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

## Exemplo

```
$
// Obter os pontos de experiência atuais do primeiro herói
v[1] = Actor[1].EXP;
// Aumentar em 100 os pontos de experiência atuais do primeiro herói
Actor[1].EXP += 100;
// Aumentar em 100 os pontos de experiência do primeiro herói (ajustando o nível)
Command.AddExp(1, 100, true);
```

## Veja também

`Actor[]`, `Command.AddExp()`, `Command.SubtractExp()`

PROPRIEDADE

# Propriedade: Actor[].HP

## Sintaxe

```
Actor[Herói-Id].HP
```

## Descrição

Com a propriedade HP (HP = Health Points) é possível ler/gravar os pontos de vida de um herói. Se esse valor for definido como 0, o jogador não morre automaticamente. Em vez disso, o jogo trava no próximo combate, quando o jogador definir o comando de ataque para esse herói.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

## Exemplo

```
$
// Ler os pontos de vida do primeiro herói e guardar em v[1]
v[1] = Actor[1].HP;
// Obter os pontos de vida máximos do primeiro herói
Command.ChangeVariable(2, OPERATION_SET, VALUETYP
// Definir os pontos de vida do primeiro herói como o valor obtido
Actor[1].HP = v[2]
```

## Veja também

`Actor[]`, `Command.ChangeVariable()`, `Actor[].LevelAttribute[].MaxHP`, `Actor[].MaxHPDiff`

PROPRIEDADE

# Propriedade: Actor[].Level

## Sintaxe

```
Actor[Herói-Id].Level
```

## Descrição

Com a propriedade Level é possível ler/gravar o nível de um herói. Mas cuidado: alterar o nível não ajusta a quantidade de pontos de experiência. Além disso, um valor grande demais (ou pequeno demais) pode fazer o `RPG_RT` travar. Resumindo: o valor mínimo é 1 e o valor máximo é a quantidade de níveis definida no banco de dados do jogo.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não
- **Faixa de valores:** 1 ... Actor[].MaxLevel

## Exemplo

```
$
// Ler o nível do primeiro herói e guardar em v[1]
v[1] = Actor[1].Level;
// Aumentar em 1 o nível do primeiro herói (se ainda
if(Actor[1].Level < Actor[1].MaxLevel)
  Actor[1].Level++;
endif;
```

## Veja também

`Actor[]`, `Actor[].MaxLevel`, `Actor[].EXP`

PROPRIEDADE

# Propriedade: Actor[].MaxHPDiff

## Sintaxe

```
Actor[Herói-Id].MaxHPDiff
```

## Descrição

Com a propriedade MaxHPDiff é possível ler/gravar a diferença dos pontos de vida máximos (HP = Health Points) de um herói (em relação aos pontos de vida máximos normais do seu nível atual). Esse valor não corresponde aos pontos de vida máximos atuais. Para obter os pontos de vida máximos atuais, pode-se usar o método `ChangeVariable` do objeto `Command` (veja o exemplo).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

## Exemplo

```
$
// Obter a diferença dos pontos de vida máximos do primeiro herói
v[1] = Actor[1].MaxHPDiff;
// Aumentar em 1 os pontos de vida máximos do primeiro herói
Actor[1].MaxHPDiff++;
// Obter os pontos de vida máximos do primeiro herói
Command.ChangeVariable(2, OPERATION_SET, VALUETYP
// Obter os pontos de vida máximos "normais" do primeiro herói
v[3] = Actor[1].LevelAttribute[Actor[1].Level].Ma
```

A diferença entre os pontos de vida máximos (em v[2]) e os pontos de vida máximos "normais" (em v[3]) é que, nos pontos de vida máximos "normais", as alterações causadas pelo equipamento ou pelas condições não são consideradas.

## Veja também

`Actor[]`, `Command.ChangeVariable()`, `Actor[].LevelAttribute[].MaxHP`, `Actor[].Level`

PROPRIEDADE

# Propriedade: Actor[].MaxMPDiff

## Sintaxe

```
Actor[Herói-Id].MaxMPDiff
```

## Descrição

Com a propriedade MaxMPDiff é possível ler/gravar a diferença dos pontos de magia máximos (MP = Magic Points) de um herói (em relação aos pontos de magia máximos normais do seu nível atual). Esse valor não corresponde aos pontos de magia máximos atuais. Para obter os pontos de magia máximos atuais, pode-se usar o método `ChangeVariable` do objeto `Command` (veja o exemplo).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

## Exemplo

```
$
// Obter a diferença dos pontos de magia máximos do primeiro herói
v[1] = Actor[1].MaxMPDiff;
// Aumentar em 1 os pontos de magia máximos do primeiro herói
Actor[1].MaxMPDiff++;
// Obter os pontos de magia máximos do primeiro herói
Command.ChangeVariable(2, OPERATION_SET, VALUETYP
// Obter os pontos de magia máximos "normais" do primeiro herói
v[3] = Actor[1].MaxMP[Actor[1].Level] + Actor[1].
```

A diferença entre os pontos de magia máximos (em v[2]) e os pontos de magia máximos "normais" (em v[3]) é que, nos pontos de magia máximos "normais", as alterações causadas pelo equipamento ou pelas condições não são consideradas.

## Veja também

`Actor[]`, `Command.ChangeVariable()`, `Actor[].LevelAttribute[].MaxMP`, `Actor[].Level`

PROPRIEDADE

# Propriedade: Actor[].MindDiff

## Sintaxe

```
Actor[Herói-Id].MindDiff
```

## Descrição

Com a propriedade MindDiff é possível ler/gravar a diferença dos pontos de inteligência de um herói (em relação aos pontos de inteligência normais do seu nível atual). Esse valor não corresponde aos pontos de inteligência atuais. Para obter os pontos de inteligência atuais, pode-se usar o método `ChangeVariable` do objeto `Command` (veja o exemplo).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

## Exemplo

```
$
// Obter a diferença dos pontos de inteligência do primeiro herói
v[1] = Actor[1].MindDiff;
// Aumentar em 1 os pontos de inteligência do primeiro herói
Actor[1].MindDiff++;
// Obter os pontos de inteligência do primeiro herói
Command.ChangeVariable(2, OPERATION_SET, VALUETYP
// Obter os pontos de inteligência "normais" do primeiro herói
v[3] = Actor[1].LevelAttribute[Actor[1].Level].Mi
```

A diferença entre os pontos de inteligência (em v[2]) e os pontos de inteligência "normais" (em v[3]) é que, nos pontos de inteligência "normais", as alterações causadas pelo equipamento ou pelas condições não são consideradas.

## Veja também

`Actor[]`, `Command.ChangeVariable()`, `Actor[].LevelAttribute[].Mind`, `Actor[].Level`

PROPRIEDADE

# Propriedade: Actor[].MP

## Sintaxe

```
Actor[Herói-Id].MP
```

## Descrição

Com a propriedade MP (MP = Magic Points) é possível ler/gravar os pontos de magia de um herói.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

## Exemplo

```
$
// Ler os pontos de magia do primeiro herói e guardar em v[1]
v[1] = Actor[1].MP;
// Obter os pontos de vida máximos do primeiro herói
Command.ChangeVariable(2, OPERATION_SET, VALUETYP
// Definir os pontos de magia do primeiro herói como o valor obtido
Actor[1].MP = v[2]
```

## Veja também

`Actor[]`, `Command.ChangeVariable()`, `Actor[].LevelAttribute[].MaxMP`, `Actor[].MaxMPDiff`

OBJETO

# Objeto: Attribute

## Sintaxe

```
Attribute
```

## Descrição

O objeto Attribute serve para tornar acessíveis todos os atributos de armaduras e técnicas do banco de dados do jogo.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Count` | Propriedade | Devolve a quantidade de atributos. |
| `Attribute[]` | Objeto | Permite o acesso a um determinado elemento do objeto Attribute. |

PROPRIEDADE

# Propriedade: Attribute.Count

## Sintaxe

```
Attribute.Count
```

## Descrição

Devolve a quantidade de atributos. Isso pode ser usado para fazer um laço sobre todos os atributos.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

## Exemplo

```
$
// Montar uma lista de todos os atributos e
a[1] = "";
for(v[1] = 1; v[1] <= Attribute.Count; v[1]++)
  a[1] ..= Attribute[v[1]].Name .. CRLF;
next;
```

OBJETO

# Objeto: Attribute[]

## Sintaxe

```
Attribute[Atributo-Id]
```

## Descrição

Permite o acesso a um determinado elemento do objeto `Attribute`.

## Lista de parâmetros

### Parâmetro: `Atributo-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... Attribute.Count
- **Descrição:** A Attribute-Id do respectivo atributo. Esse valor precisa estar na faixa de 1 ... `Attribute.Count`.

## Erros possíveis

`DESTINYERROR_ARRAYBOUND`

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Name` | Propriedade | Com a propriedade Name é possível descobrir o nome de um atributo. |
| `Rate` | Objeto | Com o objeto Rate é possível descobrir as taxas de efetividade de um atributo. |
| `Rate[]` | Propriedade | Com a propriedade Rate é possível descobrir a efetividade em porcentagem (0 ... |
| `Type` | Propriedade | Com a propriedade Type é possível ler o tipo de um atributo. |

PROPRIEDADE

# Propriedade: Attribute[].Name

## Sintaxe

```
Attribute[Atributo-Id].Name
```

## Descrição

Com a propriedade Name é possível descobrir o nome de um atributo.

## Propriedades

- **Tipo de dado:** `string`

- **Somente leitura:** sim

## Exemplo

No exemplo a seguir, partimos do princípio de que o jogo acabou de ser iniciado e que o banco de dados padrão do jogo está sendo usado.

```
$
// Ler o nome do primeiro atributo e guardar em
a[1] = Attribute[1].Name;                // "Sword"
// Listar todos os atributos da primeira arma e
// Isso resultaria, ao final, em "Shock" .. CRLF
a[2] = "Attribute der Waffe """ .. Item[17].Name
for(v[1] = 1; v[1] <= Attribute.Count; v[1]++)
  if(Item[17].HasAttribute(v[1]))
    a[2] ..= Attribute[v[1]].Name .. CRLF;
  endif;
next;
```

## Veja também

`Attribute`, `Attribute.Count`, `Item[].Name`, `Item[].HasAttribute()`

OBJETO

# Objeto: Attribute[].Rate

## Sintaxe

```
Attribute[Atributo-Id].Rate
```

## Descrição

Com o objeto Rate é possível descobrir as taxas de efetividade de um atributo.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Count` | Propriedade | Devolve sempre 5, já que o RPG Maker prevê exatamente 5 taxas de efetividade. |

PROPRIEDADE

# Propriedade: Attribute[].Rate.Count

## Sintaxe

```
Attribute[Atributo-Id].Rate.Count
```

## Descrição

Devolve sempre 5, já que o RPG Maker prevê exatamente 5 taxas de efetividade.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Attribute[].Rate[]

## Sintaxe

```
Attribute[Atributo-Id].Rate[Nível]
```

## Descrição

Com a propriedade Rate é possível descobrir a efetividade em porcentagem (0 ... 100) em um dos cinco níveis (A ... E).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** 0 ... 100

## Lista de parâmetros

### Parâmetro: `Nível`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** Indica o respectivo nível (A = 0, B = 1, C = 2, D = 3, E = 4). Para isso podem ser usadas as constantes `Effect`.

## Erros possíveis

`DESTINYERROR_ARRAYBOUND`

## Veja também

`Actor[].AttributeEffect[]`

PROPRIEDADE

# Propriedade: Attribute[].Type

## Sintaxe

```
Attribute[Atributo-Id].Type
```

## Descrição

Com a propriedade Type é possível ler o tipo de um atributo. Para identificar esse valor, podem ser usadas as constantes `AttributeType`.

`ATTRIBUTETYPE_WEAPON`: o atributo é um atributo de arma (físico) `ATTRIBUTETYPE_MAGIC`: o atributo é um atributo de magia (mágico)

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

OBJETO

# Objeto: Client

## Sintaxe

```
Client
```

## Descrição

O objeto Client serve para estabelecer conexões com outros computadores pela internet ou pela rede. Isso pode acontecer tanto estabelecendo explicitamente uma conexão com outro computador através de `Client[].Connect()`, quanto aceitando uma conexão de entrada através de `Server.Accept()`. Os dados são sempre transmitidos por `TCP/IP`, o que traz as seguintes características:

Trabalha-se de forma orientada a conexão. Isso significa que é preciso estabelecer uma conexão pela rede/internet. Se a conexão for interrompida, não é mais possível enviar dados. Se o endereço de destino não puder ser alcançado, um erro é reportado. Os dados são recebidos na mesma ordem em que são enviados. Como a ordem dos dados recebidos coincide com a dos dados enviados, não é preciso se preocupar que dados relacionados cheguem fora de ordem. Os dados são verificados quanto à validade e, se necessário, requisitados novamente. Como o TCP/IP tem verificação de validade, os dados normalmente chegam ao destinatário sem erros e sem perda de dados.

A desvantagem do TCP/IP é que ele é bastante lento para grandes volumes de dados voláteis (por exemplo, posições de tela).

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `CloseAll()` | Método | Com o método CloseAll é possível fechar todos os sockets cliente abertos no objeto Client. |
| `Count` | Propriedade | Devolve a quantidade de sockets cliente no objeto Client. |
| `NextFree()` | Método | Com o método NextFree é possível descobrir o índice do primeiro socket cliente livre no objeto Client. |
| `ReAlloc()` | Método | Com o método ReAlloc é possível aumentar/diminuir a quantidade de sockets cliente no objeto Client. |
| `Client[]` | Objeto | O objeto Client[] serve para reunir todas as propriedades e métodos de um socket cliente. |

## Veja também

`Server`

MÉTODO

# Método: Client.CloseAll()

## Sintaxe

```
Client.CloseAll()
```

## Descrição

Com o método CloseAll é possível fechar todos os sockets cliente abertos no objeto `Client`.

## Exemplo

```
$
// Conectar três sockets ao Google
Client[0].Connect("www.google.de", 80, SOCK_RAW);
Client[1].Connect("www.google.de", 80, SOCK_RAW);
Client[2].Connect("www.google.de", 80, SOCK_RAW);

// Fechar todos os sockets abertos
Client.CloseAll();

// Consultar o status dos três primeiros sockets e
a[1] = Client[0].State .. ", " .. Client[1].State
```

## Veja também

`Client`, `Client[]`, `Client.Count`, `Client.ReAlloc()`, `Client.NextFree()`, `Client[].Connect()`, `Client[].State`, `Client[].Close()`

PROPRIEDADE

# Propriedade: Client.Count

## Sintaxe

```
Client.Count
```

## Descrição

Devolve a quantidade de sockets cliente no objeto `Client`. Isso pode ser usado para fazer um laço sobre todos os clientes.

> **Informação:** A quantidade de clientes disponível ao iniciar o jogo é sempre 32. Essa quantidade pode ser alterada com `Client.ReAlloc()`.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** >= 0

## Exemplo

```
$
// Fechar todos os sockets
for(v[1] = 0; v[1] < Client.Count; v[1]++)
  Client[v[1]].Close();
next;
```

## Veja também

`Client`, `Client[]`, `Client.ReAlloc()`, `Client.CloseAll()`, `Client.NextFree()`, `Client[].Close()`

MÉTODO

# Método: Client.NextFree()

## Sintaxe

```
Client.NextFree()
```

## Descrição

Com o método NextFree é possível descobrir o índice do primeiro socket cliente livre no objeto `Client`. Se não existir nenhum socket cliente livre, a quantidade de sockets cliente é automaticamente aumentada em 1, de modo que um novo socket cliente fica disponível.

## Exemplo

```
$
// Remover todos os sockets
Client.ReAlloc(0);

// Descobrir o índice do próximo socket livre
v[1] = Client.NextFree();  // 0 (a quantidade foi aumen
// Conectar o socket ao Google
Client[v[1]].Connect("www.google.de", 80, SOCK_RA
// Descobrir o índice do próximo socket livre
v[2] = Client.NextFree();  // 1 (a quantidade foi aumen
// Conectar o socket ao Google
Client[v[2]].Connect("www.google.de", 80, SOCK_RA

// Fechar novamente o primeiro socket
Client[v[1]].Close();
// Descobrir o índice do próximo socket livre
v[3] = Client.NextFree();  // 0 (a quantidade de client
                           //    já que há um socket livr

// Fechar todos os sockets ainda abertos
Client.CloseAll();
```

## Observações adicionais

A funcionalidade do método NextFree pode ser entendida através do seguinte código DestinyScript:

```
$
// Descobrir o próximo socket livre e guardar em v[1]
for(v[1] = 0; v[1] < Client.Count; v[1]++)
  if(Client[v[1]].State == STATE_CLOSED)
    break;
  endif;
next;
if(v[1] == Client.Count)
  Client.ReAlloc(v[1] + 1);
endif;

// v[1] é o índice do próximo socket cliente
// ...
```

## Erros possíveis

`DESTINYERROR_MEMORY`

## Veja também

`Client`, `Client[]`, `Client.Count`, `Client.ReAlloc()`, `Client.CloseAll()`, `Client[].Connect()`, `Client[].Close()`

MÉTODO

# Método: Client.ReAlloc()

## Sintaxe

```
Client.ReAlloc(Quantidade)
```

## Descrição

Com o método ReAlloc é possível aumentar/diminuir a quantidade de sockets cliente no objeto `Client`.

> **Informação:** Se a quantidade de Dwords informada for grande demais, é lançado o erro `DESTINYERROR_MEMORY`. O conteúdo original do objeto `Client` é mantido.

> **Informação:** Ao aumentar a quantidade de sockets cliente, os novos sockets cliente ficam fechados.

> **Informação:** Ao diminuir a quantidade de sockets cliente, os sockets cliente eliminados pela redução são fechados automaticamente.

## Lista de parâmetros

### Parâmetro: `Quantidade`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 0
- **Descrição:** A nova quantidade de sockets cliente no objeto `Client`. Esse valor precisa ser maior/igual a 0. Com um número grande demais, a chamada deste método falha.

## Exemplo

```
$
// Aumentar em 1 a quantidade máxima de sockets clien
Client.ReAlloc(Client.Count + 1);
// Conectar com o novo socket cliente a
Client[Client.Count].Connect("www.google.de", 80,
// Enviar "Olá, seu trambolho de dados!" ao Google
Client[Client.Count].SendRawData("Hallo, du olle
// Fechar a conexão novamente
Client[Client.Count].Close();
```

## Erros possíveis

`DESTINYERROR_VALUE`, `DESTINYERROR_MEMORY`

## Veja também

`Client`, `Client[]`, `Client.Count`, `Client.CloseAll()`, `Client.NextFree()`, `Client[].Connect()`, `Client[].SendRawData()`, `Client[].Close()`

OBJETO

# Objeto: Client[]

## Sintaxe

```
Client[Cliente-Índice]
```

## Descrição

O objeto Client[] serve para reunir todas as propriedades e métodos de um socket cliente. Ele é identificado pelo índice de cliente. Esse índice de cliente costuma ser guardado em variáveis para identificar um socket, e pode ser devolvido por outros métodos (como `Server.Accept()` ou `Client.NextFree()`).

Um socket cliente sempre tem um determinado tipo, que pode ser consultado com `Client[].Type`. Esse tipo decide quais métodos ficam disponíveis para o socket cliente. Os seguintes tipos são usados:

Tipo | Descrição

`SOCK_DESTINY`: o socket usa o `protocolo Destiny`.

`SOCK_RAW`: o socket não usa nenhum protocolo especial (o protocolo precisa ser implementado pelo próprio criador do DestinyScript).

## Lista de parâmetros

### Parâmetro: `Cliente-Índice`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... (Client.Count - 1)
- **Descrição:** O índice do socket cliente. Precisa estar na faixa de 0 a (`Client.Count` - 1).

## Erros possíveis

`DESTINYERROR_ARRAYBOUND`

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Propriedades gerais` | Categoria | Nesta categoria estão reunidas as propriedades gerais de um socket cliente. |
| `Destiny` | Objeto | O objeto Destiny dentro do objeto Client[] serve para reunir todos os métodos disponíveis apenas para sockets cliente do tipo Destiny. |
| `Raw` | Objeto | O objeto Raw dentro do objeto Client[] serve para reunir todos os métodos disponíveis apenas para sockets cliente do tipo Raw. |
| `Métodos obsoletos` | Categoria | Nesta categoria estão reunidos os métodos obsoletos de um socket cliente. |
| `Dados de conexão` | Categoria | Nesta categoria estão as propriedades relativas à conexão. |

CATEGORIA

# Categoria: Propriedades gerais

## Descrição

Nesta categoria estão reunidas as propriedades gerais de um socket cliente.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Close()` | Método | Com o método Close é possível fechar um socket cliente aberto. |
| `Connect()` | Método | Com o método Connect é possível estabelecer uma conexão com outro computador. |
| `GetRecvLength()` | Método | Com o método GetRecvLength é possível descobrir o comprimento (em bytes) dos dados recebidos por um socket cliente. |
| `State` | Propriedade | Com a propriedade State é possível consultar o status de conexão do socket cliente. |
| `Type` | Propriedade | Com a propriedade Type é possível consultar o tipo de um socket cliente. |

MÉTODO

# Método: Client[].Close()

## Sintaxe

```
Client[Cliente-Índice].Close()
```

## Descrição

Com o método Close é possível fechar um socket cliente aberto. Se o socket cliente já estiver fechado, nada acontece.

## Exemplo

```
$
// Estabelecer conexão com o Google
Client[0].Connect("www.google.de", 80, SOCK_RAW);

if(Client[0].State != STATE_CONNECTED)
  // NÃO foi possível estabelecer conexão com o Google
else
  // Foi possível estabelecer conexão com o Google
  // -> o socket está pronto para enviar/rece

  // Enviar uma requisição simples ao Google
  Client[0].Raw.Send(<<<HEREDOC_END
GET / HTTP/1.1
Host: www.google.de
Connection: close

HEREDOC_END);

  // Aguardar a resposta e guardá-la em a[1]
  a[1] = "";
  do until(Client[0].State != STATE_CONNECTED)
    if(Client[0].GetRecvLength() > 0)
      a[1] ..= Client[0].Raw.Recv();
      if(String.Pos(a[1], CRLF .. CRLF) >= 0)
        // os cabeçalhos de resposta HTTP terminam com uma l
        Client[0].Close();
      endif;
    endif;
  loop;
endif;

// Fechar o socket novamente
Client[0].Close();
```

## Erros possíveis

`DESTINYERROR_VALUE`, `DESTINYERROR_SOCKETSTARTUP`, `DESTINYERROR_SOCKETSTILLOPEN`, `DESTINYERROR_CANTCREATESOCKET`, `DESTINYERROR_CANTCONNECT`

## Veja também

`Client`, `Client[]`, `Client[].Destiny`, `Client[].Raw`, `Client[].Raw.Recv()`, `Client[].Type`, `Client[].State`, `Client[].Connect()`, `Client[].GetRecvLength()`

MÉTODO

# Método: Client[].Connect()

## Sintaxe

```
Client[Cliente-Índice].Connect(Zielhost, Zielport, [Protokoll,
[Flags]])
```

## Descrição

Com o método Connect é possível estabelecer uma conexão com outro computador. Para isso, o outro computador precisa aceitar conexões de entrada. A chamada deste método só é concluída quando a conexão é estabelecida ou ocorre um timeout. Se não for possível estabelecer conexão, é lançado um erro.

## Lista de parâmetros

### Parâmetro: `Host de destino`

- **Tipo de dado:** `string Optional:nein`
- **Descrição:** Indica o endereço de destino do servidor. Pode ser um nome de host (por exemplo, "www.google.de") ou um endereço IP (IPv4; por exemplo, "127.0.0.1").

### Parâmetro: `Porta de destino`

- **Tipo de dado:** `dword Optional:nein`
- **Faixa de valores:** Indica a porta de destino (TCP) do servidor. O servidor precisa ter uma porta de entrada aberta no endereço indicado.

### Parâmetro: `Protocolo`

- **Tipo de dado:** `dword`
- **Opcional:** ja Standardwert: SOCK_DESTINY
- **Faixa de valores:** 0 ... 1
- **Descrição:** Indica o protocolo a ser usado. Para isso são usadas as constantes de `tipo de socket`:

Constante | Descrição

`SOCK_DESTINY`: o socket usa o `protocolo Destiny`. Veja o subobjeto `Destiny`.

`SOCK_RAW`: o socket não usa nenhum protocolo especial (o protocolo precisa ser implementado pelo próprio criador do DestinyScript). Veja o subobjeto `Raw`.

### Parâmetro: `Flags`

- **Tipo de dado:** `dword`
- **Opcional:** ja Standardwert: 0
- **Descrição:** Indica flags adicionais para o socket. Vários valores são combinados com o operador OU binário ( | ). Para isso são usadas as constantes de `flag de socket`:

Constante | Descrição

`SOCKETFLAG_NOTBLOCKING`: indica que o DestinyScript (e

portanto o RPG_RT) não deve esperar até que a tentativa de conexão termine. Isso é útil, por exemplo, quando se tenta conectar a um endereço inexistente, já que caso contrário o RPG_RT ficaria travado por alguns segundos (até o timeout da tentativa de conexão).

Para verificar se a conexão foi estabelecida ou se ocorreu um timeout, é preciso verificar a propriedade `State`:

Durante uma tentativa de conexão, ela tem o valor `STATE_CONNECTING`. Se a tentativa de conexão for bem-sucedida, ela passa a ter o valor `STATE_CONNECTED`. Se a tentativa de conexão falhar (por exemplo, após um timeout), ela passa a ter o valor `STATE_ERROR`.

O uso desse flag também evita que ocorra o erro `DESTINYERROR_CANTCONNECT` por o computador de destino não aceitar a conexão. O erro ainda pode ocorrer se o endereço de destino não existir.

## Exemplo

```
$
// Estabelecer conexão com o Google
Client[0].Connect("www.google.de", 80, SOCK_RAW);

if(Client[0].State != STATE_CONNECTED)
  // NÃO foi possível estabelecer conexão com o Google
else
  // Foi possível estabelecer conexão com o Google
  // -> o socket está pronto para enviar/rece

  // Enviar uma requisição simples ao Google
  Client[0].Raw.Send(<<<HEREDOC_END
GET / HTTP/1.1
Host: www.google.de
Connection: close

HEREDOC_END);

  // Aguardar a resposta e guardá-la em a[1]
  a[1] = "";
  do until(Client[0].State != STATE_CONNECTED)
    if(Client[0].GetRecvLength() > 0)
      a[1] ..= Client[0].Raw.Recv();
      if(String.Pos(a[1], CRLF .. CRLF) >= 0)
        // os cabeçalhos de resposta HTTP terminam com uma l
        Client[0].Close();
      endif;
    endif;
  loop;
endif;

// Fechar o socket novamente
Client[0].Close();
```

Exemplo de uso de um socket não bloqueante:

```
$

// Estabelecer conexão com um IP não utilizado
// (127.0.0.2 é reservado e não é usado
Client[0].Connect("127.0.0.2", 1, SOCK_DESTINY, S

do while(Client[0].State == STATE_CONNECTING)
  pause;  // Para que o RPG_RT não trave
```

```
loop;

if(Client[0].State == STATE_CONNECTED)
  // A tentativa de conexão foi bem-sucedida
  // -> enviar/receber dados

  // ...
else
  // A tentativa de conexão falhou!
endif;

Client[0].Close();
```

## Erros possíveis

`DESTINYERROR_VALUE`, `DESTINYERROR_SOCKETSTARTUP`, `DESTINYERROR_SOCKETSTILLOPEN`, `DESTINYERROR_CANTCREATESOCKET`, `DESTINYERROR_CANTCONNECT`, `DESTINYERROR_SOCKETERROR`

## Veja também

`Client`, `Client[]`, `Client[].Destiny`, `Client[].Raw`, `Client[].Raw.Recv()`, `Client[].Type`, `Client[].State`, `Client[].Close()`, `Client[].GetRecvLength()`

MÉTODO

# Método: Client[].GetRecvLength()

## Sintaxe

```
Client[Cliente-Índice].GetRecvLength()
```

## Descrição

Com o método GetRecvLength é possível descobrir o comprimento (em bytes) dos dados recebidos por um socket cliente. Em um socket `Raw`, este método precisa ser chamado antes do método `Recv`, para verificar se há dados de entrada pendentes (e recebê-los, se for o caso). Em um socket `Destiny`, é preciso chamar este método ou o método `GetRecvType`, para verificar se há dados de entrada pendentes (e recebê-los, se for o caso).

## Exemplo

```
$
// Estabelecer conexão com o Google
Client[0].Connect("www.google.de", 80, SOCK_RAW);

if(Client[0].State != STATE_CONNECTED)
  // NÃO foi possível estabelecer conexão com o Google
else
  // Foi possível estabelecer conexão com o Google
  // -> o socket está pronto para enviar/rece

  // Enviar uma requisição simples ao Google
  Client[0].Raw.Send(<<<HEREDOC_END
GET / HTTP/1.1
Host: www.google.de
Connection: close
```

```
HEREDOC_END);

  // Aguardar a resposta e guardá-la em a[1]
  a[1] = "";
  do until(Client[0].State != STATE_CONNECTED)
    if(Client[0].GetRecvLength() > 0)
      a[1] ..= Client[0].Raw.Recv();
      if(String.Pos(a[1], CRLF .. CRLF) >= 0)
        // os cabeçalhos de resposta HTTP terminam com uma l
        Client[0].Close();
      endif;
    endif;
  loop;
endif;

// Fechar o socket novamente
Client[0].Close();
```

## Observações adicionais

Ao chamar este método, é verificado internamente se há novos dados pendentes.

## Erros possíveis

`DESTINYERROR_VALUE`, `DESTINYERROR_SOCKETSTARTUP`, `DESTINYERROR_SOCKETSTILLOPEN`, `DESTINYERROR_CANTCREATESOCKET`, `DESTINYERROR_CANTCONNECT`

## Veja também

`Client`, `Client[]`, `Client[].Destiny`, `Client[].Raw`, `Client[].Raw.Recv()`, `Client[].Type`, `Client[].State`, `Client[].Connect()`, `Client[].Close()`

PROPRIEDADE

# Propriedade: Client[].State

## Sintaxe

```
Client[Cliente-Índice].State
```

## Descrição

Com a propriedade State é possível consultar o status de conexão do socket cliente. Para isso são usadas as constantes de `status de socket`:

Constante | Descrição

Ocorreu um erro. A única coisa que se pode fazer aqui é fechar o socket pelo método `Close` e depois

`STATE_ERROR` estabelecer uma nova conexão.

Se uma tentativa de conexão não bloqueante falhar, o socket fica com esse status em seguida.

O socket está fechado no momento. Esse é o status padrão de um socket cliente. Um

`STATE_CLOSED` socket pode ser colocado explicitamente nesse status

chamando-se o método `Close`.

O socket está conectado no momento. Portanto, dados podem ser enviados e recebidos. Esse status pode ser alcançado por o socket ter se conectado a um servidor (via

`STATE_CONNECTED`

`Client[].Connect()`) ou por o socket servidor ter sido usado para aceitar uma conexão de entrada (via `Server.Listen()` seguido de `Server.Accept()`).

O socket foi fechado pelo outro lado. Os dados pendentes ainda podem ser recebidos, mas não é mais possível enviar dados. Se o socket não for mais

`STATE_CLOSING` necessário (por exemplo, se os dados não forem mais

necessários), ele ainda precisa ser fechado pelo método `Close`. Assim que todos os dados pendentes forem recebidos, o socket é fechado automaticamente.

`STATE_CONNECTING` O socket ainda está tentando estabelecer uma

conexão. Esse status só está disponível em uma tentativa de conexão não bloqueante. Com o parâmetro Flags do método `Connect`, indica-se que a tentativa de conexão deve ser não bloqueante.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** -1; 0; 1; 3; 4
- **Volátil:** sim

## Exemplo

```
$
// Estabelecer conexão com o Google
Client[0].Connect("www.google.de", 80, SOCK_RAW);

if(Client[0].State != STATE_CONNECTED)
  // NÃO foi possível estabelecer conexão com o Google
else
  // Foi possível estabelecer conexão com o Google
  // -> o socket está pronto para enviar/rece
endif;

// Fechar o socket novamente
Client[0].Close();
```

## Observações adicionais

Ao chamar este método, é verificado internamente se há novos dados pendentes.

## Veja também

`Client`, `Client[]`, `Client[].Destiny`, `Client[].Raw`, `Client[].Type`, `Client[].Connect()`, `Client[].Close()`

PROPRIEDADE

# Propriedade: Client[].Type

## Sintaxe

```
Client[Cliente-Índice].Type
```

## Descrição

Com a propriedade Type é possível consultar o tipo de um socket cliente. Para isso são usadas as constantes de `tipo de socket`:

Constante | Descrição

`SOCK_DESTINY`: o socket usa o `protocolo Destiny`. Veja o subobjeto `Destiny`.

`SOCK_RAW`: o socket não usa nenhum protocolo especial (o protocolo precisa ser implementado pelo próprio criador do DestinyScript). Veja o subobjeto `Raw`.

> **Informação:** O tipo de um socket é definido na chamada do método `Connect` do objeto `Client[]`, ou no método `Listen` do objeto `Server`.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** 0 ... 1

## Exemplo

```
$
// Suposição: Client[0] é um socket conectado

if(Client[0].Type == SOCK_DESTINY)
  // Acesso a Client[0].Destiny pode ser feito
  Client[0].Destiny.SendString(0, "Hallo");
elseif(Client[0].Type == SOCK_RAW)
  // Acesso a Client[0].Raw pode ser feito
  Client[0].Raw.Send("Hallo");
endif;
```

## Veja também

`Client`, `Client[]`, `Client[].Destiny`, `Client[].Raw`, `Client[].Destiny.SendString()`, `Client[].Raw.Send()`, `Client[].State`, `Client[].Connect()`, `Client[].Close()`

OBJETO

# Objeto: Client[].Destiny

## Sintaxe

```
Client[Cliente-Índice].Destiny
```

## Descrição

O objeto Destiny dentro do objeto `Client[]` serve para reunir todos os métodos disponíveis apenas para sockets cliente do tipo Destiny. O tipo de um socket pode ser obtido com `Client[].Type`, e é definido no método `Connect` do objeto `Client[]` ou no método `Listen` do objeto `Server`.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `GetRecvType()` | Método | Com o método GetRecvType é possível descobrir o tipo de dado dos dados recebidos. |
| `RecvByte()` | Método | Com o método RecvByte é possível obter o valor de um Byte recebido. |
| `RecvDword()` | Método | Com o método RecvDword é possível obter o valor de um Dword recebido. |
| `RecvFloat()` | Método | Com o método RecvFloat é possível obter o valor de um Float recebido. |
| `RecvId()` | Método | Com o método RecvId é possível descobrir o Id de um valor recebido. |
| `RecvString()` | Método | Com o método RecvString é possível obter o valor de uma String recebida. |
| `RecvSwitch()` | Método | Com o método RecvSwitch é possível obter o valor de um switch recebido. |
| `RecvVariable()` | Método | Com o método RecvVariable é possível obter o valor de uma variável recebida. |
| `RecvWord()` | Método | Com o método RecvWord é possível obter o valor de um Word recebido. |
| `SendByte()` | Método | Com o método SendByte é possível enviar um Byte por um socket cliente Destiny conectado. |
| `SendDword()` | Método | Com o método SendDword é possível enviar um Dword por um socket cliente Destiny conectado. |
| `SendFloat()` | Método | Com o método SendFloat é possível enviar um Float por um socket cliente Destiny conectado. |
| `SendString()` | Método | Com o método SendString é possível enviar uma String por um socket cliente Destiny conectado. |
| `SendSwitch()` | Método | Com o método SendSwitch é possível enviar um switch por um socket cliente Destiny conectado. |
| `SendVariable()` | Método | Com o método SendVariable é possível enviar um variável por um socket cliente Destiny conectado. |
| `SendWord()` | Método | Com o método SendWord é possível enviar um Word por um socket cliente Destiny conectado. |

## Veja também

`Client`, `Client[]`, `Client[].Raw`

MÉTODO

# Método: Client[].Destiny.GetRecvType()

## Sintaxe

```
Client[Cliente-Índice].Destiny.GetRecvType()
```

## Descrição

Com o método GetRecvType é possível descobrir o tipo de dado dos dados recebidos. Uma vez definido o tipo de dado a usar, pode-se usar o respectivo método de recebimento (por exemplo, `RecvVariable()`, se o valor de retorno deste método for `TYPE_VARIABLE`). Para descobrir o Id dos dados recebidos, é preciso chamar o método `RecvId` antes do respectivo método de recebimento do tipo de dado.

## Lista de parâmetros

### Valor de retorno

- **Tipo de dado:** `dword`
- **Descrição:** Indica o tipo dos dados recebidos.

Se o valor de retorno deste método for = 0, não há dados pendentes para receber no momento. Se o valor de retorno deste método for > 0, os dados foram recebidos por completo. Para identificar o tipo de dado, podem ser usadas as constantes citadas abaixo. Se o valor de retorno deste método for < 0, dados foram recebidos, mas ainda não estão totalmente disponíveis. Não é possível acessar esses dados ainda. Invertendo o sinal do valor de retorno, obtém-se o tipo de dado, que pode ser identificado com as constantes abaixo.

Para identificar o tipo de dado, são usadas as constantes de `tipo de dado`:

Constante | Descrição

`TYPE_VARIABLE`: foi recebida uma variável. Para receber, é preciso chamar o método `RecvVariable`.

`TYPE_SWITCH`: foi recebido um switch. Para receber, é preciso chamar o método `RecvSwitch`.

`TYPE_DWORD`: foi recebido um Dword. Para receber, é preciso chamar o método `RecvDword`.

`TYPE_FLOAT`: foi recebido um Float. Para receber, é preciso chamar o método `RecvFloat`.

`TYPE_STRING`: foi recebida uma String. Para receber, é preciso chamar o

método `RecvString`.

`TYPE_BYTE`: foi recebido um Byte. Para receber, é preciso chamar o método `RecvByte`.

`TYPE_WORD`: foi recebido um Word. Para receber, é preciso chamar o método `RecvWord`.

## Exemplo

> **Informação:** Para testar os exemplos a seguir localmente, recomenda-se iniciar o jogo duas vezes: uma como servidor e outra como cliente. O servidor precisa ser iniciado primeiro, pois caso contrário o cliente não consegue se conectar. Como o exemplo se conecta ao próprio computador (o IP 127.0.0.1 sempre representa o próprio computador), este exemplo deve funcionar mesmo atrás de roteadores/firewalls. Com o uso do comando pause, o DestinyScript dentro do laço é brevemente interrompido, para que o RPG_RT continue funcionando e o jogo não trave.

DestinyScript do servidor (recebe os dados aqui; precisa ser iniciado primeiro):

```
$

// ------------------------------------------
// PARTE 1: iniciar o servidor brevemente e aceitar a conexã
// ------------------------------------------

// Aguardar conexões de entrada na porta 12000
Server.Listen(12000, SOCK_DESTINY);

do
  v[1] = Server.Accept(Client.NextFree());
  pause;
loop until(v[1] != -1);

// A conexão foi aceita
// -> fechar o servidor (a conexão já aceita se
Server.Close();

// ------------------------------------------
// PARTE 2: receber os dados (assim que disponíveis)
// ------------------------------------------

// Aguardar dados de entrada
Destiny.Debugger.Output.Clear();
do
  v[2] = Client[v[1]].Destiny.GetRecvType();
  if(v[2] > 0)
    // Os dados estão disponíveis e prontos para serem recebidos
    // -> Primeiro consultar o Id
    v[3] = Client[v[1]].Destiny.RecvId();
    // -> Depois consultar o valor (dependendo do tipo)
    switch(v[2])
      case TYPE_VARIABLE:
        Destiny.Debugger.Output.Print("Variable e
        break;
      case TYPE_DWORD:
        Destiny.Debugger.Output.Print("Dword empf
        break;
      case TYPE_WORD:
        Destiny.Debugger.Output.Print("Word empfa
        break;
      case TYPE_BYTE:
        Destiny.Debugger.Output.Print("Byte empfa
        break;
      case TYPE_SWITCH:
        Destiny.Debugger.Output.Print("Switch emp
        break;
      case TYPE_FLOAT:
        Destiny.Debugger.Output.Print("Float empf
        break;
      case TYPE_STRING:
        Destiny.Debugger.Output.Print("String emp
        break;
    endswitch;
  endif;
  pause;
loop while(v[2] > 0 || Client[v[1]].State == STAT

// Ao final do script, ainda fechar a conexão
Client[v[1]].Close();

// Exibir a saída de depuração
Destiny.Debugger.Output.Show();
```

DestinyScript do cliente (envia os dados aqui; o servidor já precisa estar rodando):

```
$

// ------------------------------------------
// PARTE 1: estabelecer conexão com o servidor (aqui
// ------------------------------------------
v[1] = Client.NextFree();
Client[v[1]].Connect("127.0.0.1", 12000, SOCK_DES

// ------------------------------------------
// PARTE 2: enviar os dados
// ------------------------------------------
Client[v[1]].Destiny.SendVariable(1, 1234);   //
d[1] = 4321;
Client[v[1]].Destiny.SendDword(2, d[1]);      //
Client[v[1]].Destiny.SendWord(13, -11);       //
Client[v[1]].Destiny.SendByte(0, 255);        //
Client[v[1]].Destiny.SendFloat(-1, 3.1415);   //
Client[v[1]].Destiny.SendSwitch(1, true);     //
Client[v[1]].Destiny.SendString(2, "Hallo");  //

// Fechar a conexão (isso faz o exemplo do servidor
Client[v[1]].Close();
```

## Observações adicionais

Ao chamar este método, é verificado internamente se há dados novos pendentes.

## Erros possíveis

`DESTINYERROR_SOCKETNOTOPEN`, `DESTINYERROR_SOCKETTYPE`, `DESTINYERROR_SOCKETERROR`, `DESTINYERROR_OOB`

## Veja também

`Client`, `Client[]`, `Client[].Destiny`, `Server`, `Server.Listen()`, `Server.Accept()`, `Server.Close()`, `Client[].Connect()`, `Client[].Close()`, `Client[].Destiny.SendVariable()`, `Client[].Destiny.SendDword()`, `Client[].Destiny.SendWord()`, `Client[].Destiny.SendByte()`, `Client[].Destiny.SendFloat()`, `Client[].Destiny.SendString()`, `Client[].Destiny.SendSwitch()`, `Client[].Destiny.RecvId()`, `Client[].Destiny.RecvVariable()`, `Client[].Destiny.RecvDword()`, `Client[].Destiny.RecvWord()`, `Client[].Destiny.RecvByte()`, `Client[].Destiny.RecvFloat()`, `Client[].Destiny.RecvString()`, `Client[].Destiny.RecvSwitch()`

MÉTODO

# Método: Client[].Destiny.RecvByte()

## Sintaxe

```
Client[Cliente-Índice].Destiny.RecvByte()
```

## Descrição

Com o método RecvByte é possível obter o valor de um Byte recebido. Ao chamar este método, o Byte recebido é automaticamente removido dos dados recebidos do socket cliente, de modo que o valor desse Byte, bem como o Id

associado, não podem mais ser consultados depois. Por isso, o Id precisa ser consultado antes do valor do Byte. A chamada deste método só é bem-sucedida quando `GetRecvType()` tem o valor de retorno `TYPE_BYTE`. Depois que o Byte for recebido, os valores eventualmente enviados em seguida podem ser recebidos.

## Lista de parâmetros

### Valor de retorno

- **Tipo de dado:** `byte`
- **Descrição:** O valor do Byte recebido. Esse é o valor que foi informado em `SendByte()`.

## Exemplo

> **Informação:** Para testar os exemplos a seguir localmente, recomenda-se iniciar o jogo duas vezes: uma como servidor e outra como cliente. O servidor precisa ser iniciado primeiro, pois caso contrário o cliente não consegue se conectar. Como o exemplo se conecta ao próprio computador (o IP 127.0.0.1 sempre representa o próprio computador), este exemplo deve funcionar mesmo atrás de roteadores/firewalls. Com o uso do comando pause, o DestinyScript dentro do laço é brevemente interrompido, para que o RPG_RT continue funcionando e o jogo não trave.

DestinyScript do servidor (recebe os dados aqui; precisa ser iniciado primeiro):

```
$

// ------------------------------------------
// PARTE 1: iniciar o servidor brevemente e aceitar a conexã
// ------------------------------------------

// Aguardar conexões de entrada na porta 12000
Server.Listen(12000, SOCK_DESTINY);

do
  v[1] = Server.Accept(Client.NextFree());
  pause;
loop until(v[1] != -1);

// A conexão foi aceita
// -> fechar o servidor (a conexão já aceita se
Server.Close();

// ------------------------------------------
// PARTE 2: receber os dados (assim que disponíveis)
// ------------------------------------------

// Aguardar dados de entrada
Destiny.Debugger.Output.Clear();
do
  v[2] = Client[v[1]].Destiny.GetRecvType();
  if(v[2] > 0)
    // Os dados estão disponíveis e prontos para serem recebidos
    // -> Primeiro consultar o Id
    v[3] = Client[v[1]].Destiny.RecvId();
    // -> Depois consultar o valor (dependendo do tipo)
    switch(v[2])
      case TYPE_VARIABLE:
        Destiny.Debugger.Output.Print("Variable e
        break;
      case TYPE_DWORD:
        Destiny.Debugger.Output.Print("Dword empf
```

```
        break;
      case TYPE_WORD:
        Destiny.Debugger.Output.Print("Word empfa
        break;
      case TYPE_BYTE:
        Destiny.Debugger.Output.Print("Byte empfa
        break;
      case TYPE_SWITCH:
        Destiny.Debugger.Output.Print("Switch emp
        break;
      case TYPE_FLOAT:
        Destiny.Debugger.Output.Print("Float empf
        break;
      case TYPE_STRING:
        Destiny.Debugger.Output.Print("String emp
        break;
    endswitch;
  endif;
  pause;
loop while(v[2] > 0 || Client[v[1]].State == STAT

// Ao final do script, ainda fechar a conexão
Client[v[1]].Close();

// Exibir a saída de depuração
Destiny.Debugger.Output.Show();
```

DestinyScript do cliente (envia os dados aqui; o servidor já precisa estar rodando):

```
$

// ---------------------------------------------
// PARTE 1: estabelecer conexão com o servidor (aqui
// ---------------------------------------------
v[1] = Client.NextFree();
Client[v[1]].Connect("127.0.0.1", 12000, SOCK_DES

// ---------------------------------------------
// PARTE 2: enviar os dados
// ---------------------------------------------
Client[v[1]].Destiny.SendVariable(1, 1234);   //
d[1] = 4321;
Client[v[1]].Destiny.SendDword(2, d[1]);      //
Client[v[1]].Destiny.SendWord(13, -11);       //
Client[v[1]].Destiny.SendByte(0, 255);        //
Client[v[1]].Destiny.SendFloat(-1, 3.1415);   //
Client[v[1]].Destiny.SendSwitch(1, true);     //
Client[v[1]].Destiny.SendString(2, "Hallo");  //

// Fechar a conexão (isso faz o exemplo do servidor
Client[v[1]].Close();
```

## Erros possíveis

`DESTINYERROR_SOCKETTYPE`, `DESTINYERROR_SOCKETERROR`, `DESTINYERROR_OOB`

## Veja também

`Client`, `Client[]`, `Client[].Destiny`, `Server`, `Server.Listen()`, `Server.Accept()`, `Server.Close()`, `Client[].Connect()`, `Client[].Close()`, `Client[].Destiny.SendVariable()`, `Client[].Destiny.SendDword()`, `Client[].Destiny.SendWord()`, `Client[].Destiny.SendByte()`, `Client[].Destiny.SendFloat()`, `Client[].Destiny.SendString()`, `Client[].Destiny.SendSwitch()`,

`Client[].Destiny.GetRecvType()`, `Client[].Destiny.RecvId()`, `Client[].Destiny.RecvVariable()`, `Client[].Destiny.RecvDword()`, `Client[].Destiny.RecvWord()`, `Client[].Destiny.RecvFloat()`, `Client[].Destiny.RecvString()`, `Client[].Destiny.RecvSwitch()`

MÉTODO

# Método: Client[].Destiny.RecvDword()

## Sintaxe

```
Client[Cliente-Índice].Destiny.RecvDword()
```

## Descrição

Com o método RecvDword é possível obter o valor de um Dword recebido. Ao chamar este método, o Dword recebido é automaticamente removido dos dados recebidos do socket cliente, de modo que o valor desse Dword, bem como o Id associado, não podem mais ser consultados depois. Por isso, o Id precisa ser consultado antes do valor do Dword. A chamada deste método só é bem-sucedida quando `GetRecvType()` tem o valor de retorno `TYPE_DWORD`. Depois que o Dword for recebido, os valores eventualmente enviados em seguida podem ser recebidos.

## Lista de parâmetros

### Valor de retorno

- **Tipo de dado:** `dword`
- **Descrição:** O valor do Dword recebido. Esse é o valor que foi informado em `SendDword()`.

## Exemplo

> **Informação:** Para testar os exemplos a seguir localmente, recomenda-se iniciar o jogo duas vezes: uma como servidor e outra como cliente. O servidor precisa ser iniciado primeiro, pois caso contrário o cliente não consegue se conectar. Como o exemplo se conecta ao próprio computador (o IP 127.0.0.1 sempre representa o próprio computador), este exemplo deve funcionar mesmo atrás de roteadores/firewalls. Com o uso do comando pause, o DestinyScript dentro do laço é brevemente interrompido, para que o RPG_RT continue funcionando e o jogo não trave.

DestinyScript do servidor (recebe os dados aqui; precisa ser iniciado primeiro):

```
$

// ------------------------------------------------
// PARTE 1: iniciar o servidor brevemente e aceitar a conexã
// ------------------------------------------------

// Aguardar conexões de entrada na porta 12000
Server.Listen(12000, SOCK_DESTINY);

do
  v[1] = Server.Accept(Client.NextFree());
  pause;
loop until(v[1] != -1);

// A conexão foi aceita
```

```
// -> fechar o servidor (a conexão já aceita se
Server.Close();

// ---------------------------------------------
// PARTE 2: receber os dados (assim que disponíveis)
// ---------------------------------------------

// Aguardar dados de entrada
Destiny.Debugger.Output.Clear();
do
  v[2] = Client[v[1]].Destiny.GetRecvType();
  if(v[2] > 0)
    // Os dados estão disponíveis e prontos para serem recebidos
    // -> Primeiro consultar o Id
    v[3] = Client[v[1]].Destiny.RecvId();
    // -> Depois consultar o valor (dependendo do tipo)
    switch(v[2])
      case TYPE_VARIABLE:
        Destiny.Debugger.Output.Print("Variable e
        break;
      case TYPE_DWORD:
        Destiny.Debugger.Output.Print("Dword empf
        break;
      case TYPE_WORD:
        Destiny.Debugger.Output.Print("Word empfa
        break;
      case TYPE_BYTE:
        Destiny.Debugger.Output.Print("Byte empfa
        break;
      case TYPE_SWITCH:
        Destiny.Debugger.Output.Print("Switch emp
        break;
      case TYPE_FLOAT:
        Destiny.Debugger.Output.Print("Float empf
        break;
      case TYPE_STRING:
        Destiny.Debugger.Output.Print("String emp
        break;
    endswitch;
  endif;
  pause;
loop while(v[2] > 0 || Client[v[1]].State == STAT

// Ao final do script, ainda fechar a conexão
Client[v[1]].Close();

// Exibir a saída de depuração
Destiny.Debugger.Output.Show();
```

DestinyScript do cliente (envia os dados aqui; o servidor já precisa estar rodando):

```
$

// ---------------------------------------------
// PARTE 1: estabelecer conexão com o servidor (aqui
// ---------------------------------------------
v[1] = Client.NextFree();
Client[v[1]].Connect("127.0.0.1", 12000, SOCK_DES

// ---------------------------------------------
// PARTE 2: enviar os dados
// ---------------------------------------------
Client[v[1]].Destiny.SendVariable(1, 1234);   //
d[1] = 4321;
Client[v[1]].Destiny.SendDword(2, d[1]);      //
```

```
Client[v[1]].Destiny.SendWord(13, -11);       //
Client[v[1]].Destiny.SendByte(0, 255);        //
Client[v[1]].Destiny.SendFloat(-1, 3.1415);   //
Client[v[1]].Destiny.SendSwitch(1, true);     //
Client[v[1]].Destiny.SendString(2, "Hallo");  //

// Fechar a conexão (isso faz o exemplo do servidor
Client[v[1]].Close();
```

## Erros possíveis

`DESTINYERROR_SOCKETTYPE`, `DESTINYERROR_SOCKETERROR`, `DESTINYERROR_OOB`

## Veja também

`Client`, `Client[]`, `Client[].Destiny`, `Server`, `Server.Listen()`, `Server.Accept()`, `Server.Close()`, `Client[].Connect()`, `Client[].Close()`, `Client[].Destiny.SendVariable()`, `Client[].Destiny.SendDword()`, `Client[].Destiny.SendWord()`, `Client[].Destiny.SendByte()`, `Client[].Destiny.SendFloat()`, `Client[].Destiny.SendString()`, `Client[].Destiny.SendSwitch()`, `Client[].Destiny.GetRecvType()`, `Client[].Destiny.RecvId()`, `Client[].Destiny.RecvVariable()`, `Client[].Destiny.RecvWord()`, `Client[].Destiny.RecvByte()`, `Client[].Destiny.RecvFloat()`, `Client[].Destiny.RecvString()`, `Client[].Destiny.RecvSwitch()`

MÉTODO

# Método: Client[].Destiny.RecvFloat()

## Sintaxe

```
Client[Cliente-Índice].Destiny.RecvFloat()
```

## Descrição

Com o método RecvFloat é possível obter o valor de um Float recebido. Ao chamar este método, o Float recebido é automaticamente removido dos dados recebidos do socket cliente, de modo que o valor desse Float, bem como o Id associado, não podem mais ser consultados depois. Por isso, o Id precisa ser consultado antes do valor do Float. A chamada deste método só é bem-sucedida quando `GetRecvType()` tem o valor de retorno `TYPE_FLOAT`. Depois que o Float for recebido, os valores eventualmente enviados em seguida podem ser recebidos.

## Lista de parâmetros

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** O valor do Float recebido. Esse é o valor que foi informado em `SendFloat()`.

## Exemplo

> **Informação:** Para testar os exemplos a seguir localmente, recomenda-se iniciar o jogo duas vezes: uma como servidor e outra como cliente. O servidor precisa ser iniciado primeiro, pois caso contrário o cliente não consegue se conectar. Como o exemplo se conecta ao próprio computador (o IP 127.0.0.1 sempre representa

> o próprio computador), este exemplo deve funcionar mesmo atrás de roteadores/firewalls. Com o uso do comando pause, o DestinyScript dentro do laço é brevemente interrompido, para que o RPG_RT continue funcionando e o jogo não trave.

DestinyScript do servidor (recebe os dados aqui; precisa ser iniciado primeiro):

```
$

// ------------------------------------------
// PARTE 1: iniciar o servidor brevemente e aceitar a conexã
// ------------------------------------------

// Aguardar conexões de entrada na porta 12000
Server.Listen(12000, SOCK_DESTINY);

do
  v[1] = Server.Accept(Client.NextFree());
  pause;
loop until(v[1] != -1);

// A conexão foi aceita
// -> fechar o servidor (a conexão já aceita se
Server.Close();

// ------------------------------------------
// PARTE 2: receber os dados (assim que disponíveis)
// ------------------------------------------

// Aguardar dados de entrada
Destiny.Debugger.Output.Clear();
do
  v[2] = Client[v[1]].Destiny.GetRecvType();
  if(v[2] > 0)
    // Os dados estão disponíveis e prontos para serem recebidos
    // -> Primeiro consultar o Id
    v[3] = Client[v[1]].Destiny.RecvId();
    // -> Depois consultar o valor (dependendo do tipo)
    switch(v[2])
      case TYPE_VARIABLE:
        Destiny.Debugger.Output.Print("Variable e
        break;
      case TYPE_DWORD:
        Destiny.Debugger.Output.Print("Dword empf
        break;
      case TYPE_WORD:
        Destiny.Debugger.Output.Print("Word empfa
        break;
      case TYPE_BYTE:
        Destiny.Debugger.Output.Print("Byte empfa
        break;
      case TYPE_SWITCH:
        Destiny.Debugger.Output.Print("Switch emp
        break;
      case TYPE_FLOAT:
        Destiny.Debugger.Output.Print("Float empf
        break;
      case TYPE_STRING:
        Destiny.Debugger.Output.Print("String emp
        break;
    endswitch;
  endif;
  pause;
loop while(v[2] > 0 || Client[v[1]].State == STAT

// Ao final do script, ainda fechar a conexão
```

```
Client[v[1]].Close();

// Exibir a saída de depuração
Destiny.Debugger.Output.Show();
```

DestinyScript do cliente (envia os dados aqui; o servidor já precisa estar rodando):

```
$

// ------------------------------------------
// PARTE 1: estabelecer conexão com o servidor (aqui
// ------------------------------------------
v[1] = Client.NextFree();
Client[v[1]].Connect("127.0.0.1", 12000, SOCK_DES

// ------------------------------------------
// PARTE 2: enviar os dados
// ------------------------------------------
Client[v[1]].Destiny.SendVariable(1, 1234);   //
d[1] = 4321;
Client[v[1]].Destiny.SendDword(2, d[1]);      //
Client[v[1]].Destiny.SendWord(13, -11);       //
Client[v[1]].Destiny.SendByte(0, 255);        //
Client[v[1]].Destiny.SendFloat(-1, 3.1415);   //
Client[v[1]].Destiny.SendSwitch(1, true);     //
Client[v[1]].Destiny.SendString(2, "Hallo");  //

// Fechar a conexão (isso faz o exemplo do servidor
Client[v[1]].Close();
```

## Erros possíveis

`DESTINYERROR_SOCKETTYPE`, `DESTINYERROR_SOCKETERROR`, `DESTINYERROR_OOB`

## Veja também

`Client`, `Client[]`, `Client[].Destiny`, `Server`, `Server.Listen()`, `Server.Accept()`, `Server.Close()`, `Client[].Connect()`, `Client[].Close()`, `Client[].Destiny.SendVariable()`, `Client[].Destiny.SendDword()`, `Client[].Destiny.SendWord()`, `Client[].Destiny.SendByte()`, `Client[].Destiny.SendFloat()`, `Client[].Destiny.SendString()`, `Client[].Destiny.SendSwitch()`, `Client[].Destiny.GetRecvType()`, `Client[].Destiny.RecvId()`, `Client[].Destiny.RecvVariable()`, `Client[].Destiny.RecvDword()`, `Client[].Destiny.RecvWord()`, `Client[].Destiny.RecvByte()`, `Client[].Destiny.RecvString()`, `Client[].Destiny.RecvSwitch()`

MÉTODO

# Método: Client[].Destiny.RecvId()

## Sintaxe

```
Client[Cliente-Índice].Destiny.RecvId()
```

## Descrição

Com o método RecvId é possível descobrir o Id de um valor recebido. Este método é usado da mesma forma para todos os tipos de dado, e precisa ser chamado antes do método de recebimento dos dados (por exemplo, RecvId()

antes de `RecvVariable()`). A chamada deste método só é bem-sucedida quando os dados já foram totalmente recebidos (ou seja, `GetRecvType()` precisa devolver um valor positivo).

## Lista de parâmetros

### Valor de retorno

- **Tipo de dado:** `dword`
- **Descrição:** O Id do valor recebido. Esse é o Id que foi informado nos métodos de envio (por exemplo, `SendVariable()`). O significado desse valor é definido pelo criador do DestinyScript.

## Exemplo

> **Informação:** Para testar os exemplos a seguir localmente, recomenda-se iniciar o jogo duas vezes: uma como servidor e outra como cliente. O servidor precisa ser iniciado primeiro, pois caso contrário o cliente não consegue se conectar. Como o exemplo se conecta ao próprio computador (o IP 127.0.0.1 sempre representa o próprio computador), este exemplo deve funcionar mesmo atrás de roteadores/firewalls. Com o uso do comando pause, o DestinyScript dentro do laço é brevemente interrompido, para que o RPG_RT continue funcionando e o jogo não trave.

DestinyScript do servidor (recebe os dados aqui; precisa ser iniciado primeiro):

```
$

// ------------------------------------------
// PARTE 1: iniciar o servidor brevemente e aceitar a conexã
// ------------------------------------------

// Aguardar conexões de entrada na porta 12000
Server.Listen(12000, SOCK_DESTINY);

do
  v[1] = Server.Accept(Client.NextFree());
  pause;
loop until(v[1] != -1);

// A conexão foi aceita
// -> fechar o servidor (a conexão já aceita se
Server.Close();

// ------------------------------------------
// PARTE 2: receber os dados (assim que disponíveis)
// ------------------------------------------

// Aguardar dados de entrada
Destiny.Debugger.Output.Clear();
do
  v[2] = Client[v[1]].Destiny.GetRecvType();
  if(v[2] > 0)
    // Os dados estão disponíveis e prontos para serem recebidos
    // -> Primeiro consultar o Id
    v[3] = Client[v[1]].Destiny.RecvId();
    // -> Depois consultar o valor (dependendo do tipo)
    switch(v[2])
      case TYPE_VARIABLE:
        Destiny.Debugger.Output.Print("Variable e
        break;
      case TYPE_DWORD:
        Destiny.Debugger.Output.Print("Dword empf
```

```
        break;
      case TYPE_WORD:
        Destiny.Debugger.Output.Print("Word empfa
        break;
      case TYPE_BYTE:
        Destiny.Debugger.Output.Print("Byte empfa
        break;
      case TYPE_SWITCH:
        Destiny.Debugger.Output.Print("Switch emp
        break;
      case TYPE_FLOAT:
        Destiny.Debugger.Output.Print("Float empf
        break;
      case TYPE_STRING:
        Destiny.Debugger.Output.Print("String emp
        break;
    endswitch;
  endif;
  pause;
loop while(v[2] > 0 || Client[v[1]].State == STAT

// Ao final do script, ainda fechar a conexão
Client[v[1]].Close();

// Exibir a saída de depuração
Destiny.Debugger.Output.Show();
```

DestinyScript do cliente (envia os dados aqui; o servidor já precisa estar rodando):

```
$

// ---------------------------------------------
// PARTE 1: estabelecer conexão com o servidor (aqui
// ---------------------------------------------
v[1] = Client.NextFree();
Client[v[1]].Connect("127.0.0.1", 12000, SOCK_DES

// ---------------------------------------------
// PARTE 2: enviar os dados
// ---------------------------------------------
Client[v[1]].Destiny.SendVariable(1, 1234);   //
d[1] = 4321;
Client[v[1]].Destiny.SendDword(2, d[1]);      //
Client[v[1]].Destiny.SendWord(13, -11);       //
Client[v[1]].Destiny.SendByte(0, 255);        //
Client[v[1]].Destiny.SendFloat(-1, 3.1415);   //
Client[v[1]].Destiny.SendSwitch(1, true);     //
Client[v[1]].Destiny.SendString(2, "Hallo");  //

// Fechar a conexão (isso faz o exemplo do servidor
Client[v[1]].Close();
```

## Erros possíveis

`DESTINYERROR_SOCKETTYPE`, `DESTINYERROR_SOCKETERROR`, `DESTINYERROR_OOB`

## Veja também

`Client`, `Client[]`, `Client[].Destiny`, `Server`, `Server.Listen()`, `Server.Accept()`, `Server.Close()`, `Client[].Connect()`, `Client[].Close()`, `Client[].Destiny.SendVariable()`, `Client[].Destiny.SendDword()`, `Client[].Destiny.SendWord()`, `Client[].Destiny.SendByte()`, `Client[].Destiny.SendFloat()`, `Client[].Destiny.SendString()`, `Client[].Destiny.SendSwitch()`,

`Client[].Destiny.GetRecvType()`, `Client[].Destiny.RecvVariable()`, `Client[].Destiny.RecvDword()`, `Client[].Destiny.RecvWord()`, `Client[].Destiny.RecvByte()`, `Client[].Destiny.RecvFloat()`, `Client[].Destiny.RecvString()`, `Client[].Destiny.RecvSwitch()`

MÉTODO

# Método: Client[].Destiny.RecvString()

## Sintaxe

```
Client[Cliente-Índice].Destiny.RecvString()
```

## Descrição

Com o método RecvString é possível obter o valor de uma String recebida. Ao chamar este método, a String recebida é automaticamente removida dos dados recebidos do socket cliente, de modo que o valor dessa String, bem como o Id associado, não podem mais ser consultados depois. Por isso, o Id precisa ser consultado antes do valor da String. A chamada deste método só é bem-sucedida quando `GetRecvType()` tem o valor de retorno `TYPE_STRING`. Depois que a String for recebida, os valores eventualmente enviados em seguida podem ser recebidos.

## Lista de parâmetros

### Valor de retorno

- **Tipo de dado:** `string`
- **Descrição:** O valor da String recebida. Esse é o valor que foi informado em `SendString()`.

## Exemplo

> **Informação:** Para testar os exemplos a seguir localmente, recomenda-se iniciar o jogo duas vezes: uma como servidor e outra como cliente. O servidor precisa ser iniciado primeiro, pois caso contrário o cliente não consegue se conectar. Como o exemplo se conecta ao próprio computador (o IP 127.0.0.1 sempre representa o próprio computador), este exemplo deve funcionar mesmo atrás de roteadores/firewalls. Com o uso do comando pause, o DestinyScript dentro do laço é brevemente interrompido, para que o RPG_RT continue funcionando e o jogo não trave.

DestinyScript do servidor (recebe os dados aqui; precisa ser iniciado primeiro):

```
$

// ------------------------------------------------
// PARTE 1: iniciar o servidor brevemente e aceitar a conexã
// ------------------------------------------------

// Aguardar conexões de entrada na porta 12000
Server.Listen(12000, SOCK_DESTINY);

do
  v[1] = Server.Accept(Client.NextFree());
  pause;
loop until(v[1] != -1);

// A conexão foi aceita
```

```
// -> fechar o servidor (a conexão já aceita se
Server.Close();

// ---------------------------------------------
// PARTE 2: receber os dados (assim que disponíveis)
// ---------------------------------------------

// Aguardar dados de entrada
Destiny.Debugger.Output.Clear();
do
  v[2] = Client[v[1]].Destiny.GetRecvType();
  if(v[2] > 0)
    // Os dados estão disponíveis e prontos para serem recebidos
    // -> Primeiro consultar o Id
    v[3] = Client[v[1]].Destiny.RecvId();
    // -> Depois consultar o valor (dependendo do tipo)
    switch(v[2])
      case TYPE_VARIABLE:
        Destiny.Debugger.Output.Print("Variable e
        break;
      case TYPE_DWORD:
        Destiny.Debugger.Output.Print("Dword empf
        break;
      case TYPE_WORD:
        Destiny.Debugger.Output.Print("Word empfa
        break;
      case TYPE_BYTE:
        Destiny.Debugger.Output.Print("Byte empfa
        break;
      case TYPE_SWITCH:
        Destiny.Debugger.Output.Print("Switch emp
        break;
      case TYPE_FLOAT:
        Destiny.Debugger.Output.Print("Float empf
        break;
      case TYPE_STRING:
        Destiny.Debugger.Output.Print("String emp
        break;
    endswitch;
  endif;
  pause;
loop while(v[2] > 0 || Client[v[1]].State == STAT

// Ao final do script, ainda fechar a conexão
Client[v[1]].Close();

// Exibir a saída de depuração
Destiny.Debugger.Output.Show();
```

DestinyScript do cliente (envia os dados aqui; o servidor já precisa estar rodando):

```
$

// ---------------------------------------------
// PARTE 1: estabelecer conexão com o servidor (aqui
// ---------------------------------------------
v[1] = Client.NextFree();
Client[v[1]].Connect("127.0.0.1", 12000, SOCK_DES

// ---------------------------------------------
// PARTE 2: enviar os dados
// ---------------------------------------------
Client[v[1]].Destiny.SendVariable(1, 1234);   //
d[1] = 4321;
Client[v[1]].Destiny.SendDword(2, d[1]);      //
```

```
Client[v[1]].Destiny.SendWord(13, -11);       //
Client[v[1]].Destiny.SendByte(0, 255);        //
Client[v[1]].Destiny.SendFloat(-1, 3.1415);   //
Client[v[1]].Destiny.SendSwitch(1, true);     //
Client[v[1]].Destiny.SendString(2, "Hallo");  //

// Fechar a conexão (isso faz o exemplo do servidor
Client[v[1]].Close();
```

## Erros possíveis

`DESTINYERROR_SOCKETTYPE`, `DESTINYERROR_SOCKETERROR`, `DESTINYERROR_OOB`, `DESTINYERROR_MEMORY`

## Veja também

`Client`, `Client[]`, `Client[].Destiny`, `Server`, `Server.Listen()`, `Server.Accept()`, `Server.Close()`, `Client[].Connect()`, `Client[].Close()`, `Client[].Destiny.SendVariable()`, `Client[].Destiny.SendDword()`, `Client[].Destiny.SendWord()`, `Client[].Destiny.SendByte()`, `Client[].Destiny.SendFloat()`, `Client[].Destiny.SendString()`, `Client[].Destiny.SendSwitch()`, `Client[].Destiny.GetRecvType()`, `Client[].Destiny.RecvId()`, `Client[].Destiny.RecvVariable()`, `Client[].Destiny.RecvDword()`, `Client[].Destiny.RecvWord()`, `Client[].Destiny.RecvByte()`, `Client[].Destiny.RecvFloat()`, `Client[].Destiny.RecvSwitch()`

MÉTODO

# Método: Client[].Destiny.RecvSwitch()

## Sintaxe

```
Client[Cliente-Índice].Destiny.RecvSwitch()
```

## Descrição

Com o método RecvSwitch é possível obter o valor de um switch recebido. Ao chamar este método, o switch recebido é automaticamente removido dos dados recebidos do socket cliente, de modo que o valor desse switch, bem como o Id associado, não podem mais ser consultados depois. Por isso, o Id precisa ser consultado antes do valor do switch. A chamada deste método só é bem-sucedida quando `GetRecvType()` tem o valor de retorno `TYPE_SWITCH`. Depois que o switch for recebido, os valores eventualmente enviados em seguida podem ser recebidos.

## Sinônimos

RecvBool()

## Lista de parâmetros

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** O valor do switch recebido. Esse é o valor que foi informado em `SendSwitch()`.

## Exemplo

> **Informação:** Para testar os exemplos a seguir localmente, recomenda-se iniciar o jogo duas vezes: uma como servidor e outra como cliente. O servidor precisa ser iniciado primeiro, pois caso contrário o cliente não consegue se conectar. Como o exemplo se conecta ao próprio computador (o IP 127.0.0.1 sempre representa o próprio computador), este exemplo deve funcionar mesmo atrás de roteadores/firewalls. Com o uso do comando pause, o DestinyScript dentro do laço é brevemente interrompido, para que o RPG_RT continue funcionando e o jogo não trave.

DestinyScript do servidor (recebe os dados aqui; precisa ser iniciado primeiro):

```
$

// ---------------------------------------------
// PARTE 1: iniciar o servidor brevemente e aceitar a conexã
// ---------------------------------------------

// Aguardar conexões de entrada na porta 12000
Server.Listen(12000, SOCK_DESTINY);

do
  v[1] = Server.Accept(Client.NextFree());
  pause;
loop until(v[1] != -1);

// A conexão foi aceita
// -> fechar o servidor (a conexão já aceita se
Server.Close();

// ---------------------------------------------
// PARTE 2: receber os dados (assim que disponíveis)
// ---------------------------------------------

// Aguardar dados de entrada
Destiny.Debugger.Output.Clear();
do
  v[2] = Client[v[1]].Destiny.GetRecvType();
  if(v[2] > 0)
    // Os dados estão disponíveis e prontos para serem recebidos
    // -> Primeiro consultar o Id
    v[3] = Client[v[1]].Destiny.RecvId();
    // -> Depois consultar o valor (dependendo do tipo)
    switch(v[2])
      case TYPE_VARIABLE:
        Destiny.Debugger.Output.Print("Variable e
        break;
      case TYPE_DWORD:
        Destiny.Debugger.Output.Print("Dword empf
        break;
      case TYPE_WORD:
        Destiny.Debugger.Output.Print("Word empfa
        break;
      case TYPE_BYTE:
        Destiny.Debugger.Output.Print("Byte empfa
        break;
      case TYPE_SWITCH:
        Destiny.Debugger.Output.Print("Switch emp
        break;
      case TYPE_FLOAT:
        Destiny.Debugger.Output.Print("Float empf
        break;
      case TYPE_STRING:
        Destiny.Debugger.Output.Print("String emp
```

```
        break;
    endswitch;
  endif;
  pause;
loop while(v[2] > 0 || Client[v[1]].State == STAT

// Ao final do script, ainda fechar a conexão
Client[v[1]].Close();

// Exibir a saída de depuração
Destiny.Debugger.Output.Show();
```

DestinyScript do cliente (envia os dados aqui; o servidor já precisa estar rodando):

```
$

// ------------------------------------------
// PARTE 1: estabelecer conexão com o servidor (aqui
// ------------------------------------------
v[1] = Client.NextFree();
Client[v[1]].Connect("127.0.0.1", 12000, SOCK_DES

// ------------------------------------------
// PARTE 2: enviar os dados
// ------------------------------------------
Client[v[1]].Destiny.SendVariable(1, 1234);   //
d[1] = 4321;
Client[v[1]].Destiny.SendDword(2, d[1]);      //
Client[v[1]].Destiny.SendWord(13, -11);       //
Client[v[1]].Destiny.SendByte(0, 255);        //
Client[v[1]].Destiny.SendFloat(-1, 3.1415);   //
Client[v[1]].Destiny.SendSwitch(1, true);     //
Client[v[1]].Destiny.SendString(2, "Hallo");  //

// Fechar a conexão (isso faz o exemplo do servidor
Client[v[1]].Close();
```

## Erros possíveis

`DESTINYERROR_SOCKETTYPE`, `DESTINYERROR_SOCKETERROR`, `DESTINYERROR_OOB`

## Veja também

`Client`, `Client[]`, `Client[].Destiny`, `Server`, `Server.Listen()`, `Server.Accept()`, `Server.Close()`, `Client[].Connect()`, `Client[].Close()`, `Client[].Destiny.SendVariable()`, `Client[].Destiny.SendDword()`, `Client[].Destiny.SendWord()`, `Client[].Destiny.SendByte()`, `Client[].Destiny.SendFloat()`, `Client[].Destiny.SendString()`, `Client[].Destiny.SendSwitch()`, `Client[].Destiny.GetRecvType()`, `Client[].Destiny.RecvId()`, `Client[].Destiny.RecvVariable()`, `Client[].Destiny.RecvDword()`, `Client[].Destiny.RecvWord()`, `Client[].Destiny.RecvByte()`, `Client[].Destiny.RecvFloat()`, `Client[].Destiny.RecvString()`

MÉTODO

# Método: Client[].Destiny.RecvVariable()

## Sintaxe

```
Client[Cliente-Índice].Destiny.RecvVariable()
```

## Descrição

Com o método RecvVariable é possível obter o valor de uma variável recebida. Ao chamar este método, a variável recebida é automaticamente removida dos dados recebidos do socket cliente, de modo que o valor dessa variável, bem como o Id associado, não podem mais ser consultados depois. Por isso, o Id precisa ser consultado antes do valor da variável. A chamada deste método só é bem-sucedida quando `GetRecvType()` tem o valor de retorno `TYPE_VARIABLE`. Depois que a variável for recebida, os valores eventualmente enviados em seguida podem ser recebidos.

## Lista de parâmetros

### Valor de retorno

- **Tipo de dado:** `dword`
- **Descrição:** O valor da variável recebida. Esse é o valor que foi informado em `SendVariable()`.

## Exemplo

> **Informação:** Para testar os exemplos a seguir localmente, recomenda-se iniciar o jogo duas vezes: uma como servidor e outra como cliente. O servidor precisa ser iniciado primeiro, pois caso contrário o cliente não consegue se conectar. Como o exemplo se conecta ao próprio computador (o IP 127.0.0.1 sempre representa o próprio computador), este exemplo deve funcionar mesmo atrás de roteadores/firewalls. Com o uso do comando pause, o DestinyScript dentro do laço é brevemente interrompido, para que o RPG_RT continue funcionando e o jogo não trave.

DestinyScript do servidor (recebe os dados aqui; precisa ser iniciado primeiro):

```
$

// ------------------------------------------------
// PARTE 1: iniciar o servidor brevemente e aceitar a conexã
// ------------------------------------------------

// Aguardar conexões de entrada na porta 12000
Server.Listen(12000, SOCK_DESTINY);

do
  v[1] = Server.Accept(Client.NextFree());
  pause;
loop until(v[1] != -1);

// A conexão foi aceita
// -> fechar o servidor (a conexão já aceita se
Server.Close();

// ------------------------------------------------
// PARTE 2: receber os dados (assim que disponíveis)
// ------------------------------------------------

// Aguardar dados de entrada
Destiny.Debugger.Output.Clear();
do
  v[2] = Client[v[1]].Destiny.GetRecvType();
  if(v[2] > 0)
    // Os dados estão disponíveis e prontos para serem recebidos
    // -> Primeiro consultar o Id
    v[3] = Client[v[1]].Destiny.RecvId();
    // -> Depois consultar o valor (dependendo do tipo)
```

```
    switch(v[2])
      case TYPE_VARIABLE:
        Destiny.Debugger.Output.Print("Variable e
        break;
      case TYPE_DWORD:
        Destiny.Debugger.Output.Print("Dword empf
        break;
      case TYPE_WORD:
        Destiny.Debugger.Output.Print("Word empfa
        break;
      case TYPE_BYTE:
        Destiny.Debugger.Output.Print("Byte empfa
        break;
      case TYPE_SWITCH:
        Destiny.Debugger.Output.Print("Switch emp
        break;
      case TYPE_FLOAT:
        Destiny.Debugger.Output.Print("Float empf
        break;
      case TYPE_STRING:
        Destiny.Debugger.Output.Print("String emp
        break;
    endswitch;
  endif;
  pause;
loop while(v[2] > 0 || Client[v[1]].State == STAT

// Ao final do script, ainda fechar a conexão
Client[v[1]].Close();

// Exibir a saída de depuração
Destiny.Debugger.Output.Show();
```

DestinyScript do cliente (envia os dados aqui; o servidor já precisa estar rodando):

```
$

// ------------------------------------------
// PARTE 1: estabelecer conexão com o servidor (aqui
// ------------------------------------------
v[1] = Client.NextFree();
Client[v[1]].Connect("127.0.0.1", 12000, SOCK_DES

// ------------------------------------------
// PARTE 2: enviar os dados
// ------------------------------------------
Client[v[1]].Destiny.SendVariable(1, 1234);   //
d[1] = 4321;
Client[v[1]].Destiny.SendDword(2, d[1]);      //
Client[v[1]].Destiny.SendWord(13, -11);       //
Client[v[1]].Destiny.SendByte(0, 255);        //
Client[v[1]].Destiny.SendFloat(-1, 3.1415);   //
Client[v[1]].Destiny.SendSwitch(1, true);     //
Client[v[1]].Destiny.SendString(2, "Hallo");  //

// Fechar a conexão (isso faz o exemplo do servidor
Client[v[1]].Close();
```

## Erros possíveis

`DESTINYERROR_SOCKETTYPE`, `DESTINYERROR_SOCKETERROR`, `DESTINYERROR_OOB`

## Veja também

`Client`, `Client[]`, `Client[].Destiny`, `Server`, `Server.Listen()`, `Server.Accept()`, `Server.Close()`, `Client[].Connect()`, `Client[].Close()`, `Client[].Destiny.SendVariable()`, `Client[].Destiny.SendDword()`, `Client[].Destiny.SendWord()`, `Client[].Destiny.SendByte()`, `Client[].Destiny.SendFloat()`, `Client[].Destiny.SendString()`, `Client[].Destiny.SendSwitch()`, `Client[].Destiny.GetRecvType()`, `Client[].Destiny.RecvId()`, `Client[].Destiny.RecvDword()`, `Client[].Destiny.RecvWord()`, `Client[].Destiny.RecvByte()`, `Client[].Destiny.RecvFloat()`, `Client[].Destiny.RecvString()`, `Client[].Destiny.RecvSwitch()`

MÉTODO

# Método: Client[].Destiny.RecvWord()

## Sintaxe

```
Client[Cliente-Índice].Destiny.RecvWord()
```

## Descrição

Com o método RecvWord é possível obter o valor de um Word recebido. Ao chamar este método, o Word recebido é automaticamente removido dos dados recebidos do socket cliente, de modo que o valor desse Word, bem como o Id associado, não podem mais ser consultados depois. Por isso, o Id precisa ser consultado antes do valor do Word. A chamada deste método só é bem-sucedida quando `GetRecvType()` tem o valor de retorno `TYPE_WORD`. Depois que o Word for recebido, os valores eventualmente enviados em seguida podem ser recebidos.

## Lista de parâmetros

### Valor de retorno

- **Tipo de dado:** `word`
- **Descrição:** O valor do Word recebido. Esse é o valor que foi informado em `SendWord()`.

## Exemplo

> **Informação:** Para testar os exemplos a seguir localmente, recomenda-se iniciar o jogo duas vezes: uma como servidor e outra como cliente. O servidor precisa ser iniciado primeiro, pois caso contrário o cliente não consegue se conectar. Como o exemplo se conecta ao próprio computador (o IP 127.0.0.1 sempre representa o próprio computador), este exemplo deve funcionar mesmo atrás de roteadores/firewalls. Com o uso do comando pause, o DestinyScript dentro do laço é brevemente interrompido, para que o RPG_RT continue funcionando e o jogo não trave.

DestinyScript do servidor (recebe os dados aqui; precisa ser iniciado primeiro):

```
$

// ------------------------------------------------
// PARTE 1: iniciar o servidor brevemente e aceitar a conexã
// ------------------------------------------------

// Aguardar conexões de entrada na porta 12000
Server.Listen(12000, SOCK_DESTINY);
```

```
do
  v[1] = Server.Accept(Client.NextFree());
  pause;
loop until(v[1] != -1);

// A conexão foi aceita
// -> fechar o servidor (a conexão já aceita se
Server.Close();

// ---------------------------------------------
// PARTE 2: receber os dados (assim que disponíveis)
// ---------------------------------------------

// Aguardar dados de entrada
Destiny.Debugger.Output.Clear();
do
  v[2] = Client[v[1]].Destiny.GetRecvType();
  if(v[2] > 0)
    // Os dados estão disponíveis e prontos para serem recebidos
    // -> Primeiro consultar o Id
    v[3] = Client[v[1]].Destiny.RecvId();
    // -> Depois consultar o valor (dependendo do tipo)
    switch(v[2])
      case TYPE_VARIABLE:
        Destiny.Debugger.Output.Print("Variable e
        break;
      case TYPE_DWORD:
        Destiny.Debugger.Output.Print("Dword empf
        break;
      case TYPE_WORD:
        Destiny.Debugger.Output.Print("Word empfa
        break;
      case TYPE_BYTE:
        Destiny.Debugger.Output.Print("Byte empfa
        break;
      case TYPE_SWITCH:
        Destiny.Debugger.Output.Print("Switch emp
        break;
      case TYPE_FLOAT:
        Destiny.Debugger.Output.Print("Float empf
        break;
      case TYPE_STRING:
        Destiny.Debugger.Output.Print("String emp
        break;
    endswitch;
  endif;
  pause;
loop while(v[2] > 0 || Client[v[1]].State == STAT

// Ao final do script, ainda fechar a conexão
Client[v[1]].Close();

// Exibir a saída de depuração
Destiny.Debugger.Output.Show();
```

DestinyScript do cliente (envia os dados aqui; o servidor já precisa estar rodando):

```
$

// ---------------------------------------------
// PARTE 1: estabelecer conexão com o servidor (aqui
// ---------------------------------------------
v[1] = Client.NextFree();
Client[v[1]].Connect("127.0.0.1", 12000, SOCK_DES
```

```
// ---------------------------------------------
// PARTE 2: enviar os dados
// ---------------------------------------------
Client[v[1]].Destiny.SendVariable(1, 1234);   //
d[1] = 4321;
Client[v[1]].Destiny.SendDword(2, d[1]);      //
Client[v[1]].Destiny.SendWord(13, -11);       //
Client[v[1]].Destiny.SendByte(0, 255);        //
Client[v[1]].Destiny.SendFloat(-1, 3.1415);   //
Client[v[1]].Destiny.SendSwitch(1, true);     //
Client[v[1]].Destiny.SendString(2, "Hallo");  //

// Fechar a conexão (isso faz o exemplo do servidor
Client[v[1]].Close();
```

## Erros possíveis

`DESTINYERROR_SOCKETTYPE`, `DESTINYERROR_SOCKETERROR`, `DESTINYERROR_OOB`

## Veja também

`Client`, `Client[]`, `Client[].Destiny`, `Server`, `Server.Listen()`, `Server.Accept()`, `Server.Close()`, `Client[].Connect()`, `Client[].Close()`, `Client[].Destiny.SendVariable()`, `Client[].Destiny.SendDword()`, `Client[].Destiny.SendWord()`, `Client[].Destiny.SendByte()`, `Client[].Destiny.SendFloat()`, `Client[].Destiny.SendString()`, `Client[].Destiny.SendSwitch()`, `Client[].Destiny.GetRecvType()`, `Client[].Destiny.RecvId()`, `Client[].Destiny.RecvVariable()`, `Client[].Destiny.RecvDword()`, `Client[].Destiny.RecvByte()`, `Client[].Destiny.RecvFloat()`, `Client[].Destiny.RecvString()`, `Client[].Destiny.RecvSwitch()`

MÉTODO

# Método: Client[].Destiny.SendByte()

## Sintaxe

```
Client[Cliente-Índice].Destiny.SendByte(Id, Valor)
```

## Descrição

Com o método SendByte é possível enviar um Byte por um socket cliente Destiny conectado.

## Lista de parâmetros

### Parâmetro: `Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** O Id, que é transmitido junto com o Byte. O significado desse parâmetro é definido pelo criador do DestinyScript. Não é necessário que esse Id coincida com o Id real do Byte (por exemplo, com o índice em `v[]`), podendo em vez disso ser usado, por exemplo, como identificador de uma ação a ser executada.

### Parâmetro: `Valor`

- **Tipo de dado:** `byte`

- **Opcional:** não
- **Descrição:** O valor a ser usado. Esse valor não precisa ser informado diretamente (por exemplo, 123); pode vir de um valor de memória qualquer (compatível com byte).

## Exemplo

> **Informação:** Para testar os exemplos a seguir localmente, recomenda-se iniciar o jogo duas vezes: uma como servidor e outra como cliente. O servidor precisa ser iniciado primeiro, pois caso contrário o cliente não consegue se conectar. Como o exemplo se conecta ao próprio computador (o IP 127.0.0.1 sempre representa o próprio computador), este exemplo deve funcionar mesmo atrás de roteadores/firewalls. Com o uso do comando pause, o DestinyScript dentro do laço é brevemente interrompido, para que o RPG_RT continue funcionando e o jogo não trave.

DestinyScript do servidor (recebe os dados aqui; precisa ser iniciado primeiro):

```
$

// ---------------------------------------------
// PARTE 1: iniciar o servidor brevemente e aceitar a conexã
// ---------------------------------------------

// Aguardar conexões de entrada na porta 12000
Server.Listen(12000, SOCK_DESTINY);

do
  v[1] = Server.Accept(Client.NextFree());
  pause;
loop until(v[1] != -1);

// A conexão foi aceita
// -> fechar o servidor (a conexão já aceita se
Server.Close();

// ---------------------------------------------
// PARTE 2: receber os dados (assim que disponíveis)
// ---------------------------------------------

// Aguardar dados de entrada
Destiny.Debugger.Output.Clear();
do
  v[2] = Client[v[1]].Destiny.GetRecvType();
  if(v[2] > 0)
    // Os dados estão disponíveis e prontos para serem recebidos
    // -> Primeiro consultar o Id
    v[3] = Client[v[1]].Destiny.RecvId();
    // -> Depois consultar o valor (dependendo do tipo)
    switch(v[2])
      case TYPE_VARIABLE:
        Destiny.Debugger.Output.Print("Variable e
        break;
      case TYPE_DWORD:
        Destiny.Debugger.Output.Print("Dword empf
        break;
      case TYPE_WORD:
        Destiny.Debugger.Output.Print("Word empfa
        break;
      case TYPE_BYTE:
        Destiny.Debugger.Output.Print("Byte empfa
        break;
      case TYPE_SWITCH:
        Destiny.Debugger.Output.Print("Switch emp
```

```
        break;
      case TYPE_FLOAT:
        Destiny.Debugger.Output.Print("Float empf
        break;
      case TYPE_STRING:
        Destiny.Debugger.Output.Print("String emp
        break;
    endswitch;
  endif;
  pause;
loop while(v[2] > 0 || Client[v[1]].State == STAT

// Ao final do script, ainda fechar a conexão
Client[v[1]].Close();

// Exibir a saída de depuração
Destiny.Debugger.Output.Show();
```

DestinyScript do cliente (envia os dados aqui; o servidor já precisa estar rodando):

```
$

// ------------------------------------------
// PARTE 1: estabelecer conexão com o servidor (aqui
// ------------------------------------------
v[1] = Client.NextFree();
Client[v[1]].Connect("127.0.0.1", 12000, SOCK_DES

// ------------------------------------------
// PARTE 2: enviar os dados
// ------------------------------------------
Client[v[1]].Destiny.SendVariable(1, 1234);   //
d[1] = 4321;
Client[v[1]].Destiny.SendDword(2, d[1]);      //
Client[v[1]].Destiny.SendWord(13, -11);       //
Client[v[1]].Destiny.SendByte(0, 255);        //
Client[v[1]].Destiny.SendFloat(-1, 3.1415);   //
Client[v[1]].Destiny.SendSwitch(1, true);     //
Client[v[1]].Destiny.SendString(2, "Hallo");  //

// Fechar a conexão (isso faz o exemplo do servidor
Client[v[1]].Close();
```

## Erros possíveis

`DESTINYERROR_SOCKETNOTOPEN`, `DESTINYERROR_SOCKETTYPE`

## Veja também

`Client`, `Client[]`, `Client[].Destiny`, `Server`, `Server.Listen()`, `Server.Accept()`, `Server.Close()`, `Client[].Connect()`, `Client[].Close()`, `Client[].Destiny.SendVariable()`, `Client[].Destiny.SendDword()`, `Client[].Destiny.SendWord()`, `Client[].Destiny.SendFloat()`, `Client[].Destiny.SendString()`, `Client[].Destiny.SendSwitch()`, `Client[].Destiny.GetRecvType()`, `Client[].Destiny.RecvId()`, `Client[].Destiny.RecvVariable()`, `Client[].Destiny.RecvDword()`, `Client[].Destiny.RecvWord()`, `Client[].Destiny.RecvByte()`, `Client[].Destiny.RecvFloat()`, `Client[].Destiny.RecvString()`, `Client[].Destiny.RecvSwitch()`

MÉTODO

# Método: Client[].Destiny.SendDword()

## Sintaxe

```
Client[Cliente-Índice].Destiny.SendDword(Id, Valor)
```

## Descrição

Com o método SendDword é possível enviar um Dword por um socket cliente Destiny conectado.

> **Informação:** Como o Id usado aqui não precisa coincidir com o Id real do Dword, esse Id pode ser usado, por exemplo, para identificar uma ação a ser executada.

## Lista de parâmetros

### Parâmetro: `Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** O Id, que é transmitido junto com o Dword. Esse Id não precisa coincidir com o Id real do Dword (por exemplo, com o índice em `d[]`).

### Parâmetro: `Valor`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** O valor a ser usado. Esse valor não precisa vir de um Dword (`d[]`); pode vir de um valor de memória qualquer (compatível com dword).

## Exemplo

> **Informação:** Para testar os exemplos a seguir localmente, recomenda-se iniciar o jogo duas vezes: uma como servidor e outra como cliente. O servidor precisa ser iniciado primeiro, pois caso contrário o cliente não consegue se conectar. Como o exemplo se conecta ao próprio computador (o IP 127.0.0.1 sempre representa o próprio computador), este exemplo deve funcionar mesmo atrás de roteadores/firewalls. Com o uso do comando pause, o DestinyScript dentro do laço é brevemente interrompido, para que o RPG_RT continue funcionando e o jogo não trave.

DestinyScript do servidor (recebe os dados aqui; precisa ser iniciado primeiro):

```
$

// ------------------------------------------
// PARTE 1: iniciar o servidor brevemente e aceitar a conexã
// ------------------------------------------

// Aguardar conexões de entrada na porta 12000
Server.Listen(12000, SOCK_DESTINY);

do
  v[1] = Server.Accept(Client.NextFree());
```

```
  pause;
loop until(v[1] != -1);

// A conexão foi aceita
// -> fechar o servidor (a conexão já aceita se
Server.Close();

// ------------------------------------------
// PARTE 2: receber os dados (assim que disponíveis)
// ------------------------------------------

// Aguardar dados de entrada
Destiny.Debugger.Output.Clear();
do
  v[2] = Client[v[1]].Destiny.GetRecvType();
  if(v[2] > 0)
    // Os dados estão disponíveis e prontos para serem recebidos
    // -> Primeiro consultar o Id
    v[3] = Client[v[1]].Destiny.RecvId();
    // -> Depois consultar o valor (dependendo do tipo)
    switch(v[2])
      case TYPE_VARIABLE:
        Destiny.Debugger.Output.Print("Variable e
        break;
      case TYPE_DWORD:
        Destiny.Debugger.Output.Print("Dword empf
        break;
      case TYPE_WORD:
        Destiny.Debugger.Output.Print("Word empfa
        break;
      case TYPE_BYTE:
        Destiny.Debugger.Output.Print("Byte empfa
        break;
      case TYPE_SWITCH:
        Destiny.Debugger.Output.Print("Switch emp
        break;
      case TYPE_FLOAT:
        Destiny.Debugger.Output.Print("Float empf
        break;
      case TYPE_STRING:
        Destiny.Debugger.Output.Print("String emp
        break;
    endswitch;
  endif;
  pause;
loop while(v[2] > 0 || Client[v[1]].State == STAT

// Ao final do script, ainda fechar a conexão
Client[v[1]].Close();

// Exibir a saída de depuração
Destiny.Debugger.Output.Show();
```

DestinyScript do cliente (envia os dados aqui; o servidor já precisa estar rodando):

```
$

// ------------------------------------------
// PARTE 1: estabelecer conexão com o servidor (aqui
// ------------------------------------------
v[1] = Client.NextFree();
Client[v[1]].Connect("127.0.0.1", 12000, SOCK_DES

// ------------------------------------------
// PARTE 2: enviar os dados
```

```
// ---------------------------------------------
Client[v[1]].Destiny.SendVariable(1, 1234);   //
d[1] = 4321;
Client[v[1]].Destiny.SendDword(2, d[1]);      //
Client[v[1]].Destiny.SendWord(13, -11);       //
Client[v[1]].Destiny.SendByte(0, 255);        //
Client[v[1]].Destiny.SendFloat(-1, 3.1415);   //
Client[v[1]].Destiny.SendSwitch(1, true);     //
Client[v[1]].Destiny.SendString(2, "Hallo");  //

// Fechar a conexão (isso faz o exemplo do servidor
Client[v[1]].Close();
```

## Erros possíveis

`DESTINYERROR_SOCKETNOTOPEN`, `DESTINYERROR_SOCKETTYPE`

## Veja também

`Client`, `Client[]`, `Client[].Destiny`, `Server`, `Server.Listen()`, `Server.Accept()`, `Server.Close()`, `Client[].Connect()`, `Client[].Close()`, `Client[].Destiny.SendVariable()`, `Client[].Destiny.SendWord()`, `Client[].Destiny.SendByte()`, `Client[].Destiny.SendFloat()`, `Client[].Destiny.SendString()`, `Client[].Destiny.SendSwitch()`, `Client[].Destiny.GetRecvType()`, `Client[].Destiny.RecvId()`, `Client[].Destiny.RecvVariable()`, `Client[].Destiny.RecvDword()`, `Client[].Destiny.RecvWord()`, `Client[].Destiny.RecvByte()`, `Client[].Destiny.RecvFloat()`, `Client[].Destiny.RecvString()`, `Client[].Destiny.RecvSwitch()`

MÉTODO

# Método: Client[].Destiny.SendFloat()

## Sintaxe

```
Client[Cliente-Índice].Destiny.SendFloat(Id, Valor)
```

## Descrição

Com o método SendFloat é possível enviar um Float por um socket cliente Destiny conectado.

> **Informação:** Como o Id usado aqui não precisa coincidir com o Id real do Float, esse Id pode ser usado, por exemplo, para identificar uma ação a ser executada.

## Sinônimos

SendDouble()

## Lista de parâmetros

### Parâmetro: `Id`

- **Tipo de dado:** `dword`
- **Opcional:** não

- **Descrição:** O Id, que é transmitido junto com o Float. Esse Id não precisa coincidir com o Id real do Float (por exemplo, com o índice em `f[]`).

### Parâmetro: `Valor`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** O valor a ser usado. Esse valor não precisa vir de um Float (`f[]`); pode vir de um valor de memória qualquer (compatível com float).

## Exemplo

> **Informação:** Para testar os exemplos a seguir localmente, recomenda-se iniciar o jogo duas vezes: uma como servidor e outra como cliente. O servidor precisa ser iniciado primeiro, pois caso contrário o cliente não consegue se conectar. Como o exemplo se conecta ao próprio computador (o IP 127.0.0.1 sempre representa o próprio computador), este exemplo deve funcionar mesmo atrás de roteadores/firewalls. Com o uso do comando pause, o DestinyScript dentro do laço é brevemente interrompido, para que o RPG_RT continue funcionando e o jogo não trave.

DestinyScript do servidor (recebe os dados aqui; precisa ser iniciado primeiro):

```
$

// ---------------------------------------------
// PARTE 1: iniciar o servidor brevemente e aceitar a conexã
// ---------------------------------------------

// Aguardar conexões de entrada na porta 12000
Server.Listen(12000, SOCK_DESTINY);

do
  v[1] = Server.Accept(Client.NextFree());
  pause;
loop until(v[1] != -1);

// A conexão foi aceita
// -> fechar o servidor (a conexão já aceita se
Server.Close();

// ---------------------------------------------
// PARTE 2: receber os dados (assim que disponíveis)
// ---------------------------------------------

// Aguardar dados de entrada
Destiny.Debugger.Output.Clear();
do
  v[2] = Client[v[1]].Destiny.GetRecvType();
  if(v[2] > 0)
    // Os dados estão disponíveis e prontos para serem recebidos
    // -> Primeiro consultar o Id
    v[3] = Client[v[1]].Destiny.RecvId();
    // -> Depois consultar o valor (dependendo do tipo)
    switch(v[2])
      case TYPE_VARIABLE:
        Destiny.Debugger.Output.Print("Variable e
        break;
      case TYPE_DWORD:
        Destiny.Debugger.Output.Print("Dword empf
        break;
      case TYPE_WORD:
```

```
        Destiny.Debugger.Output.Print("Word empfa
        break;
      case TYPE_BYTE:
        Destiny.Debugger.Output.Print("Byte empfa
        break;
      case TYPE_SWITCH:
        Destiny.Debugger.Output.Print("Switch emp
        break;
      case TYPE_FLOAT:
        Destiny.Debugger.Output.Print("Float empf
        break;
      case TYPE_STRING:
        Destiny.Debugger.Output.Print("String emp
        break;
    endswitch;
  endif;
  pause;
loop while(v[2] > 0 || Client[v[1]].State == STAT

// Ao final do script, ainda fechar a conexão
Client[v[1]].Close();

// Exibir a saída de depuração
Destiny.Debugger.Output.Show();
```

DestinyScript do cliente (envia os dados aqui; o servidor já precisa estar rodando):

```
$

// ------------------------------------------
// PARTE 1: estabelecer conexão com o servidor (aqui
// ------------------------------------------
v[1] = Client.NextFree();
Client[v[1]].Connect("127.0.0.1", 12000, SOCK_DES

// ------------------------------------------
// PARTE 2: enviar os dados
// ------------------------------------------
Client[v[1]].Destiny.SendVariable(1, 1234);   //
d[1] = 4321;
Client[v[1]].Destiny.SendDword(2, d[1]);      //
Client[v[1]].Destiny.SendWord(13, -11);       //
Client[v[1]].Destiny.SendByte(0, 255);        //
Client[v[1]].Destiny.SendFloat(-1, 3.1415);   //
Client[v[1]].Destiny.SendSwitch(1, true);     //
Client[v[1]].Destiny.SendString(2, "Hallo");  //

// Fechar a conexão (isso faz o exemplo do servidor
Client[v[1]].Close();
```

## Erros possíveis

`DESTINYERROR_SOCKETNOTOPEN`, `DESTINYERROR_SOCKETTYPE`

## Veja também

`Client`, `Client[]`, `Client[].Destiny`, `Server`, `Server.Listen()`, `Server.Accept()`, `Server.Close()`, `Client[].Connect()`, `Client[].Close()`, `Client[].Destiny.SendVariable()`, `Client[].Destiny.SendDword()`, `Client[].Destiny.SendWord()`, `Client[].Destiny.SendByte()`, `Client[].Destiny.SendString()`, `Client[].Destiny.SendSwitch()`, `Client[].Destiny.GetRecvType()`, `Client[].Destiny.RecvId()`, `Client[].Destiny.RecvVariable()`, `Client[].Destiny.RecvDword()`,

`Client[].Destiny.RecvWord()`, `Client[].Destiny.RecvByte()`, `Client[].Destiny.RecvFloat()`, `Client[].Destiny.RecvString()`, `Client[].Destiny.RecvSwitch()`

MÉTODO

# Método: Client[].Destiny.SendString()

## Sintaxe

```
Client[Cliente-Índice].Destiny.SendString(Id, Valor)
```

## Descrição

Com o método SendString é possível enviar uma String por um socket cliente Destiny conectado.

> **Informação:** Como o Id usado aqui não precisa coincidir com o Id real da String, esse Id pode ser usado, por exemplo, para identificar uma ação a ser executada.

> **Informação:** Por limitações técnicas, a String a ser enviada não pode ter mais de 255 caracteres.

## Lista de parâmetros

### Parâmetro: `Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** O Id, que é transmitido junto com a String. Esse Id não precisa coincidir com o Id real da String (por exemplo, com o índice em `a[]`).

### Parâmetro: `Valor`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** O valor a ser usado. Esse valor não precisa vir de uma String (`a[]`); pode vir de um valor de memória qualquer (compatível com string).

Atenção: por limitações técnicas, a String não pode ter mais de 255 caracteres.

## Exemplo

> **Informação:** Para testar os exemplos a seguir localmente, recomenda-se iniciar o jogo duas vezes: uma como servidor e outra como cliente. O servidor precisa ser iniciado primeiro, pois caso contrário o cliente não consegue se conectar. Como o exemplo se conecta ao próprio computador (o IP 127.0.0.1 sempre representa o próprio computador), este exemplo deve funcionar mesmo atrás de roteadores/firewalls. Com o uso do comando pause, o DestinyScript dentro do laço é brevemente interrompido, para que o RPG_RT continue funcionando e o jogo não trave.

DestinyScript do servidor (recebe os dados aqui; precisa ser iniciado primeiro):

```
$
```

```
// ---------------------------------------------
// PARTE 1: iniciar o servidor brevemente e aceitar a conexã
// ---------------------------------------------

// Aguardar conexões de entrada na porta 12000
Server.Listen(12000, SOCK_DESTINY);

do
  v[1] = Server.Accept(Client.NextFree());
  pause;
loop until(v[1] != -1);

// A conexão foi aceita
// -> fechar o servidor (a conexão já aceita se
Server.Close();

// ---------------------------------------------
// PARTE 2: receber os dados (assim que disponíveis)
// ---------------------------------------------

// Aguardar dados de entrada
Destiny.Debugger.Output.Clear();
do
  v[2] = Client[v[1]].Destiny.GetRecvType();
  if(v[2] > 0)
    // Os dados estão disponíveis e prontos para serem recebidos
    // -> Primeiro consultar o Id
    v[3] = Client[v[1]].Destiny.RecvId();
    // -> Depois consultar o valor (dependendo do tipo)
    switch(v[2])
      case TYPE_VARIABLE:
        Destiny.Debugger.Output.Print("Variable e
        break;
      case TYPE_DWORD:
        Destiny.Debugger.Output.Print("Dword empf
        break;
      case TYPE_WORD:
        Destiny.Debugger.Output.Print("Word empfa
        break;
      case TYPE_BYTE:
        Destiny.Debugger.Output.Print("Byte empfa
        break;
      case TYPE_SWITCH:
        Destiny.Debugger.Output.Print("Switch emp
        break;
      case TYPE_FLOAT:
        Destiny.Debugger.Output.Print("Float empf
        break;
      case TYPE_STRING:
        Destiny.Debugger.Output.Print("String emp
        break;
    endswitch;
  endif;
  pause;
loop while(v[2] > 0 || Client[v[1]].State == STAT

// Ao final do script, ainda fechar a conexão
Client[v[1]].Close();

// Exibir a saída de depuração
Destiny.Debugger.Output.Show();
```

DestinyScript do cliente (envia os dados aqui; o servidor já precisa estar rodando):

```
$

// ------------------------------------------
// PARTE 1: estabelecer conexão com o servidor (aqui
// ------------------------------------------
v[1] = Client.NextFree();
Client[v[1]].Connect("127.0.0.1", 12000, SOCK_DES

// ------------------------------------------
// PARTE 2: enviar os dados
// ------------------------------------------
Client[v[1]].Destiny.SendVariable(1, 1234);   //
d[1] = 4321;
Client[v[1]].Destiny.SendDword(2, d[1]);      //
Client[v[1]].Destiny.SendWord(13, -11);       //
Client[v[1]].Destiny.SendByte(0, 255);        //
Client[v[1]].Destiny.SendFloat(-1, 3.1415);   //
Client[v[1]].Destiny.SendSwitch(1, true);     //
Client[v[1]].Destiny.SendString(2, "Hallo");  //

// Fechar a conexão (isso faz o exemplo do servidor
Client[v[1]].Close();
```

## Erros possíveis

`DESTINYERROR_SOCKETNOTOPEN`, `DESTINYERROR_SOCKETTYPE`, `DESTINYERROR_STRINGTOOLONG`

## Veja também

`Client`, `Client[]`, `Client[].Destiny`, `Server`, `Server.Listen()`, `Server.Accept()`, `Server.Close()`, `Client[].Connect()`, `Client[].Close()`, `Client[].Destiny.SendVariable()`, `Client[].Destiny.SendDword()`, `Client[].Destiny.SendWord()`, `Client[].Destiny.SendByte()`, `Client[].Destiny.SendFloat()`, `Client[].Destiny.SendSwitch()`, `Client[].Destiny.GetRecvType()`, `Client[].Destiny.RecvId()`, `Client[].Destiny.RecvVariable()`, `Client[].Destiny.RecvDword()`, `Client[].Destiny.RecvWord()`, `Client[].Destiny.RecvByte()`, `Client[].Destiny.RecvFloat()`, `Client[].Destiny.RecvString()`, `Client[].Destiny.RecvSwitch()`

MÉTODO

# Método: Client[].Destiny.SendSwitch()

## Sintaxe

```
Client[Cliente-Índice].Destiny.SendSwitch(Id, Valor)
```

## Descrição

Com o método SendSwitch é possível enviar um switch por um socket cliente Destiny conectado.

> **Informação:** Como o Id usado aqui não precisa coincidir com o Id real do switch, esse Id pode ser usado, por exemplo, para identificar uma ação a ser executada.

## Sinônimos

SendBool()

## Lista de parâmetros

### Parâmetro: `Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** O Id, que é transmitido junto com o switch. Esse Id não precisa coincidir com o Id real do switch (por exemplo, com o índice em `s[]`).

### Parâmetro: `Valor`

- **Tipo de dado:** `bool`
- **Opcional:** não
- **Descrição:** O valor a ser usado. Esse valor não precisa vir de um switch (`s[]`); pode vir de um valor de memória qualquer (compatível com bool).

## Exemplo

> **Informação:** Para testar os exemplos a seguir localmente, recomenda-se iniciar o jogo duas vezes: uma como servidor e outra como cliente. O servidor precisa ser iniciado primeiro, pois caso contrário o cliente não consegue se conectar. Como o exemplo se conecta ao próprio computador (o IP 127.0.0.1 sempre representa o próprio computador), este exemplo deve funcionar mesmo atrás de roteadores/firewalls. Com o uso do comando pause, o DestinyScript dentro do laço é brevemente interrompido, para que o RPG_RT continue funcionando e o jogo não trave.

DestinyScript do servidor (recebe os dados aqui; precisa ser iniciado primeiro):

```
$

// ------------------------------------------
// PARTE 1: iniciar o servidor brevemente e aceitar a conexã
// ------------------------------------------

// Aguardar conexões de entrada na porta 12000
Server.Listen(12000, SOCK_DESTINY);

do
  v[1] = Server.Accept(Client.NextFree());
  pause;
loop until(v[1] != -1);

// A conexão foi aceita
// -> fechar o servidor (a conexão já aceita se
Server.Close();

// ------------------------------------------
// PARTE 2: receber os dados (assim que disponíveis)
// ------------------------------------------

// Aguardar dados de entrada
Destiny.Debugger.Output.Clear();
do
  v[2] = Client[v[1]].Destiny.GetRecvType();
```

```
  if(v[2] > 0)
    // Os dados estão disponíveis e prontos para serem recebidos
    // -> Primeiro consultar o Id
    v[3] = Client[v[1]].Destiny.RecvId();
    // -> Depois consultar o valor (dependendo do tipo)
    switch(v[2])
      case TYPE_VARIABLE:
        Destiny.Debugger.Output.Print("Variable e
        break;
      case TYPE_DWORD:
        Destiny.Debugger.Output.Print("Dword empf
        break;
      case TYPE_WORD:
        Destiny.Debugger.Output.Print("Word empfa
        break;
      case TYPE_BYTE:
        Destiny.Debugger.Output.Print("Byte empfa
        break;
      case TYPE_SWITCH:
        Destiny.Debugger.Output.Print("Switch emp
        break;
      case TYPE_FLOAT:
        Destiny.Debugger.Output.Print("Float empf
        break;
      case TYPE_STRING:
        Destiny.Debugger.Output.Print("String emp
        break;
    endswitch;
  endif;
  pause;
loop while(v[2] > 0 || Client[v[1]].State == STAT

// Ao final do script, ainda fechar a conexão
Client[v[1]].Close();

// Exibir a saída de depuração
Destiny.Debugger.Output.Show();
```

DestinyScript do cliente (envia os dados aqui; o servidor já precisa estar rodando):

```
$

// ------------------------------------------
// PARTE 1: estabelecer conexão com o servidor (aqui
// ------------------------------------------
v[1] = Client.NextFree();
Client[v[1]].Connect("127.0.0.1", 12000, SOCK_DES

// ------------------------------------------
// PARTE 2: enviar os dados
// ------------------------------------------
Client[v[1]].Destiny.SendVariable(1, 1234);   //
d[1] = 4321;
Client[v[1]].Destiny.SendDword(2, d[1]);      //
Client[v[1]].Destiny.SendWord(13, -11);       //
Client[v[1]].Destiny.SendByte(0, 255);        //
Client[v[1]].Destiny.SendFloat(-1, 3.1415);   //
Client[v[1]].Destiny.SendSwitch(1, true);     //
Client[v[1]].Destiny.SendString(2, "Hallo");  //

// Fechar a conexão (isso faz o exemplo do servidor
Client[v[1]].Close();
```

## Erros possíveis

`DESTINYERROR_SOCKETNOTOPEN`, `DESTINYERROR_SOCKETTYPE`

## Veja também

`Client`, `Client[]`, `Client[].Destiny`, `Server`, `Server.Listen()`, `Server.Accept()`, `Server.Close()`, `Client[].Connect()`, `Client[].Close()`, `Client[].Destiny.SendVariable()`, `Client[].Destiny.SendDword()`, `Client[].Destiny.SendWord()`, `Client[].Destiny.SendByte()`, `Client[].Destiny.SendFloat()`, `Client[].Destiny.SendString()`, `Client[].Destiny.GetRecvType()`, `Client[].Destiny.RecvId()`, `Client[].Destiny.RecvVariable()`, `Client[].Destiny.RecvDword()`, `Client[].Destiny.RecvWord()`, `Client[].Destiny.RecvByte()`, `Client[].Destiny.RecvFloat()`, `Client[].Destiny.RecvString()`, `Client[].Destiny.RecvSwitch()`

MÉTODO

# Método: Client[].Destiny.SendVariable()

## Sintaxe

```
Client[Cliente-Índice].Destiny.SendVariable(Id, Valor)
```

## Descrição

Com o método SendVariable é possível enviar um variável por um socket cliente Destiny conectado.

> **Informação:** Como o Id usado aqui não precisa coincidir com o Id real do variável, esse Id pode ser usado, por exemplo, para identificar uma ação a ser executada.

## Lista de parâmetros

### Parâmetro: `Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** O Id, que é transmitido junto com o variável. Esse Id não precisa coincidir com o Id real do variável (por exemplo, com o índice em `v[]`).

### Parâmetro: `Valor`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** O valor a ser usado. Esse valor não precisa vir de um variável (`v[]`); pode vir de um valor de memória qualquer (compatível com dword).

## Exemplo

> **Informação:** Para testar os exemplos a seguir localmente, recomenda-se iniciar o jogo duas vezes: uma como servidor e outra como cliente. O servidor precisa ser iniciado primeiro, pois caso contrário o cliente não consegue se conectar. Como o exemplo se conecta ao próprio computador (o IP 127.0.0.1 sempre representa

> o próprio computador), este exemplo deve funcionar mesmo atrás de roteadores/firewalls. Com o uso do comando pause, o DestinyScript dentro do laço é brevemente interrompido, para que o RPG_RT continue funcionando e o jogo não trave.

DestinyScript do servidor (recebe os dados aqui; precisa ser iniciado primeiro):

```
$

// ------------------------------------------
// PARTE 1: iniciar o servidor brevemente e aceitar a conexã
// ------------------------------------------

// Aguardar conexões de entrada na porta 12000
Server.Listen(12000, SOCK_DESTINY);

do
  v[1] = Server.Accept(Client.NextFree());
  pause;
loop until(v[1] != -1);

// A conexão foi aceita
// -> fechar o servidor (a conexão já aceita se
Server.Close();

// ------------------------------------------
// PARTE 2: receber os dados (assim que disponíveis)
// ------------------------------------------

// Aguardar dados de entrada
Destiny.Debugger.Output.Clear();
do
  v[2] = Client[v[1]].Destiny.GetRecvType();
  if(v[2] > 0)
    // Os dados estão disponíveis e prontos para serem recebidos
    // -> Primeiro consultar o Id
    v[3] = Client[v[1]].Destiny.RecvId();
    // -> Depois consultar o valor (dependendo do tipo)
    switch(v[2])
      case TYPE_VARIABLE:
        Destiny.Debugger.Output.Print("Variable e
        break;
      case TYPE_DWORD:
        Destiny.Debugger.Output.Print("Dword empf
        break;
      case TYPE_WORD:
        Destiny.Debugger.Output.Print("Word empfa
        break;
      case TYPE_BYTE:
        Destiny.Debugger.Output.Print("Byte empfa
        break;
      case TYPE_SWITCH:
        Destiny.Debugger.Output.Print("Switch emp
        break;
      case TYPE_FLOAT:
        Destiny.Debugger.Output.Print("Float empf
        break;
      case TYPE_STRING:
        Destiny.Debugger.Output.Print("String emp
        break;
    endswitch;
  endif;
  pause;
loop while(v[2] > 0 || Client[v[1]].State == STAT

// Ao final do script, ainda fechar a conexão
```

```
Client[v[1]].Close();

// Exibir a saída de depuração
Destiny.Debugger.Output.Show();
```

DestinyScript do cliente (envia os dados aqui; o servidor já precisa estar rodando):

```
$

// ------------------------------------------
// PARTE 1: estabelecer conexão com o servidor (aqui
// ------------------------------------------
v[1] = Client.NextFree();
Client[v[1]].Connect("127.0.0.1", 12000, SOCK_DES

// ------------------------------------------
// PARTE 2: enviar os dados
// ------------------------------------------
Client[v[1]].Destiny.SendVariable(1, 1234);   //
d[1] = 4321;
Client[v[1]].Destiny.SendDword(2, d[1]);      //
Client[v[1]].Destiny.SendWord(13, -11);       //
Client[v[1]].Destiny.SendByte(0, 255);        //
Client[v[1]].Destiny.SendFloat(-1, 3.1415);   //
Client[v[1]].Destiny.SendSwitch(1, true);     //
Client[v[1]].Destiny.SendString(2, "Hallo");  //

// Fechar a conexão (isso faz o exemplo do servidor
Client[v[1]].Close();
```

## Erros possíveis

`DESTINYERROR_SOCKETNOTOPEN`, `DESTINYERROR_SOCKETTYPE`

## Veja também

`Client`, `Client[]`, `Client[].Destiny`, `Server`, `Server.Listen()`, `Server.Accept()`, `Server.Close()`, `Client[].Connect()`, `Client[].Close()`, `Client[].Destiny.SendDword()`, `Client[].Destiny.SendWord()`, `Client[].Destiny.SendByte()`, `Client[].Destiny.SendFloat()`, `Client[].Destiny.SendString()`, `Client[].Destiny.SendSwitch()`, `Client[].Destiny.GetRecvType()`, `Client[].Destiny.RecvId()`, `Client[].Destiny.RecvVariable()`, `Client[].Destiny.RecvDword()`, `Client[].Destiny.RecvWord()`, `Client[].Destiny.RecvByte()`, `Client[].Destiny.RecvFloat()`, `Client[].Destiny.RecvString()`, `Client[].Destiny.RecvSwitch()`

MÉTODO

# Método: Client[].Destiny.SendWord()

## Sintaxe

```
Client[Cliente-Índice].Destiny.SendWord(Id, Valor)
```

## Descrição

Com o método SendWord é possível enviar um Word por um socket cliente Destiny conectado.

## Lista de parâmetros

### Parâmetro: `Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** O Id, que é transmitido junto com o Word. O significado desse parâmetro é definido pelo criador do DestinyScript. Não é necessário que esse Id coincida com o Id real do Word (por exemplo, com o índice em `v[]`), podendo em vez disso ser usado, por exemplo, como identificador de uma ação a ser executada.

### Parâmetro: `Valor`

- **Tipo de dado:** `word`
- **Opcional:** não
- **Descrição:** O valor a ser usado. Esse valor não precisa ser informado diretamente (por exemplo, 123); pode vir de um valor de memória qualquer (compatível com word).

## Exemplo

> **Informação:** Para testar os exemplos a seguir localmente, recomenda-se iniciar o jogo duas vezes: uma como servidor e outra como cliente. O servidor precisa ser iniciado primeiro, pois caso contrário o cliente não consegue se conectar. Como o exemplo se conecta ao próprio computador (o IP 127.0.0.1 sempre representa o próprio computador), este exemplo deve funcionar mesmo atrás de roteadores/firewalls. Com o uso do comando pause, o DestinyScript dentro do laço é brevemente interrompido, para que o RPG_RT continue funcionando e o jogo não trave.

DestinyScript do servidor (recebe os dados aqui; precisa ser iniciado primeiro):

```
$

// ---------------------------------------------
// PARTE 1: iniciar o servidor brevemente e aceitar a conexã
// ---------------------------------------------

// Aguardar conexões de entrada na porta 12000
Server.Listen(12000, SOCK_DESTINY);

do
  v[1] = Server.Accept(Client.NextFree());
  pause;
loop until(v[1] != -1);

// A conexão foi aceita
// -> fechar o servidor (a conexão já aceita se
Server.Close();

// ---------------------------------------------
// PARTE 2: receber os dados (assim que disponíveis)
// ---------------------------------------------

// Aguardar dados de entrada
Destiny.Debugger.Output.Clear();
do
  v[2] = Client[v[1]].Destiny.GetRecvType();
  if(v[2] > 0)
    // Os dados estão disponíveis e prontos para serem recebidos
    // -> Primeiro consultar o Id
    v[3] = Client[v[1]].Destiny.RecvId();
```

```
    // -> Depois consultar o valor (dependendo do tipo)
    switch(v[2])
      case TYPE_VARIABLE:
        Destiny.Debugger.Output.Print("Variable e
        break;
      case TYPE_DWORD:
        Destiny.Debugger.Output.Print("Dword empf
        break;
      case TYPE_WORD:
        Destiny.Debugger.Output.Print("Word empfa
        break;
      case TYPE_BYTE:
        Destiny.Debugger.Output.Print("Byte empfa
        break;
      case TYPE_SWITCH:
        Destiny.Debugger.Output.Print("Switch emp
        break;
      case TYPE_FLOAT:
        Destiny.Debugger.Output.Print("Float empf
        break;
      case TYPE_STRING:
        Destiny.Debugger.Output.Print("String emp
        break;
    endswitch;
  endif;
  pause;
loop while(v[2] > 0 || Client[v[1]].State == STAT

// Ao final do script, ainda fechar a conexão
Client[v[1]].Close();

// Exibir a saída de depuração
Destiny.Debugger.Output.Show();
```

DestinyScript do cliente (envia os dados aqui; o servidor já precisa estar rodando):

```
$

// ------------------------------------------
// PARTE 1: estabelecer conexão com o servidor (aqui
// ------------------------------------------
v[1] = Client.NextFree();
Client[v[1]].Connect("127.0.0.1", 12000, SOCK_DES

// ------------------------------------------
// PARTE 2: enviar os dados
// ------------------------------------------
Client[v[1]].Destiny.SendVariable(1, 1234);   //
d[1] = 4321;
Client[v[1]].Destiny.SendDword(2, d[1]);      //
Client[v[1]].Destiny.SendWord(13, -11);       //
Client[v[1]].Destiny.SendByte(0, 255);        //
Client[v[1]].Destiny.SendFloat(-1, 3.1415);   //
Client[v[1]].Destiny.SendSwitch(1, true);     //
Client[v[1]].Destiny.SendString(2, "Hallo");  //

// Fechar a conexão (isso faz o exemplo do servidor
Client[v[1]].Close();
```

## Erros possíveis

`DESTINYERROR_SOCKETNOTOPEN`, `DESTINYERROR_SOCKETTYPE`

## Veja também

`Client`, `Client[]`, `Client[].Destiny`, `Server`, `Server.Listen()`, `Server.Accept()`, `Server.Close()`, `Client[].Connect()`, `Client[].Close()`, `Client[].Destiny.SendVariable()`, `Client[].Destiny.SendDword()`, `Client[].Destiny.SendByte()`, `Client[].Destiny.SendFloat()`, `Client[].Destiny.SendString()`, `Client[].Destiny.SendSwitch()`, `Client[].Destiny.GetRecvType()`, `Client[].Destiny.RecvId()`, `Client[].Destiny.RecvVariable()`, `Client[].Destiny.RecvDword()`, `Client[].Destiny.RecvWord()`, `Client[].Destiny.RecvByte()`, `Client[].Destiny.RecvFloat()`, `Client[].Destiny.RecvString()`, `Client[].Destiny.RecvSwitch()`

OBJETO

# Objeto: Client[].Raw

## Sintaxe

```
Client[Cliente-Índice].Raw
```

## Descrição

O objeto Raw dentro do objeto `Client[]` serve para reunir todos os métodos disponíveis apenas para sockets cliente do tipo Raw. O tipo de um socket pode ser obtido com `Client[].Type`, e é definido no método `Connect` do objeto `Client[]` ou no método `Listen` do objeto `Server`.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Recv()` | Método | Com o método Recv é possível receber dados por um socket cliente Raw conectado. |
| `Send()` | Método | Com o método Send é possível enviar dados por um socket cliente Raw conectado. |

## Veja também

`Client`, `Client[]`, `Client[].Destiny`

MÉTODO

# Método: Client[].Raw.Recv()

## Sintaxe

```
Client[Cliente-Índice].Raw.Recv([Datentyp, [Comprimento]])
```

## Descrição

Com o método Recv é possível receber dados por um socket cliente Raw conectado. A quantidade de bytes a receber pode ser informada; se essa indicação faltar, o valor inteiro é recebido.

> **Informação:** Para descobrir o comprimento dos dados já recebidos, e para de fato receber dados, é necessário chamar antes o método `GetRecvLength` do objeto `Client[]`. Se o valor de retorno desse método

> for maior que 0, há dados pendentes que podem ser recebidos. Se o método `GetRecvLength` não for chamado antes, nenhum dado é recebido por este método, pois o recebimento efetivo dos dados acontece em `GetRecvLength`.

## Lista de parâmetros

### Parâmetro: `Tipo de dado`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 5 (TYPE_STRING)
- **Faixa de valores:** 1 ... 7
- **Descrição:** O tipo de dado dos dados a receber. Para isso são usadas as constantes de tipo de dado:

Constante | Descrição

`TYPE_VARIABLE` Veja em TYPE_DWORD

`TYPE_DWORD` O valor é devolvido como Dword.

`TYPE_WORD` O valor é devolvido como Word.

`TYPE_BYTE` O valor é devolvido como Byte.

`TYPE_FLOAT` O valor é devolvido como Float.

`TYPE_STRING` O valor é devolvido como String.

`TYPE_SWITCH` O valor é devolvido como Bool.

### Parâmetro: `Comprimento`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** -1
- **Faixa de valores:** -1; 0 ... comprimento máximo dos dados
- **Descrição:** O comprimento dos dados a receber. Esse valor precisa estar na faixa de 0 até o comprimento máximo dos dados do respectivo tipo de dado. Em Strings, esse valor não pode ser maior que o buffer de recebimento (no mínimo 255 bytes). Se aqui for informado -1, o comprimento máximo dos dados é determinado e recebido automaticamente. Estão definidos os seguintes comprimentos máximos:

Tipo de dado | Comprimento máximo

Dword 4 Word 2

Byte 1

Float 8

String individual (aqui são devolvidos todos os dados já recebidos)

Bool 1

## Exemplo

```
$
// Estabelecer conexão com o Google
Client[0].Connect("www.google.de", 80, SOCK_RAW,
```

```
do while(Client[0].State == STATE_CONNECTING)
  pause;
loop;

if(Client[0].State == STATE_CONNECTED)
  // Foi possível estabelecer conexão com o Google
  // -> o socket está pronto para enviar/rece

  // Enviar uma requisição simples ao Google
  Client[0].Raw.Send(<<<HEREDOC_END
GET / HTTP/1.1
Host: www.google.de
Connection: close

HEREDOC_END);

  // Aguardar a resposta e guardá-la em a[1]
  a[1] = "";
  do until(Client[0].State != STATE_CONNECTED)
    v[1] = Client[0].GetRecvLength();
    if(v[1] > 0)
      a[1] ..= Client[0].Raw.Recv();
      if(String.Pos(a[1], CRLF .. CRLF) >= 0)
        // os cabeçalhos de resposta HTTP terminam com uma l
        break;
      endif;
    endif;
    pause;
  loop while(v[1] > 0 || Client[0].State == STATE
endif;

// Fechar o socket novamente
Client[0].Close();

// Exibir os dados recebidos na saída de depuração
Destiny.Debugger.Output.Clear();
Destiny.Debugger.Output.Print(a[1]);
Destiny.Debugger.Output.Show();
```

## Erros possíveis

`DESTINYERROR_SOCKETTYPE`, `DESTINYERROR_OOB`, `DESTINYERROR_VALUE`, `DESTINYERROR_MEMORY`

## Veja também

`Client`, `Client[]`, `Client[].Raw`, `Client[].Connect()`, `Client[].Close()`, `Client[].GetRecvLength()`, `Client[].Raw.Send()`

MÉTODO

# Método: Client[].Raw.Send()

## Sintaxe

```
Client[Cliente-Índice].Raw.Send(Daten, [Comprimento])
```

## Descrição

Com o método Send é possível enviar dados por um socket cliente Raw conectado. A quantidade de bytes a enviar pode ser informada; se essa indicação faltar, o valor inteiro é enviado.

Alteração em relação ao DestinyPatch 1.0: no DestinyPatch 1.0, o valor de parâmetro 0 em Comprimento significava que o comprimento máximo seria enviado. Isso, porém, não estava documentado, por isso essa propriedade agora foi associada ao valor de parâmetro -1.

## Lista de parâmetros

### Parâmetro: `Dados`

- **Tipo de dado:** `any`
- **Opcional:** não
- **Descrição:** Os dados a enviar.

### Parâmetro: `Comprimento`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** -1
- **Faixa de valores:** -1; 0 ... comprimento máximo dos dados
- **Descrição:** O comprimento dos dados a enviar. Esse valor precisa estar na faixa de 0 até o comprimento máximo dos dados do respectivo tipo de dado. Se aqui for informado -1, o comprimento máximo dos dados é determinado e enviado automaticamente. Estão definidos os seguintes comprimentos máximos:

Tipo de dado | Comprimento máximo

Dword 4

Word 2

Byte 1

Float 8

String individual (o valor de retorno de `String.Length()`)

Bool 1

## Exemplo

```
$
// Estabelecer conexão com o Google
Client[0].Connect("www.google.de", 80, SOCK_RAW,

do while(Client[0].State == STATE_CONNECTING)
  pause;
loop;

if(Client[0].State == STATE_CONNECTED)
  // Foi possível estabelecer conexão com o Google
  // -> o socket está pronto para enviar/rece

  // Enviar uma requisição simples ao Google
  Client[0].Raw.Send(<<<HEREDOC_END
GET / HTTP/1.1
Host: www.google.de
```

```
Connection: close

HEREDOC_END);

  // Aguardar a resposta e guardá-la em a[1]
  a[1] = "";
  do until(Client[0].State != STATE_CONNECTED)
    v[1] = Client[0].GetRecvLength();
    if(v[1] > 0)
      a[1] ..= Client[0].Raw.Recv();
      if(String.Pos(a[1], CRLF .. CRLF) >= 0)
        // os cabeçalhos de resposta HTTP terminam com uma l
        break;
      endif;
    endif;
    pause;
  loop while(v[1] > 0 || Client[0].State == STATE
endif;

// Fechar o socket novamente
Client[0].Close();

// Exibir os dados recebidos na saída de depuração
Destiny.Debugger.Output.Clear();
Destiny.Debugger.Output.Print(a[1]);
Destiny.Debugger.Output.Show();
```

## Erros possíveis

`DESTINYERROR_SOCKETNOTOPEN`, `DESTINYERROR_SOCKETTYPE`, `DESTINYERROR_CONVERT`, `DESTINYERROR_VALUE`

## Veja também

`Client`, `Client[]`, `Client[].Raw`, `Client[].Connect()`, `Client[].Close()`, `Client[].GetRecvLength()`, `Client[].Raw.Recv()`

CATEGORIA

# Categoria: Métodos obsoletos

## Descrição

Nesta categoria estão reunidos os métodos obsoletos de um socket cliente. Esses métodos ainda continuam disponíveis, mas agora são acessados de outra forma: dependendo do tipo de socket, usa-se o objeto `Client[].Destiny` ou o objeto `Client[].Raw`.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `GetRecvType()` | Método | Devido à reestruturação do objeto Client[], este método foi movido. |
| `RecvByte()` | Método | Devido à reestruturação do objeto Client[], este método foi movido. |
| `RecvDouble()` | Método | Devido à reestruturação do objeto Client[], este método foi movido. |
| `RecvDword()` | Método | Devido à reestruturação do objeto Client[], este método foi movido. |
| `RecvId()` | Método | Devido à reestruturação do objeto Client[], este método foi movido. |

| Nome | Tipo | Resumo |
|---|---|---|
| `RecvRawData()` | Método | Devido à reestruturação do objeto Client[], este método foi movido. |
| `RecvString()` | Método | Devido à reestruturação do objeto Client[], este método foi movido. |
| `RecvSwitch()` | Método | Devido à reestruturação do objeto Client[], este método foi movido. |
| `RecvVariable()` | Método | Devido à reestruturação do objeto Client[], este método foi movido. |
| `RecvWord()` | Método | Devido à reestruturação do objeto Client[], este método foi movido. |
| `SendByte()` | Método | Devido à reestruturação do objeto Client[], este método foi movido. |
| `SendDouble()` | Método | Devido à reestruturação do objeto Client[], este método foi movido. |
| `SendDword()` | Método | Devido à reestruturação do objeto Client[], este método foi movido. |
| `SendRawData()` | Método | Devido à reestruturação do objeto Client[], este método foi movido. |
| `SendString()` | Método | Devido à reestruturação do objeto Client[], este método foi movido. |
| `SendSwitch()` | Método | Devido à reestruturação do objeto Client[], este método foi movido. |
| `SendVariable()` | Método | Devido à reestruturação do objeto Client[], este método foi movido. |
| `SendWord()` | Método | Devido à reestruturação do objeto Client[], este método foi movido. |

MÉTODO

# Método: Client[].GetRecvType()

## Sintaxe

```
Client[Cliente-Índice].GetRecvType()
```

## Descrição

Devido à reestruturação do objeto `Client[]`, este método foi movido. O comando Client[].GetRecvType() ainda é válido, mas é considerado obsoleto. Em vez dele, deve-se usar o comando `Client[].Destiny.GetRecvType()`.

## Veja também

`Client[].Destiny`

MÉTODO

# Método: Client[].RecvByte()

## Sintaxe

```
Client[Cliente-Índice].RecvByte()
```

## Descrição

Devido à reestruturação do objeto `Client[]`, este método foi movido. O comando Client[].RecvByte() ainda é válido, mas é considerado obsoleto. Em vez dele, deve-se usar o comando `Client[].Destiny.RecvByte()`.

## Veja também

`Client[].Destiny`

MÉTODO

# Método: Client[].RecvDouble()

## Sintaxe

```
Client[Cliente-Índice].RecvDouble()
```

## Descrição

Devido à reestruturação do objeto `Client[]`, este método foi movido. O comando Client[].RecvDouble() ainda é válido, mas é considerado obsoleto. Em vez dele, deve-se usar o comando `Client[].Destiny.RecvFloat()`.

## Veja também

`Client[].Destiny`

MÉTODO

# Método: Client[].RecvDword()

## Sintaxe

```
Client[Cliente-Índice].RecvDword()
```

## Descrição

Devido à reestruturação do objeto `Client[]`, este método foi movido. O comando Client[].RecvDword() ainda é válido, mas é considerado obsoleto. Em vez dele, deve-se usar o comando `Client[].Destiny.RecvDword()`.

## Veja também

`Client[].Destiny`

MÉTODO

# Método: Client[].RecvId()

## Sintaxe

```
Client[Cliente-Índice].RecvId()
```

## Descrição

Devido à reestruturação do objeto `Client[]`, este método foi movido. O comando Client[].RecvId() ainda é válido, mas é considerado obsoleto. Em vez dele, deve-se usar o comando `Client[].Destiny.RecvId()`.

## Veja também

`Client[].Destiny`

MÉTODO

# Método: Client[].RecvRawData()

## Sintaxe

```
Client[Cliente-Índice].RecvRawData([Typ,
[Comprimento]])
```

## Descrição

Devido à reestruturação do objeto `Client[]`, este método foi movido. O comando Client[].RecvRawData() ainda é válido, mas é considerado obsoleto. Em vez dele, deve-se usar o comando `Client[].Raw.Recv()`.

## Veja também

`Client[].Raw`

MÉTODO

# Método: Client[].RecvString()

## Sintaxe

```
Client[Cliente-Índice].RecvString()
```

## Descrição

Devido à reestruturação do objeto `Client[]`, este método foi movido. O comando Client[].RecvString() ainda é válido, mas é considerado obsoleto. Em vez dele, deve-se usar o comando `Client[].Destiny.RecvString()`.

## Veja também

`Client[].Destiny`

MÉTODO

# Método: Client[].RecvSwitch()

## Sintaxe

```
Client[Cliente-Índice].RecvSwitch()
```

## Descrição

Devido à reestruturação do objeto `Client[]`, este método foi movido. O comando Client[].RecvSwitch() ainda é válido, mas é considerado obsoleto. Em vez dele, deve-se usar o comando `Client[].Destiny.RecvSwitch()`.

## Veja também

`Client[].Destiny`

MÉTODO

# Método: Client[].RecvVariable()

## Sintaxe

```
Client[Cliente-Índice].RecvVariable()
```

## Descrição

Devido à reestruturação do objeto `Client[]`, este método foi movido. O comando Client[].RecvVariable() ainda é válido, mas é considerado obsoleto. Em vez dele, deve-se usar o comando `Client[].Destiny.RecvVariable()`.

## Veja também

`Client[].Destiny`

MÉTODO

# Método: Client[].RecvWord()

## Sintaxe

```
Client[Cliente-Índice].RecvWord()
```

## Descrição

Devido à reestruturação do objeto `Client[]`, este método foi movido. O comando Client[].RecvWord() ainda é válido, mas é considerado obsoleto. Em vez dele, deve-se usar o comando `Client[].Destiny.RecvWord()`.

## Veja também

`Client[].Destiny`

MÉTODO

# Método: Client[].SendByte()

## Sintaxe

```
Client[Cliente-Índice].SendByte(Id, Dados)
```

## Descrição

Devido à reestruturação do objeto `Client[]`, este método foi movido. O comando Client[].SendByte() ainda é válido, mas é considerado obsoleto. Em vez dele, deve-se usar o comando `Client[].Destiny.SendByte()`.

## Veja também

`Client[].Destiny`

MÉTODO

# Método: Client[].SendDouble()

## Sintaxe

```
Client[Cliente-Índice].SendDouble(Id, Dados)
```

## Descrição

Devido à reestruturação do objeto `Client[]`, este método foi movido. O comando Client[].SendDouble() ainda é válido, mas é considerado obsoleto. Em vez dele, deve-se usar o comando `Client[].Destiny.SendFloat()`.

## Veja também

`Client[].Destiny`

MÉTODO

# Método: Client[].SendDword()

## Sintaxe

```
Client[Cliente-Índice].SendDword(Id, Dados)
```

## Descrição

Devido à reestruturação do objeto `Client[]`, este método foi movido. O comando Client[].SendDword() ainda é válido, mas é considerado obsoleto. Em vez dele, deve-se usar o comando `Client[].Destiny.SendDword()`.

## Veja também

`Client[].Destiny`

MÉTODO

## Método: Client[].SendRawData()

### Sintaxe

```
Client[Cliente-Índice].SendRawData(Daten,
[Comprimento])
```

### Descrição

Devido à reestruturação do objeto `Client[]`, este método foi movido. O comando Client[].SendRawData() ainda é válido, mas é considerado obsoleto. Em vez dele, deve-se usar o comando `Client[].Raw.Send()`.

### Veja também

`Client[].Raw`

MÉTODO

## Método: Client[].SendString()

### Sintaxe

```
Client[Cliente-Índice].SendString(Id, Dados)
```

### Descrição

Devido à reestruturação do objeto `Client[]`, este método foi movido. O comando Client[].SendString() ainda é válido, mas é considerado obsoleto. Em vez dele, deve-se usar o comando `Client[].Destiny.SendString()`.

### Veja também

`Client[].Destiny`

MÉTODO

## Método: Client[].SendSwitch()

### Sintaxe

```
Client[Cliente-Índice].SendSwitch(Id, Dados)
```

### Descrição

Devido à reestruturação do objeto `Client[]`, este método foi movido. O comando Client[].SendSwitch() ainda é válido, mas é considerado obsoleto. Em vez dele, deve-se usar o comando `Client[].Destiny.SendSwitch()`.

## Veja também

`Client[].Destiny`

MÉTODO

# Método: Client[].SendVariable()

## Sintaxe

```
Client[Cliente-Índice].SendVariable(Id, Dados)
```

## Descrição

Devido à reestruturação do objeto `Client[]`, este método foi movido. O comando Client[].SendVariable() ainda é válido, mas é considerado obsoleto. Em vez dele, deve-se usar o comando `Client[].Destiny.SendVariable()`.

## Veja também

`Client[].Destiny`

MÉTODO

# Método: Client[].SendWord()

## Sintaxe

```
Client[Cliente-Índice].SendWord(Id, Dados)
```

## Descrição

Devido à reestruturação do objeto `Client[]`, este método foi movido. O comando Client[].SendWord() ainda é válido, mas é considerado obsoleto. Em vez dele, deve-se usar o comando `Client[].Destiny.SendWord()`.

## Veja também

`Client[].Destiny`

CATEGORIA

# Categoria: Dados de conexão

## Descrição

Nesta categoria estão as propriedades relativas à conexão.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `LocalIP` | Propriedade | Com a propriedade LocalIP é possível descobrir o IP local (= do próprio computador) usado (IP de origem) do socket cliente. |
| `LocalPort` | Propriedade | Com a propriedade LocalPort é possível descobrir a porta local (= do próprio computador) usada (porta de origem) do socket cliente. |
| `RemoteIP` | Propriedade | Com a propriedade RemoteIP é possível descobrir o IP remoto (= do outro computador) usado (IP de destino) do socket cliente. |
| `RemotePort` | Propriedade | Com a propriedade RemotePort é possível descobrir a porta remota (= do outro computador) usada (porta de destino) do socket cliente. |

PROPRIEDADE

# Propriedade: Client[].LocalIP

## Sintaxe

```
Client[Cliente-Índice].LocalIP
```

## Descrição

Com a propriedade LocalIP é possível descobrir o IP local (= do próprio computador) usado (IP de origem) do socket cliente.

> **Informação:** O valor dessa propriedade só pode ser consultado quando o socket cliente está conectado.

> **Informação:** O IP usado localmente não precisa coincidir com o IP da internet. Se o computador estiver conectado à internet por um roteador, aqui costuma ser devolvido o IP dentro da rede local.

## Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** sim

## Exemplo

```
$
// Estabelecer conexão com o Google pelo primeiro soc
Client[0].Connect("www.google.de", 80, SOCK_RAW);

// Guardar os dados de conexão locais em a[1]
a[1] = Client[0].LocalIP .. ":" .. Client[0].Loca
// Guardar os dados de conexão remotos em a[2]
a[2] = Client[0].RemoteIP .. ":" .. Client[0] ..

// Fechar o socket novamente
Client[0].Close();
```

## Erros possíveis

`DESTINYERROR_SOCKETNOTOPEN`, `DESTINYERROR_MEMORY`

## Veja também

`Client`, `Client[]`, `Client[].Connect()`, `Client[].Close()`, `Client[].LocalPort`, `Client[].RemotePort`, `Client[].RemoteIP`

PROPRIEDADE

# Propriedade: Client[].LocalPort

## Sintaxe

```
Client[Cliente-Índice].LocalPort
```

## Descrição

Com a propriedade LocalPort é possível descobrir a porta local (= do próprio computador) usada (porta de origem) do socket cliente.

> **Informação:** O valor dessa propriedade só pode ser consultado quando o socket cliente está conectado.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** 1 .. 65535

## Exemplo

```
$
// Estabelecer conexão com o Google pelo primeiro soc
Client[0].Connect("www.google.de", 80, SOCK_RAW);

// Guardar os dados de conexão locais em a[1]
a[1] = Client[0].LocalIP .. ":" .. Client[0].Loca
// Guardar os dados de conexão remotos em a[2]
a[2] = Client[0].RemoteIP .. ":" .. Client[0] ..

// Fechar o socket novamente
Client[0].Close();
```

## Erros possíveis

`DESTINYERROR_SOCKETNOTOPEN`

## Veja também

`Client`, `Client[]`, `Client[].Connect()`, `Client[].Close()`, `Client[].LocalIP`, `Client[].RemotePort`, `Client[].RemoteIP`

PROPRIEDADE

# Propriedade: Client[].RemoteIP

## Sintaxe

```
Client[Cliente-Índice].RemoteIP
```

## Descrição

Com a propriedade RemoteIP é possível descobrir o IP remoto (= do outro computador) usado (IP de destino) do socket cliente.

> **Informação:** O valor dessa propriedade só pode ser consultado quando o socket cliente está conectado.

## Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** sim

## Exemplo

```
$
// Estabelecer conexão com o Google pelo primeiro soc
Client[0].Connect("www.google.de", 80, SOCK_RAW);

// Guardar os dados de conexão locais em a[1]
a[1] = Client[0].LocalIP .. ":" .. Client[0].Loca
// Guardar os dados de conexão remotos em a[2]
a[2] = Client[0].RemoteIP .. ":" .. Client[0] ..

// Fechar o socket novamente
Client[0].Close();
```

## Erros possíveis

`DESTINYERROR_SOCKETNOTOPEN`, `DESTINYERROR_MEMORY`

## Veja também

`Client`, `Client[]`, `Client[].Connect()`, `Client[].Close()`, `Client[].LocalPort`, `Client[].LocalIP`, `Client[].RemotePort`

PROPRIEDADE

# Propriedade: Client[].RemotePort

## Sintaxe

```
Client[Cliente-Índice].RemotePort
```

## Descrição

Com a propriedade RemotePort é possível descobrir a porta remota (= do outro computador) usada (porta de destino) do socket cliente.

> **Informação:** O valor dessa propriedade só pode ser consultado quando o socket cliente está conectado.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** 1 .. 65535

## Exemplo

```
$
// Estabelecer conexão com o Google pelo primeiro soc
Client[0].Connect("www.google.de", 80, SOCK_RAW);

// Guardar os dados de conexão locais em a[1]
a[1] = Client[0].LocalIP .. ":" .. Client[0].Loca
// Guardar os dados de conexão remotos em a[2]
a[2] = Client[0].RemoteIP .. ":" .. Client[0] ..

// Fechar o socket novamente
Client[0].Close();
```

## Erros possíveis

`DESTINYERROR_SOCKETNOTOPEN`

## Veja também

`Client`, `Client[]`, `Client[].Connect()`, `Client[].Close()`, `Client[].LocalPort`, `Client[].LocalIP`, `Client[].RemoteIP`

OBJETO

# Objeto: Command

## Sintaxe

```
Command
```

## Descrição

O objeto Command serve para tornar acessíveis os comandos de evento do RPG_RT. A maioria dos comandos de evento está disponível aqui. Apenas comandos de evento que contêm blocos condicionais (por exemplo, condições, combates...) ou que dependem de outros comandos de evento (por exemplo, exibição de mensagem) não podem ser usados por aqui.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `AddAbility()` | Método | Com o método AddAbility é possível adicionar pontos de habilidade a um herói. |
| `AddActor()` | Método | Com o método AddActor é possível adicionar um herói ao grupo. |
| `AddCondition()` | Método | Com o método AddCondition é possível dar uma condição a um herói. |
| `AddExp()` | Método | Com o método AddExp é possível adicionar pontos de experiência a um herói. |
| `AddGold()` | Método | Com o método AddGold é possível adicionar dinheiro ao grupo. |
| `AddHp()` | Método | Com o método AddHp é possível dar pontos de vida a um herói. |
| `AddItem()` | Método | Com o método AddItem é possível adicionar um item ao grupo. |
| `AddLevel()` | Método | Com o método AddLevel é possível adicionar níveis a um herói. |
| `AddMp()` | Método | Com o método AddMp é possível dar pontos de magia a um herói. |
| `AddSkill()` | Método | Com o método AddSkill é possível ensinar uma técnica a um herói. |
| `AddTeleportPlace()` | Método | Com o método AddTeleportPlace é possível adicionar um ponto de teletransporte. |
| `CallCommonEvent()` | Método | Com o método CallCommonEvent é possível chamar o EventScript de um Common Event. |
| `CallEvent()` | Método | Com o método CallEvent é possível chamar o EventScript de uma determinada página de um evento. |
| `CallMenu()` | Método | Com o método CallMenu é possível chamar o menu padrão. |
| `CallSaveMenu()` | Método | Com o método CallSaveMenu é possível chamar o menu de salvar padrão. |
| `ChangeSwitch()` | Método | Com o método ChangeSwitch é possível alterar o valor de um único switch. |
| `ChangeSwitchRange()` | Método | Com o método ChangeSwitchRange é possível alterar os valores de vários switches. |
| `ChangeVariable()` | Método | Com o método ChangeVariable é possível alterar o valor de uma única variável. |
| `ChangeVariableRange()` | Método | Com o método ChangeVariableRange é possível alterar os valores de várias variáveis. |
| `ClearTimer()` | Método | Com o método ClearTimer é possível marcar o evento atual como apagado. |
| `DealDamage()` | Método | Com o método DealDamage é possível causar dano a um herói. |
| `DropEquipment()` | Método | Com o método DropEquipment é possível desequipar um herói. |

| Nome | Tipo | Resumo |
|---|---|---|
| `EnableEscape()` | Método | Com o método EnableEscape é possível ativar/desativar a fuga no sistema de batalha padrão. |
| `EnableMenu()` | Método | Com o método EnableMenu é possível ativar/desativar o menu padrão. |
| `EnableSaveMenu()` | Método | Com o método EnableSaveMenu é possível ativar/desativar o menu de salvar padrão. |
| `EnableTeleport()` | Método | Com o método EnableTeleport é possível ativar/desativar o uso de pontos de teletransporte. |
| `EnterHeroName()` | Método | Com o método EnterHeroName é possível pedir ao jogador que dê um novo nome a um herói. |
| `EnterKey()` | Método | Com o método EnterKey é possível descobrir a tecla atualmente pressionada, dentre uma pré-seleção. |
| `EnterLeaveVehicle()` | Método | Com o método EnterLeaveVehicle é possível fazer o herói entrar no veículo mais próximo, ou sair do veículo que ele estiver usando no momento. |
| `EquipItem()` | Método | Com o método EquipItem é possível equipar um herói com uma peça de equipamento. |
| `ErasePicture()` | Método | Com o método ErasePicture é possível apagar uma Picture. |
| `EraseScreen()` | Método | Com o método EraseScreen é possível apagar ("esconder") a tela, deixando-a preta. |
| `FadeOutMusic()` | Método | Com o método FadeOutMusic é possível apagar (fade-out) a música de fundo. |
| `FixScreen()` | Método | Com o método FixScreen, a tela é fixada na posição atual e, com isso, deixa de seguir o herói. |
| `FlashEvent()` | Método | Com o método FlashEvent é possível fazer um evento piscar em uma determinada cor. |
| `FlashScreen()` | Método | Com o método FlashScreen é possível fazer a tela piscar em uma determinada cor. |
| `FullRecovery()` | Método | Com o método FullRecovery é possível curar completamente um herói. |
| `GameOver()` | Método | Com o método GameOver é possível encerrar a partida atual pela tela de Game Over. |
| `GetEventId()` | Método | Com o método GetEventId é possível descobrir a Event-Id de um evento em uma determinada posição do mapa. |
| `GetTerrainId()` | Método | Com o método GetTerrainId é possível descobrir a Terrain-Id de um campo do mapa. |
| `GoToLabel()` | Método | Com o método GoToLabel é possível fazer o ponteiro de comando do evento atual pular para um determinado rótulo. |
| `GoToMemorizedPlace()` | Método | Com o método GoToMemorizedPlace é possível teletransportar o herói para uma determinada posição, usando variáveis. |
| `MemorizeMusic()` | Método | Com o método MemorizeMusic é possível guardar temporariamente a música de fundo atual. |
| `MemorizePlace()` | Método | Com o método MemorizePlace é possível guardar as coordenadas atuais do herói em variáveis. |
| `MoveAll()` | Método | Com o método MoveAll, espera-se até que todos os passos de movimento a executar tenham sido processados. |
| `MoveEvent()` | Método | Com o método MoveEvent é possível mover um evento em exatamente um passo. |
| `MovePicture()` | Método | Com o método MovePicture é possível mover uma Picture pela tela. |
| `PlayMemorizedMusic()` | Método | Com o método PlayMemorizedMusic é possível tocar uma música de fundo previamente guardada. |
| `PlayMovie()` | Método | Com o método PlayMovie é possível reproduzir um filme. |
| `PlayMusic()` | Método | Com o método PlayMusic é possível tocar uma música de fundo. |
| `PlaySound()` | Método | Com o método PlaySound é possível reproduzir um som. |

| Nome | Tipo | Resumo |
|---|---|---|
| `ReleaseScreen()` | Método | Com o método ReleaseScreen, a fixação da tela (caso tenha sido feita antes) é desfeita. |
| `RemoveActor()` | Método | Com o método RemoveActor é possível tirar um herói do grupo. |
| `RemoveCondition()` | Método | Com o método RemoveCondition é possível curar um herói de uma condição. |
| `RemoveItem()` | Método | Com o método RemoveItem é possível tirar um item do grupo. |
| `RemoveSkill()` | Método | Com o método RemoveSkill é possível fazer um herói esquecer uma técnica. |
| `RemoveTeleportPlace()` | Método | Com o método RemoveTeleportPlace é possível remover um ponto de teletransporte existente. |
| `ReplaceLowerChip()` | Método | Com o método ReplaceLowerChip é possível substituir todas as ocorrências de um determinado Lower-Chip não animado no mapa por outro Lower-Chip não animado. |
| `ReplaceUpperChip()` | Método | Com o método ReplaceUpperChip é possível substituir todas as ocorrências de um determinado Upper-Chip no mapa por outro Upper-Chip. |
| `ReturnScreen()` | Método | Com o método ReturnScreen é possível deslocar a tela de volta para o herói. |
| `SetChipset()` | Método | Com o método SetChipset é possível definir o chipset atualmente usado pelo mapa. |
| `SetEnemyEncounter()` | Método | Com o método SetEnemyEncounter é possível definir a quantidade máxima de passos necessária para que um inimigo apareça. |
| `SetEscapePlace()` | Método | Com o método SetEscapePlace é possível definir o ponto de fuga do sistema de batalha padrão. |
| `SetEventPlace()` | Método | Com o método SetEventPlace é possível definir a posição de um evento. |
| `SetHeroChar()` | Método | Com o método SetHeroChar é possível definir o char de um herói. |
| `SetHeroDegree()` | Método | Com o método SetHeroDegree é possível definir o título de um herói. |
| `SetHeroFace()` | Método | Com o método SetHeroFace é possível definir o face de um herói. |
| `SetHeroName()` | Método | Com o método SetHeroName é possível definir o nome de um herói. |
| `SetHeroTransparency()` | Método | Com o método SetHeroTransparency é possível definir a transparência do evento do herói. |
| `SetMessageStyle()` | Método | Com o método SetMessageStyle é possível definir o estilo das janelas de mensagem. |
| `SetPanorama()` | Método | Com o método SetPanorama é possível alterar o panorama atualmente usado, incluindo as propriedades de scroll. |
| `SetScreenTone()` | Método | Com o método SetScreenTone é possível colorir a tela. |
| `SetScreenTransition()` | Método | Com o método SetScreenTransition é possível definir a transição de um determinado evento de transição. |
| `SetSystemGraphic()` | Método | Com o método SetSystemGraphic é possível definir o gráfico do sistema. |
| `SetSystemMusic()` | Método | Com o método SetSystemMusic é possível definir a música de fundo de um evento do sistema. |
| `SetSystemSound()` | Método | Com o método SetSystemSound é possível definir o som de um evento do sistema. |
| `SetTimer()` | Método | Com o método SetTimer é possível definir o valor atual do timer. |
| `SetVehicleChar()` | Método | Com o método SetVehicleChar é possível definir o char de um veículo. |
| `SetVehiclePlace()` | Método | Com o método SetVehiclePlace é possível definir a posição de um veículo. |
| `SetWeather()` | Método | Com o método SetWeather é possível definir um determinado clima. |
| `ShakeScreen()` | Método | Com o método ShakeScreen é possível fazer a tela tremer (por exemplo, para simular um terremoto). |

| Nome | Tipo | Resumo |
|---|---|---|
| `ShiftScreen()` | Método | Com o método ShiftScreen é possível deslocar a tela (por exemplo, para mostrar um determinado ponto do mapa). |
| `ShowBattleAnimation()` | Método | Com o método ShowBattleAnimation é possível exibir uma animação de batalha. |
| `ShowFace()` | Método | Com o método ShowFace é possível definir um face para as mensagens. |
| `ShowPicture()` | Método | Com o método ShowPicture é possível carregar e exibir um gráfico como Picture. |
| `ShowScreen()` | Método | Com o método ShowScreen é possível exibir a tela por meio de fade-in ("mostrar"). |
| `StartTimer()` | Método | Com o método StartTimer é possível iniciar o timer. |
| `StopAll()` | Método | Com o método StopAll, os passos de movimento a executar são cancelados. |
| `StopMusic()` | Método | Com o método StopMusic é possível parar a música de fundo atual. |
| `StopParallelProcesses()` | Método | Com o método StopParallelProcesses é possível interromper eventos em execução (no mapa ou CommonEvent) do tipo "Parallel Process". |
| `StopTimer()` | Método | Com o método StopTimer é possível parar o timer. |
| `SubtractAbility()` | Método | Com o método SubtractAbility é possível retirar pontos de habilidade de um herói. |
| `SubtractExp()` | Método | Com o método SubtractExp é possível retirar pontos de experiência de um herói. |
| `SubtractGold()` | Método | Com o método SubtractGold é possível retirar dinheiro do grupo. |
| `SubtractHp()` | Método | Com o método SubtractHp é possível fazer um herói perder pontos de vida. |
| `SubtractLevel()` | Método | Com o método SubtractLevel é possível retirar níveis de um herói. |
| `SubtractMp()` | Método | Com o método SubtractMp é possível fazer um herói perder pontos de magia. |
| `SwapTwoEvents()` | Método | Com o método SwapTwoEvents é possível trocar a posição de dois eventos. |
| `Teleport()` | Método | Com o método Teleport é possível teletransportar o herói para uma determinada posição de um determinado mapa. |
| `TitleScreen()` | Método | Com o método TitleScreen é possível encerrar a partida atual pela tela de título. |
| `Wait()` | Método | Com o método Wait é possível interromper a execução do EventScript por um determinado tempo. |

MÉTODO

# Método: Command.AddAbility()

## Sintaxe

```
Command.AddAbility(Herói-Id, Tipo, Quantidade)
```

## Descrição

Com o método AddAbility é possível adicionar pontos de habilidade a um herói.

## Lista de parâmetros

### Parâmetro: `Herói-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... Actor.Count

- **Descrição:** A Herói-Id do herói ao qual os pontos de habilidade devem ser adicionados. Precisa estar na faixa de 1 a `Actor.Count`. Alternativamente, pode-se usar a constante `STATE_WHOLEPARTY`, para que todo o grupo receba os pontos de habilidade.

### Parâmetro: `Tipo`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 5
- **Descrição:** O tipo dos pontos de habilidade. Para isso são usadas as `constantes de valores de status`:

`ABILITY_MAXHP`: o herói recebe pontos nos pontos de vida máximos (equivale a `Actor[].MaxHpDiff` += Valor).
`ABILITY_MAXMP`: o herói recebe pontos nos pontos de magia máximos (equivale a `Actor[].MaxMpDiff` += Valor).
`ABILITY_ATTACK`: o herói recebe pontos nos pontos de ataque (equivale a `Actor[].AttackDiff` += Valor).
`ABILITY_DEFENSE`: o herói recebe pontos nos pontos de defesa (equivale a `Actor[].DefenseDiff` += Valor).
`ABILITY_MIND`: o herói recebe pontos nos pontos de inteligência (equivale a `Actor[].MindDiff` += Valor).
`ABILITY_AGILITY`: o herói recebe pontos nos pontos de agilidade (equivale a `Actor[].AgilityDiff` += Valor).

### Parâmetro: `Quantidade`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A quantidade de pontos de habilidade que o herói deve receber.

## Exemplo

```
$
// Aumentar em 50 os pontos de ataque do primeiro he
Command.AddAbility(1, ABILITY_ATTACK, 50);

// Dar a todo o grupo (= por herói) dez pontos de defe
Command.SubtractLevel(STATE_WHOLEPARTY, ABILITY_D
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.SubtractAbility()`

MÉTODO

# Método: Command.AddActor()

## Sintaxe

```
Command.AddActor(Herói-Id)
```

## Descrição

Com o método AddActor é possível adicionar um herói ao grupo.

## Lista de parâmetros

### Parâmetro: `Herói-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... Actor.Count
- **Descrição:** A Herói-Id do herói a ser adicionado ao grupo. Precisa estar na faixa de 1 a `Actor.Count`.

## Exemplo

```
$
// Adicionar ao grupo o segundo herói (geralmente Br
Command.AddActor(2);

// Tirar do grupo o primeiro herói (geralmente Alex)
Command.RemoveActor(1);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.RemoveActor()`

MÉTODO

# Método: Command.AddCondition()

## Sintaxe

```
Command.AddCondition(Herói-Id, Condição-Id)
```

## Descrição

Com o método AddCondition é possível dar uma condição a um herói.

## Lista de parâmetros

### Parâmetro: `Herói-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... Actor.Count
- **Descrição:** A Herói-Id do herói que deve receber a condição. Precisa estar na faixa de 1 a `Actor.Count`. Alternativamente, pode-se usar a constante `STATE_WHOLEPARTY`, para que todo o grupo receba a condição.

### Parâmetro: `Condição-Id`

- **Tipo de dado:** `dword`

- **Opcional:** não
- **Faixa de valores:** 1 ... Condition.Count
- **Descrição:** A Condition-Id da condição que o herói deve receber. Precisa estar na faixa de 1 a `Condition.Count`.

## Exemplo

```
$
// Dar ao primeiro herói a segunda condição (geralm
Command.AddCondition(1, 2);

// Curar todo o grupo da terceira condição (geral
Command.RemoveCondition(STATE_WHOLEPARTY, 3);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.RemoveCondition()`

MÉTODO

# Método: Command.AddExp()

## Sintaxe

```
Command.AddExp(Actor-Id, Anzahl,
[Exibir mensagem de aumento de nível])
```

## Descrição

Com o método AddExp é possível adicionar pontos de experiência a um herói.

## Lista de parâmetros

### Parâmetro: `Herói-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... Actor.Count
- **Descrição:** A Herói-Id do herói ao qual os pontos de experiência devem ser adicionados. Precisa estar na faixa de 1 a `Actor.Count`. Alternativamente, pode-se usar a constante `STATE_WHOLEPARTY`, para que todo o grupo receba pontos de experiência.

### Parâmetro: `Quantidade`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A quantidade de pontos de experiência que o herói deve receber.

### Parâmetro: `Exibir mensagem de aumento de nível`

- **Tipo de dado:** `bool`
- **Opcional:** sim
- **Valor padrão:** false
- **Descrição:** Indica se, ao subir de nível, uma mensagem correspondente deve ser exibida.

False: nenhuma mensagem de aumento de nível é exibida (este é o valor padrão). True: uma mensagem de aumento de nível é exibida quando necessário.

## Exemplo

```
$
// Adicionar 100 pontos de experiência ao primeiro he
// (aqui, se for o caso, uma mensagem de aumento de
Command.AddExp(1, 100, true);

// Adicionar 50 pontos de experiência a todo o grupo
Command.SubtractExp(STATE_WHOLEPARTY, 50);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.SubtractExp()`

MÉTODO

# Método: Command.AddGold()

## Sintaxe

```
Command.AddGold(Quantidade)
```

## Descrição

Com o método AddGold é possível adicionar dinheiro ao grupo.

## Lista de parâmetros

### Parâmetro: `Quantidade`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A quantidade de dinheiro a ser adicionada ao grupo.

## Exemplo

```
$
// Dar 100 de dinheiro ao grupo
Command.AddGold(100);

// Tirar 100 de dinheiro do grupo (assim como veio,
Command.SubtractGold(100);
```

## Veja também

`Command`, `Command.SubtractGold()`

MÉTODO

# Método: Command.AddHp()

## Sintaxe

```
Command.AddHp(Herói-Id, Quantidade)
```

## Descrição

Com o método AddHp é possível dar pontos de vida a um herói.

## Lista de parâmetros

### Parâmetro: `Herói-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... Actor.Count
- **Descrição:** A Herói-Id do herói que deve receber pontos de vida. Precisa estar na faixa de 1 a `Actor.Count`. Alternativamente, pode-se usar a constante `STATE_WHOLEPARTY`, para que todo o grupo receba pontos de vida.

### Parâmetro: `Quantidade`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A quantidade de pontos de vida que o herói deve receber. Esses pontos de vida do herói não ultrapassam os pontos de vida máximos do respectivo herói.

## Exemplo

```
$
// Dar 100 pontos de vida ao primeiro herói
Command.AddHp(1, 100);

// Dar 50 pontos de vida a todo o grupo (= por herói)
Command.SubtractHp(STATE_WHOLEPARTY, 50, false);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.SubtractHp()`

MÉTODO

# Método: Command.AddItem()

## Sintaxe

```
Command.AddItem(Item-Id, [Quantidade])
```

## Descrição

Com o método AddItem é possível adicionar um item ao grupo.

## Lista de parâmetros

### Parâmetro: `Item-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... Item.Count
- **Descrição:** A Item-Id do item a ser adicionado ao grupo. Precisa estar na faixa de 1 a `Item.Count`.

### Parâmetro: `Quantidade`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 1
- **Descrição:** A quantidade de itens a ser adicionada ao grupo. Se esse parâmetro não for definido, é adicionado um item.

## Exemplo

```
$
// Dar ao grupo o primeiro item (geralmente Potion) uma
Command.AddItem(1);

// Dar ao grupo o segundo item (geralmente Tear Drop)
Command.AddItem(2, 10);

// Tirar do grupo o segundo item (geralmente Tear Drop)
Command.RemoveItem(2, 5);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.RemoveItem()`

MÉTODO

# Método: Command.AddLevel()

## Sintaxe

```
Command.AddLevel(Actor-Id, Anzahl,
[Exibir mensagem de aumento de nível])
```

## Descrição

Com o método AddLevel é possível adicionar níveis a um herói.

## Lista de parâmetros

### Parâmetro: `Herói-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... Actor.Count
- **Descrição:** A Herói-Id do herói ao qual os níveis devem ser adicionados. Precisa estar na faixa de 1 a `Actor.Count`. Alternativamente, pode-se usar a constante `STATE_WHOLEPARTY`, para que todo o grupo receba níveis.

### Parâmetro: `Quantidade`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A quantidade de níveis que o herói deve receber.

### Parâmetro: `Exibir mensagem de aumento de nível`

- **Tipo de dado:** `bool`
- **Opcional:** sim
- **Valor padrão:** false
- **Descrição:** Indica se, ao subir de nível, uma mensagem correspondente deve ser exibida.

False: nenhuma mensagem de aumento de nível é exibida (este é o valor padrão). True: uma mensagem de aumento de nível é exibida quando necessário.

## Exemplo

```
$
// Fazer o primeiro herói subir dois níveis
// (aqui, se for o caso, uma mensagem de aumento de
Command.AddLevel(1, 2, true);

// Fazer todo o grupo (= por herói) descer um nível
Command.SubtractLevel(STATE_WHOLEPARTY, 1);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.SubtractLevel()`

MÉTODO

# Método: Command.AddMp()

## Sintaxe

```
Command.AddMp(Herói-Id, Quantidade)
```

## Descrição

Com o método AddMp é possível dar pontos de magia a um herói.

## Lista de parâmetros

### Parâmetro: `Herói-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... Actor.Count
- **Descrição:** A Herói-Id do herói que deve receber pontos de magia. Precisa estar na faixa de 1 a `Actor.Count`. Alternativamente, pode-se usar a constante `STATE_WHOLEPARTY`, para que todo o grupo receba pontos de magia.

### Parâmetro: `Quantidade`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A quantidade de pontos de magia que o herói deve receber. Os pontos de magia do herói não ultrapassam os pontos de magia máximos do respectivo herói.

## Exemplo

```
$
// Dar 100 pontos de magia ao primeiro herói
Command.AddMp(1, 100);

// Dar 50 pontos de magia a todo o grupo (= por herói)
Command.SubtractMp(STATE_WHOLEPARTY, 50);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.SubtractMp()`

MÉTODO

# Método: Command.AddSkill()

## Sintaxe

```
Command.AddSkill(Herói-Id, Habilidade-Id)
```

## Descrição

Com o método AddSkill é possível ensinar uma técnica a um herói.

## Lista de parâmetros

### Parâmetro: `Herói-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... Actor.Count
- **Descrição:** A Herói-Id do herói que deve aprender a técnica. Precisa estar na faixa de 1 a `Actor.Count`. Alternativamente, pode-se usar a constante `STATE_WHOLEPARTY`, para que todo o grupo aprenda a técnica.

### Parâmetro: `Habilidade-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... Skill.Count
- **Descrição:** A Skill-Id da técnica a ser ensinada ao herói. Precisa estar na faixa de 1 a `Skill.Count`.

## Exemplo

```
$
// Ensinar ao primeiro herói a segunda técnica (geral
Command.AddSkill(1, 2);
```

```
// Ensinar a todo o grupo (= por herói) a primeira téc
Command.RemoveSkill(STATE_WHOLEPARTY, 1);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.RemoveSkill()`

MÉTODO

# Método: Command.AddTeleportPlace()

## Sintaxe

```
Command.AddTeleportPlace(Map-Id, X, Y, [Switch-Id])
```

## Descrição

Com o método AddTeleportPlace é possível adicionar um ponto de teletransporte. Os pontos de teletransporte podem ser usados com técnicas do tipo "Teleport".

## Lista de parâmetros

### Parâmetro: `Mapa-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** A Map-Id do novo ponto de teletransporte. Esse valor precisa ser maior/igual a 1.

### Parâmetro: `X`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A coordenada X do novo ponto de teletransporte.

### Parâmetro: `Y`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A coordenada Y do novo ponto de teletransporte.

### Parâmetro: `Switch-Id`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 0

- **Faixa de valores:** >= 0
- **Descrição:** A Switch-Id de um switch a ser ativado (quando o ponto de teletransporte for usado), ou 0, se nenhum switch deve ser ativado.

## Exemplo

```
$
// Adicionar a posição atual do herói como ponto de tele
Command.AddTeleportPlace(Map.Id, Map.HeroX, Map.H
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.RemoveTeleportPlace()`, `Command.EnableTeleport()`, `Map.Id`, `Map.HeroX`, `Map.HeroY`

MÉTODO

# Método: Command.CallCommonEvent()

## Sintaxe

```
Command.CallCommonEvent(Evento-Comum-Id)
```

## Descrição

Com o método CallCommonEvent é possível chamar o EventScript de um Common Event. O EventScript do Common Event só é executado depois do DestinyScript (e, se for o caso, do tempo de espera do evento atual).

## Lista de parâmetros

### Parâmetro: `Evento-Comum-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... CommonEvent.Count
- **Descrição:** A Common-Event-Id do Common Event a ser chamado. Precisa estar na faixa de 1 a `CommonEvent.Count`.

## Exemplo

```
$
// Chama o primeiro Common Event
Command.CallCommonEvent(1);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.CallEvent()`, `CommonEvent[]`

MÉTODO

# Método: Command.CallEvent()

## Sintaxe

```
Command.CallEvent(Event-Id, [Página-do-evento-Id])
```

## Descrição

Com o método CallEvent é possível chamar o EventScript de uma determinada página de um evento. O EventScript do evento só é executado depois do DestinyScript (e, se for o caso, do tempo de espera do evento atual).

## Lista de parâmetros

### Parâmetro: `Evento-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** Event-Id válida
- **Descrição:** A Event-Id do evento a ser executado. Esse valor precisa ser uma Event-Id válida. Além disso, também pode ser usada a constante `THIS`, para executar o evento atual.

### Parâmetro: `Página-do-evento-Id`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 1
- **Faixa de valores:** Número de página válido do respectivo evento
- **Descrição:** A Event-Page-Id da página de evento a ser executada. Esse valor precisa ser uma Event-Page-Id válida e, portanto, fica na faixa de 1 até a quantidade de páginas do respectivo evento (esse valor atualmente não pode ser consultado por DestinyScript).

## Exemplo

```
$
// Chama a primeira página do evento atual
Command.CallEvent(THIS, 1);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.CallCommonEvent()`, `Event[]`

MÉTODO

# Método: Command.CallMenu()

## Sintaxe

```
Command.CallMenu()
```

## Descrição

Com o método CallMenu é possível chamar o menu padrão.

## Exemplo

```
$
// Chamar o menu padrão
Command.CallMenu();
```

## Veja também

`Command`, `Command.EnableMenu()`

MÉTODO

# Método: Command.CallSaveMenu()

## Sintaxe

```
Command.CallSaveMenu()
```

## Descrição

Com o método CallSaveMenu é possível chamar o menu de salvar padrão.

## Exemplo

```
$
// Chamar o menu de salvar
Command.CallSaveMenu();
```

## Veja também

`Command`, `Command.EnableSaveMenu()`

MÉTODO

# Método: Command.ChangeSwitch()

## Sintaxe

```
Command.ChangeSwitch(Switch-Id, Operação)
```

## Descrição

Com o método ChangeSwitch é possível alterar o valor de um único switch.

## Lista de parâmetros

### Parâmetro: `Switch-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** A Switch-Id do switch a ser alterado.

### Parâmetro: `Operação`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 2
- **Descrição:** A operação a ser realizada. Para isso são usadas as `constantes de operação para switches`.

`OPERATION_ON`: o switch é definido como LIGADO (o valor em s[...] fica, portanto, true depois). `OPERATION_OFF`: o switch é definido como DESLIGADO (o valor em s[...] fica, portanto, false depois). `OPERATION_TOGGLE`: o valor do switch é invertido (o valor em s[...] fica, portanto, !s[...] depois).

## Exemplo

```
$
// Definir o valor do primeiro switch como true
Command.ChangeSwitch(1, OPERATION_ON);

// Acesso totalmente equivalente pelo objeto s[]
s[1] = true;
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.ChangeSwitchRange()`, `s[]`

MÉTODO

# Método: Command.ChangeSwitchRange()

## Sintaxe

```
Command.ChangeSwitchRange(Anfangs-Switch-Id, End-
Switch-Id, Operation)
```

## Descrição

Com o método ChangeSwitchRange é possível alterar os valores de vários switches. Os switches são indicados em forma de intervalo. Por isso, são alterados os valores de todos os switches de Switch-Id inicial até Switch-Id final.

## Lista de parâmetros

### Parâmetro: `Switch-Id inicial`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** A Switch-Id do primeiro switch a ser alterado.

### Parâmetro: `Switch-Id final`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** A Switch-Id do último switch a ser alterado.

### Parâmetro: `Operação`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 2
- **Descrição:** A operação a ser realizada. Para isso são usadas as `constantes de operação para switches`.

`OPERATION_ON`: o switch é definido como LIGADO (o valor em s[...] fica, portanto, true depois). `OPERATION_OFF`: o switch é definido como DESLIGADO (o valor em s[...] fica, portanto, false depois). `OPERATION_TOGGLE`: o valor do switch é invertido (o valor em s[...] fica, portanto, !s[...] depois).

## Exemplo

```
$
// Inverter os valores dos primeiros 100 switches
Command.ChangeSwitchRange(1, 100, OPERATION_TOGGL

// Acesso totalmente equivalente pelo objeto s[]
for(v[1] = 1; v[1] <= 100; v[1]++)
  s[v[1]] = !s[v[1]];
next;
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.ChangeSwitch()`, `s[]`

MÉTODO

# Método: Command.ChangeVariable()

## Sintaxe

```
Command.ChangeVariable(Variable-Id, Operation,
Wertetyp, Wert1, Wert2)
```

## Descrição

Com o método ChangeVariable é possível alterar o valor de uma única variável.

## Lista de parâmetros

### Parâmetro: `Variável-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** A Variable-Id da variável a ser alterada.

### Parâmetro: `Operação`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 5
- **Descrição:** A operação a ser realizada. Para isso são usadas as `constantes de operação para variáveis`.

`OPERATION_SET`: a variável é definida com o valor indicado (equivale a v[...] = Valor). `OPERATION_ADD`: o valor indicado é somado à variável (equivale a v[...] += Valor). `OPERATION_SUBTRACT`: o valor indicado é subtraído da variável (equivale a v[...] -= Valor). `OPERATION_MULTIPLY`: a variável é multiplicada pelo valor indicado (equivale a v[...] *= Valor). `OPERATION_DIVIDE`: a variável é dividida (como inteiro) pelo valor indicado (equivale a v[...] /= Valor). `OPERATION_MODULO`: a variável é dividida (como inteiro) pelo valor indicado, sendo devolvido o resto não divisível (equivale a v[...] %= Valor).

### Parâmetro: `Tipo de valor`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 7
- **Descrição:** O tipo do valor usado nesta operação de variável.

Para isso são usadas as `constantes de tipo de valor`.

`VALUETYPE_NUMBER`: o valor é um número informado diretamente. `VALUETYPE_VARIABLE`: o valor é obtido de outra variável. `VALUETYPE_VARIABLEINDIRECT`: o valor é obtido de outra variável, referenciada por mais uma variável. `VALUETYPE_RANDOM`: o valor é aleatório e fica em uma faixa de um mínimo a um máximo indicados. `VALUETYPE_ITEM`: o valor vem de um item. `VALUETYPE_ACTOR`: o valor vem de um herói. `VALUETYPE_EVENT`: o valor vem de um evento. `VALUETYPE_MISC`: o valor vem de outra fonte qualquer.

### Parâmetro: `Valor1`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** O significado do valor informado neste parâmetro depende do parâmetro "Tipo de valor".

Em `VALUETYPE_NUMBER`, este é o valor da operação. Em `VALUETYPE_VARIABLE`, esta é a Variable-Id da variável cujo valor deve ser usado. Em `VALUETYPE_VARIABLEINDIRECT`, esta é a Variable-Id da variável cujo valor referencia a variável cujo valor deve ser usado de fato. Em `VALUETYPE_RANDOM`, este é o mínimo do número aleatório. Em `VALUETYPE_ITEM`, esta é a Item-Id do item do qual um valor deve ser usado. Em `VALUETYPE_ACTOR`, esta é a Actor-Id do herói do qual um valor deve ser usado.

Em `VALUETYPE_EVENT`, esta é a Event-Id do evento do qual um valor deve ser usado. Em `VALUETYPE_MISC`, este é um valor que determina o tipo do valor consultado. Para isso são usadas as `constantes de outros valores`:

`MISC_MONEY`: a quantidade de dinheiro do grupo. `MISC_TIMERSECONDSLEFT`: a quantidade de segundos restantes do timer. `MISC_PARTYSIZE`: o tamanho do grupo. `MISC_NUMBEROFSAVES`: a quantidade de vezes que o jogo foi salvo até agora. `MISC_NUMBEROFBATTLES`: a quantidade de combates até agora. `MISC_NUMBEROFVICTORIES`: a quantidade de vitórias até agora. `MISC_NUMBEROFDEFEATS`: a quantidade de derrotas até agora. `MISC_NUMBEROFESCAPES`: a quantidade de fugas até agora. `MISC_MIDITICK`: o tick MIDI atual.

### Parâmetro: `Valor2`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** O significado do valor informado neste parâmetro depende do parâmetro "Tipo de valor".

Em `VALUETYPE_NUMBER`, este parâmetro não é usado e, por isso, deveria ser 0. Em `VALUETYPE_VARIABLE`, este parâmetro não é usado e, por isso, deveria ser 0. Em `VALUETYPE_VARIABLEINDIRECT`, este parâmetro não é usado e, por isso, deveria ser 0. Em `VALUETYPE_RANDOM`, este é o máximo do número aleatório. Em `VALUETYPE_ITEM`, este é um valor que determina o tipo do valor consultado. Para isso são usadas as `constantes de items`:

`ITEM_OWNED`: a quantidade de itens desse tipo que o grupo possui no momento. `ITEM_EQUIPPED`: a quantidade de itens desse tipo atualmente em uso como equipamento no grupo.

Em `VALUETYPE_ACTOR`, este é um valor que determina o tipo do valor consultado. Para isso são usadas as `constantes de heróis`:

`ACTOR_LEVEL`: o nível atual do herói (equivale a `Actor[].Level`). `ACTOR_EXP`: a experiência atual do herói (equivale a `Actor[].Exp`). `ACTOR_HP`: os pontos de vida atuais do herói (equivale a `Actor[].Hp`). `ACTOR_MP`: os pontos de magia atuais do herói (equivale a `Actor[].Mp`). `ACTOR_MAXHP`: os pontos de vida máximos atuais do herói (não equivale a `Actor[].MaxHpDiff`). `ACTOR_MAXMP`: os pontos de magia máximos atuais do herói (não equivale a `Actor[].MaxMpDiff`). `ACTOR_ATTACK`: os pontos de ataque atuais do herói (não equivale a `Actor[].AttackDiff`). `ACTOR_DEFENSE`: os pontos de defesa atuais do herói (não equivale a `Actor[].DefenseDiff`). `ACTOR_MIND`: os pontos de inteligência atuais do herói (não equivale a `Actor[].MindDiff`). `ACTOR_AGILITY`: os pontos de agilidade atuais do herói (não equivale a `Actor[].AgilityDiff`). `ACTOR_WEAPON`: a arma atual do herói (equivale a `Actor[].Weapon`). `ACTOR_SHIELD`: o escudo atual do herói (equivale a `Actor[].Shield`). `ACTOR_ARMOR`: a armadura atual do herói (equivale a `Actor[].Armor`). `ACTOR_HELMET`: o elmo atual do herói (equivale a `Actor[].Helmet`).

`ACTOR_ACCESSORY`: o acessório atual do herói (equivale a `Actor[].Accessory`).

Em `VALUETYPE_EVENT`, este é um valor que determina o tipo do valor consultado. Para isso são usadas as `constantes de eventos`:

`EVENT_MAPID`: a Map-Id do evento (equivale a `Map.Id`). `EVENT_X`: a coordenada X do evento (equivale a `Event[].X`). `EVENT_Y`: a coordenada Y do evento (equivale a `Event[].Y`). `EVENT_DIRECTION`: a direção para a qual o evento olha (8 = cima, 6 = direita, 4 = esquerda, 2 = baixo; não equivale a `Event[].DirLook`, embora possa ser usado de forma semelhante). `EVENT_SCREENX`: a coordenada X do evento na tela (equivale a `Event[].ScreenX`). `EVENT_SCREENY`: a coordenada Y do evento na tela (equivale a `Event[].ScreenY`).

## Exemplo

```
$
// Definir o valor da primeira variável como 3
Command.ChangeVariable(1, OPERATION_SET, VALUETYP
// Acesso totalmente equivalente pelo objeto v[]
v[1] = 3;

// Definir o valor da segunda variável como o valor d
Command.ChangeVariable(2, OPERATION_SET, VALUETYP
// Acesso totalmente equivalente pelo objeto v[]
v[2] = v[1];

// Guardar na terceira variável a quantidade de itens com
Command.ChangeVariable(3, OPERATION_SET, VALUETYP
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.ChangeVariableRange()`, `v[]`

MÉTODO

# Método: Command.ChangeVariableRange()

## Sintaxe

```
Command.ChangeVariableRange(Anfangs-Variable-Id, End-
Variable-Id, Operation, Wertetyp, Wert1, Wert2)
```

## Descrição

Com o método ChangeVariableRange é possível alterar os valores de várias variáveis. As variáveis são indicadas em forma de intervalo. Por isso, são alterados os valores de todas as variáveis de Variable-Id inicial até Variable-Id final.

## Lista de parâmetros

### Parâmetro: `Variável-Id inicial`

- **Tipo de dado:** `dword`

- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** A Variable-Id da primeira variável a ser alterada.

### Parâmetro: `Variável-Id final`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** A Variable-Id da última variável a ser alterada.

### Parâmetro: `Operação`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 5
- **Descrição:** A operação a ser realizada. Para isso são usadas as `constantes de operação para variáveis`.

`OPERATION_SET`: a variável é definida com o valor indicado (equivale a v[...] = Valor). `OPERATION_ADD`: o valor indicado é somado à variável (equivale a v[...] += Valor). `OPERATION_SUBTRACT`: o valor indicado é subtraído da variável (equivale a v[...] -= Valor). `OPERATION_MULTIPLY`: a variável é multiplicada pelo valor indicado (equivale a v[...] *= Valor). `OPERATION_DIVIDE`: a variável é dividida (como inteiro) pelo valor indicado (equivale a v[...] /= Valor). `OPERATION_MODULO`: a variável é dividida (como inteiro) pelo valor indicado, sendo devolvido o resto não divisível (equivale a v[...] %= Valor).

### Parâmetro: `Tipo de valor`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 7
- **Descrição:** O tipo do valor usado nesta operação de variável. Para isso são usadas as `constantes de tipo de valor`.

`VALUETYPE_NUMBER`: o valor é um número informado diretamente. `VALUETYPE_VARIABLE`: o valor é obtido de outra variável. `VALUETYPE_VARIABLEINDIRECT`: o valor é obtido de outra variável, referenciada por mais uma variável. `VALUETYPE_RANDOM`: o valor é aleatório e fica em uma faixa de um mínimo a um máximo indicados. `VALUETYPE_ITEM`: o valor vem de um item. `VALUETYPE_ACTOR`: o valor vem de um herói. `VALUETYPE_EVENT`: o valor vem de um evento. `VALUETYPE_MISC`: o valor vem de outra fonte qualquer.

### Parâmetro: `Valor1`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** O significado do valor informado neste parâmetro depende do parâmetro "Tipo de valor".

Em `VALUETYPE_NUMBER`, este é o valor da operação. Em `VALUETYPE_VARIABLE`, esta é a Variable-Id da variável cujo valor deve ser usado. Em `VALUETYPE_VARIABLEINDIRECT`, esta é a Variable-Id da variável cujo valor referencia a variável cujo valor deve ser usado de fato.

Em `VALUETYPE_RANDOM`, este é o mínimo do número aleatório. Em `VALUETYPE_ITEM`, esta é a Item-Id do item do qual um valor deve ser usado. Em `VALUETYPE_ACTOR`, esta é a Actor-Id do herói do qual um valor deve ser usado. Em `VALUETYPE_EVENT`, esta é a Event-Id do evento do qual um valor deve ser usado. Em `VALUETYPE_MISC`, este é um valor que determina o tipo do valor consultado. Para isso são usadas as `constantes de outros valores`:

`MISC_MONEY`: a quantidade de dinheiro do grupo. `MISC_TIMERSECONDSLEFT`: a quantidade de segundos restantes do timer. `MISC_PARTYSIZE`: o tamanho do grupo. `MISC_NUMBEROFSAVES`: a quantidade de vezes que o jogo foi salvo até agora. `MISC_NUMBEROFBATTLES`: a quantidade de combates até agora. `MISC_NUMBEROFVICTORIES`: a quantidade de vitórias até agora. `MISC_NUMBEROFDEFEATS`: a quantidade de derrotas até agora. `MISC_NUMBEROFESCAPES`: a quantidade de fugas até agora. `MISC_MIDITICK`: o tick MIDI atual.

### Parâmetro: `Valor2`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** O significado do valor informado neste parâmetro depende do parâmetro "Tipo de valor".

Em `VALUETYPE_NUMBER`, este parâmetro não é usado e, por isso, deveria ser 0. Em `VALUETYPE_VARIABLE`, este parâmetro não é usado e, por isso, deveria ser 0. Em `VALUETYPE_VARIABLEINDIRECT`, este parâmetro não é usado e, por isso, deveria ser 0.

Em `VALUETYPE_RANDOM`, este é o máximo do número aleatório. Em `VALUETYPE_ITEM`, este é um valor que determina o tipo do valor consultado. Para isso são usadas as `constantes de items`:

`ITEM_OWNED`: a quantidade de itens desse tipo que o grupo possui no momento. `ITEM_EQUIPPED`: a quantidade de itens desse tipo atualmente em uso como equipamento no grupo.

Em `VALUETYPE_ACTOR`, este é um valor que determina o tipo do valor consultado. Para isso são usadas as `constantes de heróis`:

`ACTOR_LEVEL`: o nível atual do herói (equivale a `Actor[].Level`). `ACTOR_EXP`: a experiência atual do herói (equivale a `Actor[].Exp`). `ACTOR_HP`: os pontos de vida atuais do herói (equivale a `Actor[].Hp`). `ACTOR_MP`: os pontos de magia atuais do herói (equivale a `Actor[].Mp`). `ACTOR_MAXHP`: os pontos de vida máximos atuais do herói (não equivale a `Actor[].MaxHpDiff`). `ACTOR_MAXMP`: os pontos de magia máximos atuais do herói (não equivale a `Actor[].MaxMpDiff`). `ACTOR_ATTACK`: os pontos de ataque atuais do herói (não equivale a `Actor[].AttackDiff`). `ACTOR_DEFENSE`: os pontos de defesa atuais do herói (não equivale a `Actor[].DefenseDiff`). `ACTOR_MIND`: os pontos de inteligência atuais do herói (não equivale a `Actor[].MindDiff`). `ACTOR_AGILITY`: os pontos de agilidade atuais do herói (não equivale a `Actor[].AgilityDiff`). `ACTOR_WEAPON`: a arma atual do herói (equivale a `Actor[].Weapon`).

`ACTOR_SHIELD`: o escudo atual do herói (equivale a `Actor[].Shield`). `ACTOR_ARMOR`: a armadura atual do herói (equivale a `Actor[].Armor`). `ACTOR_HELMET`: o elmo atual do herói (equivale a `Actor[].Helmet`). `ACTOR_ACCESSORY`: o acessório atual do herói (equivale a `Actor[].Accessory`).

Em `VALUETYPE_EVENT`, este é um valor que determina o tipo do valor consultado. Para isso são usadas as `constantes de eventos`:

`EVENT_MAPID`: a Map-Id do evento (equivale a `Map.Id`). `EVENT_X`: a coordenada X do evento (equivale a `Event[].X`). `EVENT_Y`: a coordenada Y do evento (equivale a `Event[].Y`). `EVENT_DIRECTION`: a direção para a qual o evento olha (8 = cima, 6 = direita, 4 = esquerda, 2 = baixo; não equivale a `Event[].DirLook`, embora possa ser usado de forma semelhante). `EVENT_SCREENX`: a coordenada X do evento na tela (equivale a `Event[].ScreenX`). `EVENT_SCREENY`: a coordenada Y do evento na tela (equivale a `Event[].ScreenY`).

## Exemplo

```
$
// Definir os valores das primeiras dez variáveis como
Command.ChangeVariableRange(1, 10, OPERATION_SET,
// Acesso totalmente equivalente pelo objeto v[]
for(d[1] = 1; d[1] <= 10; d[1]++)
  v[d[1]] = 123;
next;
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.ChangeVariable()`, `v[]`

MÉTODO

# Método: Command.ClearTimer()

## Sintaxe

```
Command.ClearTimer()
```

## Descrição

Com o método ClearTimer é possível marcar o evento atual como apagado. O evento só volta a aparecer quando o mapa for recarregado.

> **Informação:** Para isso também pode ser usada a propriedade `Exists` do objeto `Event[]`. Com a propriedade Exists também é possível remover a marca de apagado de um evento.

## Exemplo

```
$
// "Apaga" o evento atual, enquanto o mapa não
Command.ClearTimer();
```

## Veja também

`Command`, `Event[].Exists`

MÉTODO

# Método: Command.DealDamage()

## Sintaxe

```
Command.DealDamage(Actor-Id, Angriff, Verteidigung,
Magie, Varianz, [Variável-Id de destino])
```

## Descrição

Com o método DealDamage é possível causar dano a um herói. Aqui é usado o mesmo cálculo da fórmula de dano do sistema de batalha padrão do RPG Maker.

## Lista de parâmetros

### Parâmetro: `Herói-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... Actor.Count
- **Descrição:** A Herói-Id do herói que deve receber o dano. Precisa estar na faixa de 1 a `Actor.Count`. Alternativamente, pode-se usar a constante `STATE_WHOLEPARTY`, para que todo o grupo receba o dano.

### Parâmetro: `Ataque`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** Os pontos de ataque para o cálculo do dano. Deveriam ficar na faixa de 0 a 999.

### Parâmetro: `Defesa`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** O efeito percentual dos pontos de defesa do herói. Esse valor deveria ficar na faixa de 0 a 100.

### Parâmetro: `Magia`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** O dano mágico percentual do ataque. Esse valor deveria ficar na faixa de 0 a 100.

### Parâmetro: `Variância`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A variância (= tolerância) para o cálculo do dano. Esse valor deveria ficar na faixa de 0 a 10.

### Parâmetro: `Variável-Id de destino`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 0
- **Faixa de valores:** >= 0
- **Descrição:** A Variable-Id de uma variável na qual o valor do dano causado deve ser guardado. Se esse valor for 0 ou não for definido, o valor do dano não é guardado.

## Exemplo

```
$
// Causar cerca de 50 pontos de dano ao primeiro herói
// (a defesa atua em 25%, o ataque é puramente físi
//  o valor do dano não é guardado)
Command.DealDamage(1, 50, 25, 0, 5);

// Causar cerca de 100 pontos de dano a todo o grupo
// (a defesa atua em 0%, o ataque é metade físico e
```

```
//  o valor do dano é guardado na variável 10)
Command.DealDamage(STATE_WHOLEPARTY, 100, 0, 50,
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.FullRecovery()`

MÉTODO

# Método: Command.DropEquipment()

## Sintaxe

```
Command.DropEquipment(Herói-Id, Tipo de equipamento)
```

## Descrição

Com o método DropEquipment é possível desequipar um herói.

## Lista de parâmetros

### Parâmetro: `Herói-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... Actor.Count
- **Descrição:** A Herói-Id do herói a ser desequipado. Precisa estar na faixa de 1 a `Actor.Count`. Alternativamente, pode-se usar a constante `STATE_WHOLEPARTY`, para que todo o grupo seja desequipado.

### Parâmetro: `Tipo de equipamento`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 5
- **Descrição:** O tipo de equipamento a ser removido. Para isso são usadas as `constantes de equipamento`:

`EQUIPMENT_ALL`: todo o equipamento é removido. `EQUIPMENT_WEAPON`: a arma é removida. `EQUIPMENT_SHIELD`: o escudo é removido. `EQUIPMENT_ARMOR`: a armadura é removida. `EQUIPMENT_HELMET`: o elmo é removido. `EQUIPMENT_ACCESSORY`: o acessório é removido.

Depois que o herói for desequipado, o item permanece com o grupo.

## Exemplo

```
$
// Desequipar o primeiro herói do item 18 (geralmente
Command.EquipItem(1, 18);
```

```
// Desarmar todo o grupo (= por herói)
Command.DropEquipment(STATE_WHOLEPARTY, EQUIPMENT
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.EquipItem()`

MÉTODO

# Método: Command.EnableEscape()

## Sintaxe

```
Command.EnableEscape(Ativado)
```

## Descrição

Com o método EnableEscape é possível ativar/desativar a fuga no sistema de batalha padrão.

## Lista de parâmetros

### Parâmetro: `Ativado`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 1
- **Descrição:** Indica se a fuga no sistema de batalha padrão é ativada ou desativada. Para isso podem ser usadas as `constantes de (des)ativação`:

`DISABLE`: desativa a fuga no sistema de batalha padrão `ENABLE`: ativa a fuga no sistema de batalha padrão

Em vez de uma dessas constantes, também pode ser usado simplesmente um valor booleano (false = desativar, true = ativar).

## Exemplo

```
$
// Desativar o uso da fuga no sistema de batalha pad
Command.EnableEscape(ENABLE);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.SetEscapePlace()`

MÉTODO

# Método: Command.EnableMenu()

## Sintaxe

```
Command.EnableMenu(Ativado)
```

## Descrição

Com o método EnableMenu é possível ativar/desativar o menu padrão.

## Lista de parâmetros

### Parâmetro: `Ativado`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 1
- **Descrição:** Indica se o menu padrão é ativado ou desativado. Para isso podem ser usadas as `constantes de (des)ativação`:

`DISABLE`: desativa o menu padrão `ENABLE`: ativa o menu padrão

Em vez de uma dessas constantes, também pode ser usado simplesmente um valor booleano (false = desativar, true = ativar).

## Exemplo

```
$
// Ativar o menu padrão
Command.EnableMenu(ENABLE);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.CallMenu()`

MÉTODO

# Método: Command.EnableSaveMenu()

## Sintaxe

```
Command.EnableSaveMenu(Ativado)
```

## Descrição

Com o método EnableSaveMenu é possível ativar/desativar o menu de salvar padrão.

## Lista de parâmetros

### Parâmetro: `Ativado`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 1
- **Descrição:** Indica se o menu de salvar padrão é ativado ou desativado. Para isso podem ser usadas as `constantes de (des)ativação`:

`DISABLE`: desativa o menu de salvar padrão `ENABLE`: ativa o menu de salvar padrão

Em vez de uma dessas constantes, também pode ser usado simplesmente um valor booleano (false = desativar, true = ativar).

## Exemplo

```
$
// Desativar o menu de salvar padrão
Command.EnableSaveMenu(DISABLE);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.CallSaveMenu()`

MÉTODO

# Método: Command.EnableTeleport()

## Sintaxe

```
Command.EnableTeleport(Ativado)
```

## Descrição

Com o método EnableTeleport é possível ativar/desativar o uso de pontos de teletransporte. Os pontos de teletransporte podem ser usados com técnicas do tipo "Teleport".

## Lista de parâmetros

### Parâmetro: `Ativado`

- **Tipo de dado:** `dword`

- **Opcional:** não
- **Faixa de valores:** 0 ... 1
- **Descrição:** Indica se o uso de pontos de teletransporte é ativado ou desativado. Para isso podem ser usadas as `constantes de (des)ativação`:

`DISABLE`: desativa o uso de pontos de teletransporte `ENABLE`: ativa o uso de pontos de teletransporte

Em vez de uma dessas constantes, também pode ser usado simplesmente um valor booleano (false = desativar, true = ativar).

## Exemplo

```
$
// Desativar o uso de pontos de teletransporte
Command.EnableTeleport(DISABLE);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.AddTeleportPlace()`, `Command.RemoveTeleportPlace()`

MÉTODO

# Método: Command.EnterHeroName()

## Sintaxe

```
Command.EnterHeroName(Actor-Id, Eintragsseite,
Aktuellen Namen anzeigen?)
```

## Descrição

Com o método EnterHeroName é possível pedir ao jogador que dê um novo nome a um herói.

> **Informação:** Para usar essa função de forma útil, a função EnterHeroName do RPG_RT precisa primeiro ser corrigida. Isso pode ser feito pelo DestinyPatcher.

> **Informação:** Os caracteres disponíveis para seleção nas páginas "Hiragana" e "Katakana" podem ser alterados pelo DestinyPatcher.

## Lista de parâmetros

### Parâmetro: `Herói-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... Actor.Count

- **Descrição:** A Id do herói a renomear. Esse valor precisa ser uma Actor-Id válida e deveria estar na faixa de 1 a `Actor.Count`.

### Parâmetro: `Página de entrada`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 1
- **Descrição:** A página de entrada exibida por padrão. Para isso são usadas as `constantes de página exibida no EnterHeroName`:

`ENTRYPAGE_HIRAGANA`: a página "Hiragana" é oferecida por padrão. `ENTRYPAGE_KATAKANA`: a página "Katakana" é oferecida por padrão.

### Parâmetro: `Exibir nome atual?`

- **Tipo de dado:** `bool`
- **Opcional:** não
- **Descrição:** Indica se o nome atual é exibido.

False: o campo de entrada começa vazio, por padrão. True: o campo de entrada já vem preenchido, por padrão, com o nome atual do herói.

## Exemplo

```
$
// Deixar o jogador digitar o nome do primeiro herói
// (a página padrão é Hiragana e o nome padrão é
Command.EnterHeroName(1, ENTRYPAGE_HIRAGANA, fals
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`

MÉTODO

# Método: Command.EnterKey()

## Sintaxe

```
Command.EnterKey(Variable-Id, Warten bis Taste
gedrückt wurde, Richtungstasten, Auswahl, Abbrechen /
Menü)
```

## Descrição

Com o método EnterKey é possível descobrir a tecla atualmente pressionada, dentre uma pré-seleção.

> **Informação:** As teclas são consultadas por este método de acordo com o RPG_RT. Por isso, várias teclas, por exemplo, podem ter o significado "para a direita". A configuração dessas teclas pode ser alterada no RPG_RT pelo DestinyPatcher.

> **Informação:** O tempo de espera só é "executado" pelo RPG_RT ao final do DestinyScript, e pode ser sobrescrito por outros comandos do RPG Maker chamados pelo objeto `Command`.

> **Informação:** A consulta de teclas pelo método EnterKey é bem imprecisa, já que, por exemplo, as teclas de seta não podem ser tratadas separadamente. Para determinar com precisão o status de uma tecla específica, deve-se usar o método `GetKeyState` do objeto `Keyboard`.

## Lista de parâmetros

### Parâmetro: `Variável-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** A Variable-Id de uma variável na qual o número da tecla pressionada deve ser guardado. O valor da Variable-Id precisa ser maior/igual a 1. Ao final, essa variável pode conter os seguintes valores:

0: nenhuma tecla está pressionada (só ocorre se "Aguardar até uma tecla ser pressionada" for false). 1: Baixo 2: Esquerda 3: Direita 4: Cima 5: Confirmar 6: Cancelar/Menu

### Parâmetro: `Aguardar até uma tecla ser pressionada`

- **Tipo de dado:** `bool`
- **Opcional:** não
- **Descrição:** Indica se a continuação do EventScript é interrompida até que uma tecla seja pressionada.

False: não se espera. True: espera-se.

### Parâmetro: `Teclas direcionais`

- **Tipo de dado:** `bool`
- **Opcional:** não
- **Descrição:** Indica se as teclas direcionais são verificadas.

### Parâmetro: `Seleção`

- **Tipo de dado:** `bool`
- **Opcional:** nein Beschreibung: Gibt an, ob die Auswahltaste geprüft wird.

### Parâmetro: `Cancelar / Menu`

- **Tipo de dado:** `bool`
- **Opcional:** não
- **Descrição:** Indica se a tecla de cancelar/menu é verificada.

## Exemplo

O exemplo a seguir precisa estar sozinho em um comentário do RPG Maker, para que nenhum outro método do `Command` sobrescreva o tempo de espera (e, assim, o tempo de espera seja de fato cumprido):

```
$
// Descobrir a última tecla pressionada e guardar em v
Command.EnterKey(1, true, true, true, false);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Keyboard`, `Keyboard.GetKeyState()`

MÉTODO

# Método: Command.EnterLeaveVehicle()

## Sintaxe

```
Command.EnterLeaveVehicle()
```

## Descrição

Com o método EnterLeaveVehicle é possível fazer o herói entrar no veículo mais próximo, ou sair do veículo que ele estiver usando no momento.

## Exemplo

```
$
// Colocar a aeronave na mesma posição que o herói
Command.SetVehiclePlace(AIRSHIP, Map.Id, Map.Hero

// Fazer o herói entrar no veículo mais próximo (isto
Command.EnterLeaveVehicle();
```

## Veja também

`Command`, `Command.SetVehiclePlace()`, `Map.Id`, `Map.HeroX`, `Map.HeroY`

MÉTODO

# Método: Command.EquipItem()

## Sintaxe

```
Command.EquipItem(Herói-Id, Item-Id)
```

## Descrição

Com o método EquipItem é possível equipar um herói com uma peça de equipamento.

## Lista de parâmetros

### Parâmetro: `Herói-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... Actor.Count
- **Descrição:** A Actor-Id do herói a ser equipado com a peça de equipamento. Precisa estar na faixa de 1 a `Actor.Count`. Alternativamente, pode-se usar a constante `STATE_WHOLEPARTY`, para que todo o grupo seja equipado com essa peça.

### Parâmetro: `Item-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... Item.Count
- **Descrição:** A Item-Id da peça de equipamento com a qual o herói deve ser equipado. Se o grupo não tiver esse item, ele é adicionado ao grupo. Se o herói já estiver equipado com um item do mesmo tipo, esse item é desequipado, sem ser descartado. A Item-Id precisa estar na faixa de 1 a `Item.Count` e ser uma peça de equipamento válida.

## Exemplo

```
$
// Equipar o primeiro herói com o item 18 (geralmente
Command.EquipItem(1, 18);

// Desarmar todo o grupo (= por herói)
Command.DropEquipment(STATE_WHOLEPARTY, EQUIPMENT
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.DropEquipment()`

MÉTODO

# Método: Command.ErasePicture()

## Sintaxe

```
Command.ErasePicture(Picture-Id)
```

## Descrição

Com o método ErasePicture é possível apagar uma Picture.

> **Informação:** Ao apagar uma Picture, o AuroraSheet é apagado.

## Lista de parâmetros

### Parâmetro: `Picture-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... Picture.Count
- **Descrição:** A Picture-Id da Picture a apagar. Esse valor precisa ser uma Picture-Id válida e, portanto, fica na faixa de 1 a `Picture.Count`.

## Exemplo

```
$
// Carregar o gráfico de nuvem do RTP como Picture 1 e exi
Command.ShowPicture(1, "CLOUD", 160, 120);
// Apagar a Picture logo em seguida
Command.ErasePicture(1);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.ShowPicture()`, `Command.MovePicture()`, `Picture`

MÉTODO

# Método: Command.EraseScreen()

## Sintaxe

```
Command.EraseScreen(Transição)
```

## Descrição

Com o método EraseScreen é possível apagar ("esconder") a tela, deixando-a preta.

## Lista de parâmetros

### Parâmetro: `Transição`

- **Tipo de dado:** `dword`
- **Opcional:** não

- **Faixa de valores:** 0 ... 19
- **Descrição:** A transição a ser usada (do tipo Erase). Para isso são usadas as `constantes de erase`:

`ERASE_FADEOUT`: a tela desaparece com fade-out `ERASE_RANDOMBLOCKS`: a tela se preenche aleatoriamente com blocos `ERASE_RANDOMBLOCKSUP`: a tela se preenche aleatoriamente com blocos, de baixo para cima `ERASE_RANDOMBLOCKSDOWN`: a tela se preenche aleatoriamente com blocos, de cima para baixo `ERASE_BLINDCLOSE`: a tela se preenche com listras horizontais `ERASE_STRIPEUPDOWN`: a tela é preenchida simultaneamente de cima e de baixo, cada uma com uma faixa preta `ERASE_STRIPELEFTRIGHT`: a tela é preenchida simultaneamente da esquerda e da direita, cada uma com uma faixa preta `ERASE_OUTSIDEINSIDE`: a tela se preenche em forma de retângulo, de fora para dentro `ERASE_INSIDEOUTSIDE`: a tela se preenche em forma de retângulo, de dentro para fora `ERASE_SCROLLUP`: a tela é deslocada para fora, para cima `ERASE_SCROLLDOWN`: a tela é deslocada para fora, para baixo `ERASE_SCROLLLEFT`: a tela é deslocada para fora, para a esquerda `ERASE_SCROLLRIGHT`: a tela é deslocada para fora, para a direita `ERASE_DIVISIONUPDOWN`: a tela se divide horizontalmente ao meio, e as metades de cima e de baixo se afastam simultaneamente `ERASE_DIVISIONLEFTRIGHT`: a tela se divide verticalmente ao meio, e as metades da esquerda e da direita se afastam simultaneamente `ERASE_DIVISION`: a tela se divide horizontal e verticalmente ao meio, e os quatro retângulos se afastam simultaneamente em direção ao seu canto da tela `ERASE_ZOOMIN`: a tela é fortemente ampliada em direção ao herói `ERASE_MOSAIC`: a tela desaparece através de um mosaico cada vez mais grosso `ERASE_ROSTERSCROLL`: a tela borra de forma ondulada `ERASE_INSTANT`: a tela fica preta imediatamente, sem transição

## Exemplo

```
$
// Apagar a tela de fora para dentro
Command.EraseScreen(ERASE_OUTSIDEINSIDE);
// Teletransportar o herói para o canto inferior direito
Command.Teleport(Map.Id, Map.Width - 1, Map.Heigh
// Exibir a tela de dentro para fora
Command.ShowScreen(SHOW_INSIDEOUTSIDE);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.Teleport()`, `Command.ShowScreen()`, `Map.Id`, `Map.HeroX`, `Map.HeroY`

MÉTODO

# Método: Command.FadeOutMusic()

## Sintaxe

```
Command.FadeOutMusic(Tempo (velocidade) de fade-out)
```

## Descrição

Com o método FadeOutMusic é possível apagar (fade-out) a música de fundo.

## Lista de parâmetros

### Parâmetro: `Tempo de fade-out`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** O tempo de fade-out da música, em milissegundos. (Com o valor 1000, a música só fica muda depois de um segundo; com 2000, depois de dois segundos, e assim por diante.)

## Exemplo

```
$
// Tocar a música de fundo "ANGER"
Command.PlayMusic("ANGER");
// Apagar a música de fundo em um segundo
Command.FadeOutMusic(1000);
```

## Veja também

`Command`, `Command.PlayMusic()`, `Command.StopMusic()`

MÉTODO

# Método: Command.FixScreen()

## Sintaxe

```
Command.FixScreen()
```

## Descrição

Com o método FixScreen, a tela é fixada na posição atual e, com isso, deixa de seguir o herói.

## Exemplo

```
$
// Fixar a tela na posição atual
Command.FixScreen();

// ...

// Remover a fixação da tela
Command.ReleaseScreen();
```

## Veja também

`Command`, `Command.ReleaseScreen()`

MÉTODO

# Método: Command.FlashEvent()

## Sintaxe

```
Command.FlashEvent(Event-Id, Rot, Grün, Blau, Stärke,
Dauer, Warten)
```

## Descrição

Com o método FlashEvent é possível fazer um evento piscar em uma determinada cor.

> **Informação:** O tempo de espera só é "executado" pelo RPG_RT ao final do DestinyScript, e pode ser sobrescrito por outros comandos do RPG Maker chamados pelo objeto `Command`.

> **Informação:** O valor informado aqui na faixa de 0 ... 31 pode ser convertido aproximadamente para a faixa do respectivo componente de cor RGB multiplicando-o pelo fator 8 (ou dividindo-o, no sentido inverso). Para representar, por exemplo, cinza (#C0C0C0, ou seja, 192 de vermelho - 192 de verde - 192 de azul), deve-se definir aqui o valor (192 / 8) = 24 para vermelho, verde e azul.

## Lista de parâmetros

### Parâmetro: `Evento-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** A Event-Id do respectivo evento que deve piscar. Esse valor precisa ser uma Event-Id válida ou um evento especial. Aqui também podem ser usadas as constantes de `eventos especiais`:

`HERO`: o evento do herói `BOAT`: o evento do barco `SHIP`: o evento do navio `AIRSHIP`: o evento da aeronave `THIS`: o evento atual

### Parâmetro: `Vermelho`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A proporção de vermelho da cor durante o piscar. Esse valor deveria ficar na faixa de 0 a 31. Com valores maiores/menores, ocorrem erros gráficos divertidos.

O valor 0 significa que a proporção de vermelho da cor não é usada (equivale a 0 no sistema RGB). O valor 31 significa que a proporção de vermelho da cor é usada totalmente (equivale aproximadamente a 255 no sistema RGB).

### Parâmetro: `Verde`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A proporção de verde da cor durante o piscar. Esse valor deveria ficar na faixa de 0 a 31. Com valores maiores/menores, ocorrem erros gráficos divertidos.

O valor 0 significa que a proporção de verde da cor não é usada (equivale a 0 no sistema RGB). O valor 31 significa que a proporção de verde da cor é usada totalmente (equivale aproximadamente a 255 no sistema RGB).

### Parâmetro: `Azul`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A proporção de azul da cor durante o piscar. Esse valor deveria ficar na faixa de 0 a 31. Com valores maiores/menores, ocorrem erros gráficos divertidos.

O valor 0 significa que a proporção de azul da cor não é usada (equivale a 0 no sistema RGB). O valor 31 significa que a proporção de azul da cor é usada totalmente (equivale aproximadamente a 255 no sistema RGB).

### Parâmetro: `Intensidade`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A intensidade do piscar. Isso equivale à opacidade mais alta (aqui, equivale à opacidade da cor) durante o piscar. Esse valor deveria ficar na faixa de 0 a 31. Com valores maiores/menores, ocorrem erros gráficos divertidos.

O valor 0 significa que a cor não é exibida durante o piscar. O valor 16 significa que a cor é exibida, no seu ponto mais forte, aproximadamente meio-transparente durante o piscar. O valor 31 significa que a cor é exibida, no seu ponto mais forte, totalmente opaca durante o piscar.

### Parâmetro: `Duração`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 0
- **Descrição:** A duração, em décimos de segundo, até a alteração de cor se completar.

### Parâmetro: `Aguardar`

- **Tipo de dado:** `bool`
- **Opcional:** não
- **Descrição:** Indica se a continuação do EventScript é interrompida até que a duração indicada se esgote.

False: não se espera. True: espera-se.

## Exemplo

O exemplo a seguir precisa estar sozinho em um comentário do RPG Maker, para que nenhum outro método do `Command` sobrescreva o tempo de espera (e, assim, o tempo de espera seja de fato cumprido):

```
$
// Fazer o herói piscar em verde (sem transparência
Command.FlashEvent(HERO, 0, 31, 0, 31, 10, true);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`

MÉTODO

# Método: Command.FlashScreen()

## Sintaxe

```
Command.FlashScreen(Rot, Grün, Blau, Stärke, Dauer,
Warten)
```

## Descrição

Com o método FlashScreen é possível fazer a tela piscar em uma determinada cor.

> **Informação:** O tempo de espera só é "executado" pelo RPG_RT ao final do DestinyScript, e pode ser sobrescrito por outros comandos do RPG Maker chamados pelo objeto `Command`.

> **Informação:** O valor informado aqui na faixa de 0 ... 31 pode ser convertido aproximadamente para a faixa do respectivo componente de cor RGB multiplicando-o pelo fator 8 (ou dividindo-o, no sentido inverso). Para representar, por exemplo, cinza (#C0C0C0, ou seja, 192 de vermelho - 192 de verde - 192 de azul), deve-se definir aqui o valor (192 / 8) = 24 para vermelho, verde e azul.

## Lista de parâmetros

### Parâmetro: `Vermelho`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A proporção de vermelho da cor durante o piscar. Esse valor deveria ficar na faixa de 0 a 31. Com valores maiores/menores, ocorrem erros gráficos divertidos.

O valor 0 significa que a proporção de vermelho da cor não é usada (equivale a 0 no sistema RGB). O valor 31 significa que a proporção de vermelho da cor é usada totalmente (equivale aproximadamente a 255 no sistema RGB).

### Parâmetro: `Verde`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A proporção de verde da cor durante o piscar. Esse valor deveria ficar na faixa de 0 a 31. Com valores maiores/menores, ocorrem erros gráficos divertidos.

O valor 0 significa que a proporção de verde da cor não é usada (equivale a 0 no sistema RGB). O valor 31 significa que a proporção de verde da cor é usada totalmente (equivale aproximadamente a 255 no sistema RGB).

### Parâmetro: `Azul`

- **Tipo de dado:** `dword`
- **Opcional:** não

- **Descrição:** A proporção de azul da cor durante o piscar. Esse valor deveria ficar na faixa de 0 a 31. Com valores maiores/menores, ocorrem erros gráficos divertidos.

O valor 0 significa que a proporção de azul da cor não é usada (equivale a 0 no sistema RGB).

O valor 31 significa que a proporção de azul da cor é usada totalmente (equivale aproximadamente a 255 no sistema RGB).

### Parâmetro: `Intensidade`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A intensidade do piscar. Isso equivale à opacidade mais alta (aqui, equivale à opacidade da cor) durante o piscar. Esse valor deveria ficar na faixa de 0 a 31. Com valores maiores/menores, ocorrem erros gráficos divertidos.

O valor 0 significa que a cor não é exibida durante o piscar. O valor 16 significa que a cor é exibida, no seu ponto mais forte, aproximadamente meio-transparente durante o piscar. O valor 31 significa que a cor é exibida, no seu ponto mais forte, totalmente opaca durante o piscar.

### Parâmetro: `Duração`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 0
- **Descrição:** A duração, em décimos de segundo, até a alteração de cor se completar.

### Parâmetro: `Aguardar`

- **Tipo de dado:** `bool`
- **Opcional:** não
- **Descrição:** Indica se a continuação do EventScript é interrompida até que a duração indicada se esgote.

False: não se espera. True: espera-se.

## Exemplo

O exemplo a seguir precisa estar sozinho em um comentário do RPG Maker, para que nenhum outro método do `Command` sobrescreva o tempo de espera (e, assim, o tempo de espera seja de fato cumprido):

```
$
// Fazer a tela piscar em vermelho (sem transparê
Command.FlashScreen(31, 0, 0, 31, 10, true);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`

MÉTODO

# Método: Command.FullRecovery()

## Sintaxe

```
Command.FullRecovery(Herói-Id)
```

## Descrição

Com o método FullRecovery é possível curar completamente um herói. Isso define os pontos de vida e de magia no respectivo máximo. Além disso, todas as condições são curadas.

## Lista de parâmetros

### Parâmetro: `Herói-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... Actor.Count
- **Descrição:** A Actor-Id do herói a ser curado completamente. Precisa estar na faixa de 1 a `Actor.Count`. Alternativamente, pode-se usar a constante `STATE_WHOLEPARTY`, para que todo o grupo seja curado.

## Exemplo

```
$
// Curar todo o grupo
Command.FullRecovery(STATE_WHOLEPARTY);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.DealDamage()`

MÉTODO

# Método: Command.GameOver()

## Sintaxe

```
Command.GameOver()
```

## Descrição

Com o método GameOver é possível encerrar a partida atual pela tela de Game Over. A partida é encerrada imediatamente e o jogo entra no modo Game Over; o DestinyScript, porém, continua rodando. Depois da tela de

Game Over, normalmente vem a tela de título, de onde o jogador pode iniciar um novo jogo.

## Exemplo

```
$
// Encerrar a partida atual pela tela de Game Over
Command.GameOver();
```

## Veja também

`Command`, `Command.TitleScreen()`

MÉTODO

# Método: Command.GetEventId()

## Sintaxe

```
Command.GetEventId(Variável-Id, X, Y)
```

## Descrição

Com o método GetEventId é possível descobrir a Event-Id de um evento em uma determinada posição do mapa.

## Lista de parâmetros

### Parâmetro: `Variável-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** A Variable-Id da variável na qual a Event-Id deve ser guardada. Esse valor precisa ser maior/igual a 1.

### Parâmetro: `X`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... Map.Width - 1
- **Descrição:** A coordenada X onde o evento precisa estar. Esse valor deveria ficar na faixa de 0 ... `Map.Width` - 1.

### Parâmetro: `Y`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... Map.Height - 1
- **Descrição:** A coordenada Y onde o evento precisa estar. Esse valor deveria ficar na faixa de 0 ... `Map.Height` - 1.

## Exemplo

```
$
// A Event-Id do evento no canto superior esquerdo
Command.GetEventId(1, 0, 0);
```

## Observações adicionais

O código a seguir é equivalente ao código acima, mas usa o objeto `MapEvent[]` para a busca.

```
$
// A Event-Id do evento no canto superior esquerdo
v[1] = 0;
for(d[1] = 0; d[1] < MapEvent.Count; d[1]++)
  if(MapEvent[d[1]].X == 0 && MapEvent[d[1]].Y ==
    v[1] = d[1];
    break;
  endif;
next;
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `MapEvent`, `Map.Id`, `Map.HeroX`, `Map.HeroY`

MÉTODO

# Método: Command.GetTerrainId()

## Sintaxe

```
Command.GetTerrainId(Variável-Id, X, Y)
```

## Descrição

Com o método GetTerrainId é possível descobrir a Terrain-Id de um campo do mapa.

## Lista de parâmetros

### Parâmetro: `Variável-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** A Variable-Id da variável na qual a Terrain-Id deve ser guardada. Esse valor precisa ser maior/igual a 1.

### Parâmetro: `X`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... Map.Width - 1
- **Descrição:** A coordenada X do campo a consultar. Esse valor deveria ficar na faixa de 0 ... `Map.Width` - 1.

### Parâmetro: `Y`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... Map.Height - 1
- **Descrição:** A coordenada Y do campo a consultar. Esse valor deveria ficar na faixa de 0 ... `Map.Height` - 1.

## Exemplo

```
$
// A Terrain-Id do campo no canto superior esquerdo
Command.GetTerrainId(1, 0, 0);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Map.Id`, `Map.HeroX`, `Map.HeroY`, `Map.Lower[]`, `Map.Upper[]`

MÉTODO

# Método: Command.GoToLabel()

## Sintaxe

```
Command.GoToLabel(Nº do rótulo de salto)
```

## Descrição

Com o método GoToLabel é possível fazer o ponteiro de comando do evento atual pular para um determinado rótulo. O efeito só se manifesta ao final do DestinyScript.

> **Informação:** O rótulo usado precisa estar definido no EventScript (pelo RPG Maker). Se o rótulo indicado não existir, não ocorre erro por parte do RPG_RT.exe (1.0.7). Mas isso não deve ser considerado um comportamento garantido.

## Lista de parâmetros

### Parâmetro: `Nº do rótulo de salto`

- **Tipo de dado:** `dword`

- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** O número do rótulo. Ele deveria existir no EventScript atual.

## Exemplo

```
$
// Pula para o primeiro rótulo no evento atual
Command.GoToLabel(1);
```

## Veja também

`Command`

MÉTODO

# Método: Command.GoToMemorizedPlace()

## Sintaxe

```
Command.GoToMemorizedPlace(Variable für Map-Id,
Variable für X, Variable für Y)
```

## Descrição

Com o método GoToMemorizedPlace é possível teletransportar o herói para uma determinada posição, usando variáveis.

> **Informação:** Os valores das variáveis precisam obrigatoriamente ser válidos, caso contrário o RPG_RT trava.

## Lista de parâmetros

### Parâmetro: `Variável para Mapa-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** A Variable-Id da variável da qual a Map-Id deve ser carregada. Esse valor precisa ser uma Variable-Id válida e precisa ser maior/igual a 1. O valor dentro da variável precisa ser uma Map-Id válida, caso contrário o jogo trava.

### Parâmetro: `Variável para X`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** A Variable-Id da variável da qual a coordenada X deve ser carregada. Esse valor precisa ser uma Variable-Id válida e precisa ser maior/igual a 1.

### Parâmetro: `Variável para Y`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** A Variable-Id da variável da qual a coordenada Y deve ser carregada. Esse valor precisa ser uma Variable-Id válida e precisa ser maior/igual a 1.

## Exemplo

```
$
// Definir as coordenadas para este exemplo
v[1] = Map.Id;
v[2] = 0;
v[3] = 0;

// Teletransportar o herói para o canto superior esquer
Command.GoToMemorizedPlace(1, 2, 3);
```

## Observações adicionais

Também é possível usar o método `Teleport()` para isso.

```
$
// Teletransportar o herói para o canto superior esquer
Command.GoToMemorizedPlace(Map.Id, 0, 0);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.Teleport()`, `Command.MemorizePlace()`, `Map.Id`, `Map.HeroX`, `Map.HeroY`

MÉTODO

# Método: Command.MemorizeMusic()

## Sintaxe

```
Command.MemorizeMusic()
```

## Descrição

Com o método MemorizeMusic é possível guardar temporariamente a música de fundo atual. Mais tarde, a música de fundo guardada pode ser tocada novamente pelo método `PlayMemorizedMusic`.

## Exemplo

```
$
// Tocar a música de fundo "ANGER"
```

```
Command.PlayMusic("ANGER");
// Guardar temporariamente as configurações atuais da
Command.MemorizeMusic();
// Parar a música de fundo atual
Command.StopMusic();
// Tocar novamente a música de fundo guardada
Command.PlayMemorizedMusic();
```

## Veja também

`Command`, `Command.PlayMusic()`, `Command.StopMusic()`, `Command.PlayMemorizedMusic()`

MÉTODO

# Método: Command.MemorizePlace()

## Sintaxe

```
Command.MemorizePlace(Variable für Map-Id, Variable
für X, Variable für Y)
```

## Descrição

Com o método MemorizePlace é possível guardar as coordenadas atuais do herói em variáveis.

## Lista de parâmetros

### Parâmetro: `Variável para Mapa-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** A Variable-Id da variável na qual a Map-Id deve ser guardada. Esse valor precisa ser uma Variable-Id válida e precisa ser maior/igual a 1.

### Parâmetro: `Variável para X`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** A Variable-Id da variável na qual a coordenada X deve ser guardada. Esse valor precisa ser uma Variable-Id válida e precisa ser maior/igual a 1.

### Parâmetro: `Variável para Y`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** A Variable-Id da variável na qual a coordenada Y deve ser guardada. Esse valor precisa ser uma Variable-Id válida e precisa ser maior/igual a 1.

## Exemplo

```
$
// Guardar as coordenadas atuais do herói nas variá
Command.MemorizePlace(1, 2, 3);
```

## Observações adicionais

Também é possível usar o objeto `Map` para isso.

```
$
// Guardar as coordenadas atuais do herói nas variá
v[1] = Map.Id;
v[2] = Map.HeroX;
v[3] = Map.HeroY;
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.Teleport()`, `Command.GoToMemorizedPlace()`, `Map.Id`, `Map.HeroX`, `Map.HeroY`

MÉTODO

# Método: Command.MoveAll()

## Sintaxe

```
Command.MoveAll()
```

## Descrição

Com o método MoveAll, espera-se até que todos os passos de movimento a executar tenham sido processados. Os passos de movimento precisam ter sido executados antes em um comando MoveEvent (EventScript).

> **Informação:** O tempo de espera só é "executado" pelo RPG_RT ao final do DestinyScript, e pode ser sobrescrito por outros comandos do RPG Maker chamados pelo objeto `Command`.

## Exemplo

O exemplo a seguir precisa estar sozinho em um comentário do RPG Maker, para que nenhum outro método do `Command` sobrescreva o tempo de espera (e, assim, o tempo de espera seja de fato cumprido); além disso, antes é preciso que um comando MoveEvent (EventScript) já tenha sido executado, para que se note algum efeito:

```
$
// Espera que o(s) MoveEvent(s) indicado(s) antes
Command.MoveAll();
```

## Veja também

`Command`, `Command.MoveEvent()`, `Command.StopAll()`

MÉTODO

# Método: Command.MoveEvent()

## Sintaxe

```
Command.MoveEvent(Event-Id, Schritt, [Frequenz,
[Bewegungen wiederholen, [Ignorieren falls Bewegung
unmöglich]]])
```

## Descrição

Com o método MoveEvent é possível mover um evento em exatamente um passo. Esse passo substitui a lista previamente definida de um comando MoveEvent (EventScript).

> **Informação:** O movimento só é executado depois do DestinyScript, e pode ser substituído por novas chamadas de Command.MoveEvent().

> **Informação:** Como atualmente não é possível indicar mais de um passo no comando MoveEvent, recomenda-se, para vários passos, usar o comando MoveEvent do EventScript.

## Lista de parâmetros

### Parâmetro: `Evento-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** Event-Id válida
- **Descrição:** A Event-Id do evento a mover. Esse valor precisa ser uma Event-Id válida ou um evento especial. Aqui também podem ser usadas as constantes de `eventos especiais`:

`HERO`: o evento do herói `BOAT`: o evento do barco `SHIP`: o evento do navio `AIRSHIP`: o evento da aeronave `THIS`: o evento atual

### Parâmetro: `Passo`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 31; 36 ... 41
- **Descrição:** O passo a executar. Aqui são usadas as constantes `MoveCommand`:

`MC_STEPUP`: um passo para cima `MC_STEPRIGHT`: um passo para a direita `MC_STEPDOWN`: um passo para baixo `MC_STEPLEFT`: um passo para a esquerda `MC_STEPRIGHTUP`: um passo para a direita-cima `MC_STEPRIGHTDOWN`: um passo para a direita-baixo `MC_STEPLEFTDOWN`: um passo para a esquerda-baixo `MC_STEPLEFTUP`: um passo para a direita-cima `MC_STEPRANDOM`: um passo em uma direção aleatória `MC_STEPTOWARDHERO`: um passo em direção ao herói `MC_STEPAWAYFROMHERO`: um passo para longe do herói `MC_STEPFORWARD`: um passo na direção atual em que olha `MC_LOOKUP`: olhar para cima `MC_LOOKRIGHT`: olhar para a direita `MC_LOOKDOWN`: olhar para baixo `MC_LOOKLEFT`:

olhar para a esquerda `MC_TURNRIGHT`: girar 90° para a direita `MC_TURNLEFT`: girar 90° para a esquerda `MC_TURN180`: girar 180° `MC_RIGHTLEFTTURN`: girar 90° aleatoriamente para a esquerda ou direita `MC_LOOKRANDOM`: olhar em uma direção aleatória `MC_LOOKTOWARDHERO`: olhar em direção ao herói `MC_LOOKAWAYFROMHERO`: olhar para longe do herói `MC_WAIT`: aguardar um instante `MC_STARTJUMPING`: começar a pular `MC_STOPJUMPING`: parar de pular `MC_FIXDIRECTION`: fixar a direção em que olha `MC_RELEASEDIRECTION`: liberar a direção em que olha `MC_INCREASESPEED`: aumentar a velocidade de movimento `MC_DECREASESPEED`: diminuir a velocidade de movimento `MC_INCREASEFREQUENCY`: aumentar a frequência de movimento `MC_DECREASEFREQUENCY`: diminuir a frequência de movimento `MC_STARTPHASING`: ativar o movimento sem colisão `MC_STOPPHASING`: desativar o movimento sem colisão `MC_FIXANIMATION`: desativar a animação de movimento `MC_RELEASEANIMATION`: ativar a animação de movimento `MC_INCREASETRANSPARENCY`: aumentar a transparência `MC_DECREASETRANSPARENCY`: diminuir a transparência

As constantes `MC_SETSWITCHON`, `MC_SETSWITCHOFF`, `MC_SETCHAR` e `MC_PLAYSOUND` não podem ser usadas aqui, por motivos técnicos.

### Parâmetro: `Frequência`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 8
- **Faixa de valores:** 1...8
- **Descrição:** A frequência de movimento com que os passos são processados. Esse valor precisa estar na faixa de 1 a 8.

### Parâmetro: `Repetir movimentos`

- **Tipo de dado:** `bool`
- **Opcional:** sim
- **Valor padrão:** false
- **Descrição:** Indica se os movimentos são repetidos em laço, até que um novo comando MoveCommand seja emitido.

False: os movimentos não são repetidos. True: os movimentos são repetidos.

### Parâmetro: `Ignorar se o movimento for impossível`

- **Tipo de dado:** `bool`
- **Opcional:** sim
- **Valor padrão:** true
- **Descrição:** Indica se deve ser ignorado quando um movimento indicado não é possível.

False: espera-se até que o movimento seja possível.

Atenção: isso pode fazer com que o respectivo evento deixe de se mover sozinho. True: o movimento é executado, se possível. Se não for possível, o movimento é descartado.

## Exemplo

O exemplo a seguir precisa estar sozinho em um comentário do RPG Maker, para que nenhum outro método do `Command` sobrescreva o tempo de espera (e, assim, o tempo de espera seja de fato cumprido); além disso, antes é preciso que um comando MoveEvent (EventScript) já tenha sido executado, para que se note algum efeito:

```
$
// Move o herói 1 passo para cima.
Command.MoveEvent(HERO, MC_STEPUP);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.MoveAll()`, `Command.StopAll()`, `Event[].MoveRoute`

MÉTODO

# Método: Command.MovePicture()

## Sintaxe

```
Command.MovePicture(Picture-Id, X, Y, [Vergrößerung,
[Transparenz, [Rot, [Grün, [Blau, [Chrominanz,
[Aktion, [Intensidade da ação, Duração, Aguardar]]]]]]]])
```

## Descrição

Com o método MovePicture é possível mover uma Picture pela tela.

> **Informação:** O tempo de espera só é "executado" pelo RPG_RT ao final do DestinyScript, e pode ser sobrescrito por outros comandos do RPG Maker chamados pelo objeto `Command`.

## Lista de parâmetros

### Parâmetro: `Picture-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... Picture.Count
- **Descrição:** A Picture-Id da Picture a mover. Esse valor precisa ser uma Picture-Id válida e, portanto, fica na faixa de 1 a `Picture.Count`.

### Parâmetro: `X`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A coordenada X de destino da Picture. Ela se refere ao centro da Picture.

### Parâmetro: `Y`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A coordenada Y de destino da Picture. Ela se refere ao centro da Picture.

### Parâmetro: `Ampliação`

- **Tipo de dado:** `dword`
- **Opcional:** sim

- **Valor padrão:** 100
- **Faixa de valores:** >= 1
- **Descrição:** O fator de ampliação de destino da Picture, em porcentagem. O valor padrão 100 indica que a Picture é exibida normalmente.

### Parâmetro: `Transparência`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** A transparência de destino da Picture, em porcentagem. Esse valor deveria ficar na faixa de 0 a 100. Com valores maiores que 100, ocorrem erros de cálculo divertidos.

### Parâmetro: `Vermelho`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 100
- **Descrição:** A coloração vermelha de destino da Picture, em porcentagem. Esse valor deveria ficar na faixa de 0 a 200.

O valor 0 significa que a Picture não apresenta a cor vermelha. O valor 100 significa que a Picture exibe o vermelho normalmente. O valor 200 significa que a Picture é exibida com tons de vermelho especialmente fortes.

### Parâmetro: `Verde`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 100
- **Descrição:** A coloração verde de destino da Picture, em porcentagem. Esse valor deveria ficar na faixa de 0 a 200.

O valor 0 significa que a Picture não apresenta a cor verde. O valor 100 significa que a Picture exibe o verde normalmente. O valor 200 significa que a Picture é exibida com tons de verde especialmente fortes.

### Parâmetro: `Azul`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 100
- **Descrição:** A coloração azul de destino da Picture, em porcentagem. Esse valor deveria ficar na faixa de 0 a 200.

O valor 0 significa que a Picture não apresenta a cor azul. O valor 100 significa que a Picture exibe o azul normalmente. O valor 200 significa que a Picture é exibida com tons de azul especialmente fortes.

### Parâmetro: `Crominância`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 100
- **Descrição:** A crominância (saturação de cor) de destino da Picture, em porcentagem. Esse valor deveria ficar na faixa de 0 a 200.

O valor 0 significa que a Picture é exibida dessaturada. O valor 100 significa que a Picture é exibida normalmente. O valor 200 significa que a Picture é exibida com diferenças de cor especialmente grandes.

### Parâmetro: `Ação`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 0
- **Faixa de valores:** 0 ... 2
- **Descrição:** A ação de destino da Picture. Para isso são usadas as `constantes de ação`:

`ACTION_NONE`: a Picture não realiza nenhuma ação. `ACTION_ROTATION`: a Picture gira em torno do próprio eixo. `ACTION_RIPPLE`: a Picture realiza um efeito de onda.

### Parâmetro: `Intensidade da ação`

- **Tipo de dado:** `dword`
- **Opcional:** ja Standardwert: 0
- **Descrição:** A intensidade de ação de destino da Picture. O significado desse valor depende da ação escolhida para a Picture. Na ação `ACTION_NONE`, o valor deste parâmetro é ignorado.

### Parâmetro: `Duração`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 0
- **Descrição:** A duração, em décimos de segundo, até o movimento se completar.

### Parâmetro: `Aguardar`

- **Tipo de dado:** `bool`
- **Opcional:** não
- **Descrição:** Indica se a continuação do EventScript é interrompida até que a duração indicada se esgote.

False: não se espera. True: espera-se.

## Exemplo

```
$
// Carregar o gráfico de nuvem do RTP como Picture 1 e exi
Command.ShowPicture(1, "CLOUD", 160, 120);
// Mover a Picture em um segundo para o canto inferior
Command.MovePicture(1, 320, 240, 100, 0, 100, 100
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.ShowPicture()`, `Command.ErasePicture()`, `Picture`

MÉTODO

# Método: Command.PlayMemorizedMusic()

## Sintaxe

```
Command.PlayMemorizedMusic()
```

## Descrição

Com o método PlayMemorizedMusic é possível tocar uma música de fundo previamente guardada. A música de fundo precisa ter sido guardada antes pelo método `MemorizeMusic`. Se nenhuma música tiver sido guardada antes por MemorizeMusic, a música simplesmente termina ao chamar este método.

## Exemplo

```
$
// Tocar a música de fundo "ANGER"
Command.PlayMusic("ANGER");
// Guardar temporariamente as configurações atuais da
Command.MemorizeMusic();
// Parar a música de fundo atual
Command.StopMusic();
// Tocar novamente a música de fundo guardada
Command.PlayMemorizedMusic();
```

## Veja também

`Command`, `Command.PlayMusic()`, `Command.StopMusic()`, `Command.MemorizeMusic()`

MÉTODO

# Método: Command.PlayMovie()

## Sintaxe

```
Command.PlayMovie(Film, [X, [Y, [Breite, [Altura]]]])
```

## Descrição

Com o método PlayMovie é possível reproduzir um filme.

> **Informação:** Dependendo do caso, pode ser necessário que alguns drivers (`codecs`, ...) estejam instalados no computador em que o jogo está rodando. Se isso é necessário depende do filme usado.

## Lista de parâmetros

### Parâmetro: `Filme`

- **Tipo de dado:** `string`

- **Opcional:** não
- **Descrição:** O filme a ser usado (sem caminho/extensão), relativo à pasta Movie (também pode vir do RTP). Se o filme usa um determinado `codec`, ele precisa estar instalado no computador em que o jogo está rodando.

### Parâmetro: `X`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 0
- **Descrição:** A coordenada X do filme na tela. Essa coordenada se refere ao canto superior esquerdo do filme.

### Parâmetro: `Y`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 0
- **Descrição:** A coordenada Y do filme na tela. Essa coordenada se refere ao canto superior esquerdo do filme.

### Parâmetro: `Largura`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 320
- **Descrição:** A largura do filme na tela (320 = tela cheia, desde que a coordenada X seja 0).

### Parâmetro: `Altura`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** A altura do filme na tela (240 = tela cheia, desde que a coordenada Y seja 0).

## Exemplo

```
$
// Reproduzir o filme "BABBIT" em tela cheia
Command.PlayMovie("BABBIT");
```

```
$
// Reproduzir o filme "BABBIT" (X = 10, Y = 20, larg
Command.PlayMovie("BABBIT", 10, 20, 300, 200);
```

## Erros possíveis

`DESTINYERROR_EXCEPTION`

## Veja também

`Command`, `Command.PlayMusic()`, `Command.PlaySound()`

MÉTODO

# Método: Command.PlayMusic()

## Sintaxe

```
Command.PlayMusic(Musik, [Fade-In-Zeit, [Lautstärke,
[Tempo, [Balanço]]]])
```

## Descrição

Com o método PlayMusic é possível tocar uma música de fundo. Ela pode ser parada depois pelo método `StopMusic`.

> **Informação:** Não é necessário parar a música de fundo atual antes para tocar outra música de fundo. A outra música de fundo pode ser iniciada diretamente por Command.PlayMusic(). Se já estivesse tocando outra música de fundo antes, ela é parada automaticamente.

## Lista de parâmetros

### Parâmetro: `Música`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** A música de fundo a ser usada (sem caminho/extensão), relativa à pasta Music (também pode vir do RTP).

### Parâmetro: `Tempo de fade-in`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 0
- **Descrição:** O tempo de fade-in da música, em milissegundos. (Com o valor 1000, a música só atinge o volume total depois de um segundo; com 2000, depois de dois segundos, e assim por diante.)

### Parâmetro: `Volume`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 100
- **Descrição:** O volume da música de fundo, em porcentagem (100% = volume total).

### Parâmetro: `Tempo (velocidade)`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 100
- **Descrição:** O tempo (andamento) da música de fundo, em porcentagem (100% = velocidade normal).

### Parâmetro: `Balanço`

- **Tipo de dado:** `dword`

- **Opcional:** sim
- **Valor padrão:** A proporção do balanço da música de fundo entre os alto-falantes, em porcentagem:

0%: a música de fundo é tocada apenas pelo alto-falante esquerdo. 50%: a música de fundo é distribuída igualmente pelos dois alto-falantes. 100%: a música de fundo é tocada apenas pelo alto-falante direito.

## Exemplo

```
$
// Tocar a música de fundo "ANGER"
Command.PlayMusic("ANGER");
```

```
$
// Tocar a música de fundo "ANIMAL" (80% do volume,
Command.PlayMusic("ANIMAL", 1000, 80, 120, 45);
```

## Veja também

`Command`, `Command.StopMusic()`, `Command.FadeOutMusic()`, `Command.MemorizeMusic()`, `Command.PlayMemorizedMusic()`, `Command.PlaySound()`, `Command.PlayMovie()`

MÉTODO

# Método: Command.PlaySound()

## Sintaxe

```
Command.PlaySound(Sound, [Lautstärke, [Tempo,
[Balanço]]])
```

## Descrição

Com o método PlaySound é possível reproduzir um som.

## Lista de parâmetros

### Parâmetro: `Som`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** O som a ser usado (sem caminho/extensão), relativo à pasta Sound (também pode vir do RTP).

### Parâmetro: `Volume`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 100
- **Descrição:** O volume do som, em porcentagem (100% = volume total).

### Parâmetro: `Tempo (velocidade)`

- **Tipo de dado:** `dword`

- **Opcional:** sim
- **Valor padrão:** 100
- **Descrição:** O tempo (andamento) do som, em porcentagem (100% = velocidade normal).

### Parâmetro: `Balanço`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 50
- **Descrição:** A proporção do balanço do som entre os alto-falantes, em porcentagem:

0%: o som é tocado apenas pelo alto-falante esquerdo. 50%: o som é distribuído igualmente pelos dois alto-falantes. 100%: o som é tocado apenas pelo alto-falante direito.

## Exemplo

```
$
// Tocar o som "DOG"
Command.PlaySound("DOG");
```

```
$
// Tocar o som "CAT" (volume = 80%, tempo = 120%, ba
Command.PlaySound("CAT", 80, 120, 45);
```

## Veja também

`Command`, `Command.SetSystemSound()`, `Command.PlayMusic()`, `Command.PlayMovie()`

MÉTODO

# Método: Command.ReleaseScreen()

## Sintaxe

```
Command.ReleaseScreen()
```

## Descrição

Com o método ReleaseScreen, a fixação da tela (caso tenha sido feita antes) é desfeita. Assim, a tela volta a seguir o herói.

## Exemplo

```
$
// Fixar a tela na posição atual
Command.FixScreen();

// ...

// Desfazer a fixação da tela
Command.ReleaseScreen();
```

## Veja também

`Command`, `Command.FixScreen()`

MÉTODO

# Método: Command.RemoveActor()

## Sintaxe

```
Command.RemoveActor(Herói-Id)
```

## Descrição

Com o método RemoveActor é possível tirar um herói do grupo.

## Lista de parâmetros

### Parâmetro: `Herói-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... Actor.Count
- **Descrição:** A Actor-Id do herói a ser tirado do grupo. Precisa estar na faixa de 1 a `Actor.Count`.

## Exemplo

```
$
// Adicionar ao grupo o segundo herói (geralmente Br
Command.AddActor(2);

// Tirar do grupo o primeiro herói (geralmente Alex)
Command.RemoveActor(1);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.AddActor()`

MÉTODO

# Método: Command.RemoveCondition()

## Sintaxe

```
Command.RemoveCondition(Herói-Id, Condição-Id)
```

## Descrição

Com o método RemoveCondition é possível curar um herói de uma condição.

## Lista de parâmetros

### Parâmetro: `Herói-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... Actor.Count
- **Descrição:** A Actor-Id do herói a ser curado da condição. Precisa estar na faixa de 1 a `Actor.Count`. Alternativamente, pode-se usar a constante `STATE_WHOLEPARTY`, para que todo o grupo seja curado da condição.

### Parâmetro: `Condição-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... Condition.Count
- **Descrição:** A Condition-Id da condição a curar. Precisa estar na faixa de 1 a `Condition.Count`.

## Exemplo

```
$
// Dar ao primeiro herói a segunda condição (geralm
Command.AddCondition(1, 2);

// Curar todo o grupo da terceira condição (geral
Command.RemoveCondition(STATE_WHOLEPARTY, 3);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.AddCondition()`

MÉTODO

# Método: Command.RemoveItem()

## Sintaxe

```
Command.RemoveItem(Item-Id, [Quantidade])
```

## Descrição

Com o método RemoveItem é possível tirar um item do grupo.

## Lista de parâmetros

### Parâmetro: `Item-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... Item.Count
- **Descrição:** A Item-Id do item a ser tirado do grupo. Precisa estar na faixa de 1 a `Item.Count`.

### Parâmetro: `Quantidade`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 1
- **Descrição:** A quantidade de itens a ser tirada do grupo. Se esse parâmetro não for definido, é tirado um item.

## Exemplo

```
$
// Dar ao grupo o primeiro item (geralmente Potion) uma
Command.AddItem(1);

// Dar ao grupo o segundo item (geralmente Tear Drop)
Command.AddItem(2, 10);

// Tirar do grupo o segundo item (geralmente Tear Drop)
Command.RemoveItem(2, 5);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.AddItem()`

MÉTODO

# Método: Command.RemoveSkill()

## Sintaxe

```
Command.RemoveSkill(Herói-Id, Habilidade-Id)
```

## Descrição

Com o método RemoveSkill é possível fazer um herói esquecer uma técnica.

## Lista de parâmetros

### Parâmetro: `Herói-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... Actor.Count
- **Descrição:** A Actor-Id do herói que deve esquecer a técnica. Precisa estar na faixa de 1 a `Actor.Count`. Alternativamente, pode-se usar a constante `STATE_WHOLEPARTY`, para que todo o grupo esqueça a técnica.

### Parâmetro: `Habilidade-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... Skill.Count
- **Descrição:** A Skill-Id da técnica que o herói deve esquecer. Precisa estar na faixa de 1 a `Skill.Count`.

## Exemplo

```
$
// Ensinar ao primeiro herói a segunda técnica (geral
Command.AddSkill(1, 2);

// Ensinar a todo o grupo (= por herói) a primeira téc
Command.RemoveSkill(STATE_WHOLEPARTY, 1);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.AddSkill()`

MÉTODO

# Método: Command.RemoveTeleportPlace()

## Sintaxe

```
Command.RemoveTeleportPlace(Mapa-Id, X, Y)
```

## Descrição

Com o método RemoveTeleportPlace é possível remover um ponto de teletransporte existente. Os pontos de teletransporte podem ser usados com técnicas do tipo "Teleport".

## Lista de parâmetros

### Parâmetro: `Mapa-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** A Map-Id do ponto de teletransporte a remover. Esse valor precisa ser maior/igual a 1.

### Parâmetro: `X`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A coordenada X do ponto de teletransporte a remover.

### Parâmetro: `Y`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A coordenada Y do ponto de teletransporte a remover.

## Exemplo

```
$
// A posição atual do herói da lista de pontos de tele
Command.RemoveTeleportPlace(Map.Id, Map.HeroX, Ma
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.AddTeleportPlace()`, `Command.EnableTeleport()`, `Map.Id`, `Map.HeroX`, `Map.HeroY`

MÉTODO

# Método: Command.ReplaceLowerChip()

## Sintaxe

```
Command.ReplaceLowerChip(Zu ersetzendes Lower-Chip, Zu
verwendendes Lower-Chip)
```

## Descrição

Com o método ReplaceLowerChip é possível substituir todas as ocorrências de um determinado Lower-Chip não animado no mapa por outro Lower-Chip não animado.

> **Informação:** Essa alteração é volátil e não é salva pelo RPG_RT.

> **Informação:** Para substituir Lower-Chips individuais (inclusive animados), pode-se usar a propriedade `Lower[]` do objeto `Map`.

## Lista de parâmetros

### Parâmetro: `Lower-Chip a substituir`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 143
- **Descrição:** O índice do Lower-Chip a substituir. Esse valor precisa estar na faixa de 0 a 143 (veja a figura; área vermelha). Não é possível substituir Lower-Chips animados.

### Parâmetro: `Lower-Chip a usar`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 143
- **Descrição:** O índice do Lower-Chip pelo qual o outro Lower-Chip deve ser substituído. Esse valor precisa estar na faixa de 0 a 143

(veja a figura; área vermelha). Não é possível substituir Lower-Chips animados.

## Exemplo

```
$
// Definir o chipset como "World" (no banco de dados
Command.SetChipset(1);
// Substituir o Lower-Chip da laje de pedra por um
Command.ReplaceLowerChip(31, 54);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.SetChipset()`, `Command.ReplaceUpperChip()`, `Map.Lower[]`

MÉTODO

# Método: Command.ReplaceUpperChip()

## Sintaxe

```
Command.ReplaceUpperChip(Zu ersetzendes Upper-Chip, Zu
verwendendes Upper-Chip)
```

## Descrição

Com o método ReplaceUpperChip é possível substituir todas as ocorrências de um determinado Upper-Chip no mapa por outro Upper-Chip.

> **Informação:** Essa alteração é volátil e não é salva pelo RPG_RT.

> **Informação:** Para substituir Upper-Chips individuais, pode-se usar a propriedade `Upper[]` do objeto `Map`.

## Lista de parâmetros

### Parâmetro: `Upper-Chip a substituir`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 143
- **Descrição:** O índice do Upper-Chip a substituir. Esse valor precisa estar na faixa de 0 a 143 (veja a figura; área azul).

### Parâmetro: `Upper-Chip a usar`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 143
- **Descrição:** O índice do Upper-Chip pelo qual o outro Upper-Chip deve ser substituído. Esse valor precisa estar na faixa de 0 a 143 (veja a figura; área azul).

## Exemplo

```
$
// Definir o chipset como "World" (no banco de dados
Command.SetChipset(1);
// Substituir o Upper-Chip da árvore por um Upper-Ch
Command.ReplaceUpperChip(1, 6);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.SetChipset()`, `Command.ReplaceLowerChip()`, `Map.Upper[]`

MÉTODO

# Método: Command.ReturnScreen()

## Sintaxe

```
Command.ReturnScreen(Velocidade, Aguardar)
```

## Descrição

Com o método ReturnScreen é possível deslocar a tela de volta para o herói.

> **Informação:** O tempo de espera só é "executado" pelo RPG_RT ao final do DestinyScript, e pode ser sobrescrito por outros comandos do RPG Maker chamados pelo objeto `Command`.

## Lista de parâmetros

### Parâmetro: `Velocidade`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... 6
- **Descrição:** A velocidade do deslocamento. Esse valor precisa estar na faixa de 1 (lento) a 6 (rápido). O valor 4 corresponde à velocidade normal de caminhada.

### Parâmetro: `Aguardar`

- **Tipo de dado:** `bool`
- **Opcional:** não
- **Descrição:** Indica se a continuação do EventScript é interrompida até que a duração indicada se esgote.

False: não se espera. True: espera-se.

## Exemplo

O exemplo a seguir precisa ser inserido em dois comentários separados do RPG Maker, um após o outro (para que o tempo de espera seja de fato cumprido):

```
$
// Mover a tela três campos para a direita
Command.ShiftScreen(DIR_RIGHT, 3, 4, true);
```

```
$
// Mover a tela de volta para o herói
Command.ReturnScreen(4, true);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.ShiftScreen()`

MÉTODO

# Método: Command.SetChipset()

## Sintaxe

```
Command.SetChipset(Chipset-Id)
```

## Descrição

Com o método SetChipset é possível definir o chipset atualmente usado pelo mapa.

## Lista de parâmetros

### Parâmetro: `Chipset-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** A Chipset-Id do chipset a ser atribuído ao mapa em tempo de execução. Esse valor precisa ser uma Chipset-Id válida e, portanto, fica na faixa de 1 até a quantidade de chipsets (esse valor atualmente não pode ser consultado por DestinyScript).

## Exemplo

```
$
// Definir o chipset atual como 3 ("Inner" no banco de d
Command.SetChipset(3);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Map.Chipset`

MÉTODO

# Método: Command.SetEnemyEncounter()

## Sintaxe

```
Command.SetEnemyEncounter(Novo contador de passos)
```

## Descrição

Com o método SetEnemyEncounter é possível definir a quantidade máxima de passos necessária para que um inimigo apareça.

> **Informação:** Essa configuração só é relevante quando o sistema de batalha padrão do Maker é usado e há inimigos associados ao respectivo mapa.

## Lista de parâmetros

### Parâmetro: `Novo contador de passos`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** O novo valor do contador de passos.

## Exemplo

```
$
// Definir o contador de passos para o surgimento de
Command.SetEnemyEncounter(1);
```

## Veja também

`Command`

MÉTODO

# Método: Command.SetEscapePlace()

## Sintaxe

```
Command.SetEscapePlace(Map-Id, X, Y, [Switch-Id])
```

## Descrição

Com o método SetEscapePlace é possível definir o ponto de fuga do sistema de batalha padrão. Se o jogador fugir do combate, ele é teletransportado para o local indicado.

## Lista de parâmetros

### Parâmetro: `Mapa-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** A Map-Id do ponto de fuga. Esse valor precisa ser maior/igual a 1.

### Parâmetro: `X`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A coordenada X do ponto de fuga.

### Parâmetro: `Y`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A coordenada Y do ponto de fuga.

### Parâmetro: `Switch-Id`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 0
- **Faixa de valores:** >= 0
- **Descrição:** A Switch-Id de um switch a ser ativado (quando o ponto de fuga for usado), ou 0, se nenhum switch deve ser ativado.

## Exemplo

```
$
// Definir a posição atual do herói como ponto de fu
Command.SetEscapePlace(Map.Id, Map.HeroX, Map.Her
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.EnableEscape()`, `Map.Id`, `Map.HeroX`, `Map.HeroY`

MÉTODO

# Método: Command.SetEventPlace()

## Sintaxe

```
Command.SetEventPlace(Evento-Id, X, Y)
```

## Descrição

Com o método SetEventPlace é possível definir a posição de um evento.

## Lista de parâmetros

### Parâmetro: `Evento-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A Event-Id do evento a posicionar. Esse valor precisa ser uma Event-Id válida. Também pode ser usada a constante `THIS`, para posicionar o evento atual.

### Parâmetro: `X`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 0
- **Descrição:** A coordenada X do evento no mapa.

### Parâmetro: `Y`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 0
- **Descrição:** A coordenada Y do evento no mapa.

## Exemplo

```
$
// Posicionar o evento atual à direita do herói
Command.SetEventPlace(THIS, Map.HeroX + 1, Map.He
```

## Observações adicionais

O exemplo acima também pode ser feito pelo objeto `Event[]`.

```
$
// Posicionar o evento atual à direita do herói
Event[THIS].X = Map.HeroX + 1;
Event[THIS].Y = Map.HeroY;
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.SwapTwoEvents()`, `Map.Id`, `Map.HeroX`, `Map.HeroY`, `Event[]`

MÉTODO

# Método: Command.SetHeroChar()

## Sintaxe

```
Command.SetHeroChar(Actor-Id, Charset, Charframe,
[Transparente])
```

## Descrição

Com o método SetHeroChar é possível definir o char de um herói.

## Lista de parâmetros

### Parâmetro: `Herói-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... Actor.Count
- **Descrição:** A Actor-Id do herói cujo char deve ser alterado. Precisa estar na faixa de 1 a `Actor.Count`.

### Parâmetro: `Charset`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** O Charset a ser usado (sem caminho/extensão), relativo à pasta Charset (também pode vir do RTP).

### Parâmetro: `Charframe`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** O índice do `Frame` a ser usado. Esse valor deveria ficar na faixa de 0 ... 7 (veja a figura).

### Parâmetro: `Transparente`

- **Tipo de dado:** `bool`
- **Opcional:** ja Standardwert: false
- **Descrição:** Indica se o char do herói deve ser exibido meio-transparente.

False: o char não é exibido transparente. True: o char é exibido meio-transparente.

## Exemplo

```
$
// Deixar o char do primeiro herói meio-transparent
Command.SetHeroChar(1, "MONSTER1", 3, true);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.SetHeroFace()`, `Actor[].Charset`, `Actor[].Charframe`, `Actor[].Transparency`, `Actor[].Default.Charset`, `Actor[].Default.Charframe`, `Actor[].Default.Transparency`

MÉTODO

# Método: Command.SetHeroDegree()

## Sintaxe

```
Command.SetHeroDegree(Herói-Id, Título)
```

## Descrição

Com o método SetHeroDegree é possível definir o título de um herói.

## Lista de parâmetros

### Parâmetro: `Herói-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... Actor.Count
- **Descrição:** A Actor-Id do herói cujo título deve ser alterado. Precisa estar na faixa de 1 a `Actor.Count`.

### Parâmetro: `Título`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** O novo título do herói.

## Exemplo

```
$
// Definir o título do primeiro herói (geralmente "S
Command.SetHeroDegree(1, "Langweiler");
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.SetHeroName()`, `Actor[].Degree`, `Actor[].Default.Degree`

MÉTODO

# Método: Command.SetHeroFace()

## Sintaxe

```
Command.SetHeroFace(Herói-Id, Faceset, Faceframe)
```

## Descrição

Com o método SetHeroFace é possível definir o face de um herói.

## Lista de parâmetros

### Parâmetro: `Herói-Id`

- **Tipo de dado:** `dword`

- **Opcional:** não
- **Faixa de valores:** 1 ... Actor.Count
- **Descrição:** A Actor-Id do herói cujo face deve ser alterado. Precisa estar na faixa de 1 a `Actor.Count`.

### Parâmetro: `Faceset`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** O Faceset a ser usado (sem caminho/extensão), relativo à pasta Faceset (também pode vir do RTP).

### Parâmetro: `Faceframe`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** O índice do `Frame` a ser usado. Esse valor deveria ficar na faixa de 0 ... 15 (veja a figura).

## Exemplo

```
$
// Definir o face do primeiro herói para o rosto zang
Command.SetHeroFace(1, "MONSTERS", 3);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.SetHeroFace()`, `Actor[].Faceset`, `Actor[].Faceframe`, `Actor[].Default.Faceset`, `Actor[].Default.Faceframe`

MÉTODO

# Método: Command.SetHeroName()

## Sintaxe

```
Command.SetHeroName(Herói-Id, Nome)
```

## Descrição

Com o método SetHeroName é possível definir o nome de um herói.

## Lista de parâmetros

### Parâmetro: `Herói-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não

- **Faixa de valores:** 1 ... Actor.Count
- **Descrição:** A Actor-Id do herói cujo nome deve ser alterado. Precisa estar na faixa de 1 a `Actor.Count`.

### Parâmetro: `Nome`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** O novo nome do herói.

## Exemplo

```
$
// Definir o nome do primeiro herói (geralmente "Alex
Command.SetHeroName(1, "Bernd");
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.SetHeroDegree()`, `Actor[].Name`, `Actor[].Default.Name`

MÉTODO

# Método: Command.SetHeroTransparency()

## Sintaxe

```
Command.SetHeroTransparency(Transparência)
```

## Descrição

Com o método SetHeroTransparency é possível definir a transparência do evento do herói.

> **Informação:** A transparência do evento do herói pode ser definida de forma bem mais precisa pela propriedade `Transparency` do objeto `Event[]` (com `HERO` como índice). O valor 1 do parâmetro de transparência deste método equivale a uma transparência de 3 no objeto `Event[]`.

## Lista de parâmetros

### Parâmetro: `Transparência`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 1
- **Descrição:** Indica se o evento do herói deve ser exibido de forma transparente.

0: o evento do herói não é exibido de forma transparente. 1: o evento do herói é exibido de forma transparente.

## Exemplo

```
$
// Exibir o evento do herói de forma transparente
Command.SetHeroTransparency(1);
```

Exemplo com o objeto Event[]:

```
$
// Chamada totalmente equivalente pelo objeto Even
Event[HERO].Transparency = 3;
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Event[]`, `Event[].Transparency`

MÉTODO

# Método: Command.SetMessageStyle()

## Sintaxe

```
Command.SetMessageStyle(Fensterformat,
Fensterposition, Held nicht verstecken, Andere Events
weiterlaufen lassen)
```

## Descrição

Com o método SetMessageStyle é possível definir o estilo das janelas de mensagem.

## Lista de parâmetros

### Parâmetro: `Formato da janela`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 1
- **Descrição:** Indica o formato da janela. Para isso podem ser usadas as constantes de `formato de janela`.

`MESSAGEFORMAT_NORMAL`: as janelas de mensagem são exibidas normalmente, com fundo.
`MESSAGEFORMAT_TRANSPARENT`: as janelas de mensagem são exibidas com fundo transparente.

### Parâmetro: `Posição da janela`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 2
- **Descrição:** Indica a posição da janela. Para isso podem ser usadas as constantes de `posição de janela`.

`MESSAGEPOSITION_TOP`: as janelas de mensagem são exibidas na borda superior da tela. `MESSAGEPOSITION_MIDDLE`: as janelas de mensagem são exibidas no meio da tela. `MESSAGEPOSITION_BOTTOM`: as janelas de mensagem são exibidas na borda inferior da tela.

### Parâmetro: `Não esconder o herói`

- **Tipo de dado:** `bool`
- **Opcional:** não
- **Descrição:** Indica se o herói não deve ficar encoberto pela janela de mensagem.

False: o herói pode ficar encoberto pela janela de mensagem.

True: o herói não fica encoberto pela janela de mensagem (nesse caso, a posição da janela de mensagem muda).

### Parâmetro: `Deixar outros eventos continuarem`

- **Tipo de dado:** `bool`
- **Opcional:** não
- **Descrição:** Indica se outros eventos podem continuar rodando durante a exibição da mensagem (desde que eles próprios não queiram exibir uma mensagem).

False: outros eventos só continuam depois que a janela de mensagem for fechada (independentemente de os processos precisarem ou não da exibição da mensagem). True: outros eventos continuam rodando durante a exibição da mensagem (exceto o respectivo evento que também esteja exibindo uma mensagem).

## Exemplo

```
$
// Deixar as janelas de mensagem transparentes, sob
// encobrir o herói com a mensagem, se necessário
Command.SetMessageStyle(MESSAGEFORMAT_TRANSPARENT
```

## Erros possíveis

`DESTINYERROR_VALUE`

MÉTODO

# Método: Command.SetPanorama()

## Sintaxe

```
Command.SetPanorama(Panorama, [Horizontales Schwenken,
[Vertikales Schwenken, [Horizontale
Scrollgeschwindigkeit, [Vertikale
Scrollgeschwindigkeit]]]])
```

## Descrição

Com o método SetPanorama é possível alterar o panorama atualmente usado, incluindo as propriedades de scroll.

> **Informação:** Para alterar propriedades individuais do panorama, é preciso usar o objeto `Panorama`.

## Lista de parâmetros

### Parâmetro: `Panorama`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** O nome do arquivo do panorama a usar (sem caminho/extensão), relativo à pasta Panorama (também pode vir do RTP). Se esse parâmetro for uma string vazia, o panorama é removido.

### Parâmetro: `Deslocamento horizontal`

- **Tipo de dado:** `bool`
- **Opcional:** sim
- **Valor padrão:** false
- **Descrição:** Indica se o panorama deve se deslocar ao mover o herói na direção horizontal.

False: o panorama não se desloca na direção horizontal. True: o panorama se desloca na direção horizontal.

Esse parâmetro precisa ser true para que o scroll horizontal seja possível.

### Parâmetro: `Deslocamento vertical`

- **Tipo de dado:** `bool`
- **Opcional:** sim
- **Valor padrão:** false
- **Descrição:** Indica se o panorama deve se deslocar ao mover o herói na direção vertical.

False: o panorama não se desloca na direção vertical. True: o panorama se desloca na direção vertical.

Esse parâmetro precisa ser true para que o scroll vertical seja possível.

### Parâmetro: `Velocidade de scroll horizontal`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 0
- **Descrição:** Indica a velocidade com que o panorama deve rolar na direção horizontal.

Com valores < 0: o panorama rola para a esquerda. Com o valor 0: o panorama não rola na direção horizontal. Com valores > 0: o panorama rola para a direita.

O parâmetro "Deslocamento horizontal" precisa ser true para que este parâmetro tenha efeito.

### Parâmetro: `Velocidade de scroll vertical`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 0
- **Descrição:** Indica a velocidade com que o panorama deve rolar na direção vertical.

Com valores < 0: o panorama rola para cima. Com o valor 0: o panorama não rola na direção vertical. Com valores > 0: o panorama rola para baixo.

O parâmetro "Deslocamento vertical" precisa ser true para que este parâmetro tenha efeito.

## Exemplo

```
$
// Definir o panorama atual como "WEIRD" (sem scro
Command.SetPanorama("WEIRD");
```

```
$
// Definir o panorama atual como "DAWN1" (com scr
Command.SetPanorama("DAWN1", true, false, 5, 0);
```

```
$
// Remover o panorama atual
Command.SetPanorama("");
```

## Erros possíveis

`DESTINYERROR_EXCEPTION`

## Veja também

`Command`, `Panorama`

MÉTODO

# Método: Command.SetScreenTone()

## Sintaxe

```
Command.SetScreenTone(Rot, Grün, Blau, Chrominanz,
Dauer, Warten)
```

## Descrição

Com o método SetScreenTone é possível colorir a tela.

> **Informação:** O tempo de espera só é "executado" pelo RPG_RT ao final do DestinyScript, e pode ser sobrescrito por outros comandos do RPG Maker chamados pelo objeto `Command`.

## Lista de parâmetros

### Parâmetro: `Vermelho`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 200
- **Descrição:** A coloração vermelha da tela, em porcentagem. Esse valor precisa ficar na faixa de 0 a 200.

O valor 0 significa que a tela não apresenta a cor vermelha. O valor 100 significa que a tela exibe o vermelho normalmente. O valor 200 significa que a tela é exibida com tons de vermelho especialmente fortes.

### Parâmetro: `Verde`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 200
- **Descrição:** A coloração verde da tela, em porcentagem. Esse valor precisa ficar na faixa de 0 a 200.

O valor 0 significa que a tela não apresenta a cor verde. O valor 100 significa que a tela exibe o verde normalmente. O valor 200 significa que a tela é exibida com tons de verde especialmente fortes.

### Parâmetro: `Azul`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 200
- **Descrição:** A coloração azul da tela, em porcentagem. Esse valor precisa ficar na faixa de 0 a 200.

O valor 0 significa que a tela não apresenta a cor azul. O valor 100 significa que a tela exibe o azul normalmente. O valor 200 significa que a tela é exibida com tons de azul especialmente fortes.

### Parâmetro: `Crominância`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 200
- **Descrição:** A crominância (saturação de cor) da tela, em porcentagem. Esse valor precisa ficar na faixa de 0 a 200.

O valor 0 significa que a tela é exibida dessaturada. O valor 100 significa que a tela é exibida normalmente. O valor 200 significa que a tela é exibida com diferenças de cor especialmente grandes.

### Parâmetro: `Duração`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 0
- **Descrição:** A duração, em décimos de segundo, até a alteração de cor se completar.

### Parâmetro: `Aguardar`

- **Tipo de dado:** `bool`
- **Opcional:** não
- **Descrição:** Indica se a continuação do EventScript é interrompida até que a duração indicada se esgote.

False: não se espera. True: espera-se.

## Exemplo

O exemplo a seguir precisa ser inserido em dois comentários separados do RPG Maker, um após o outro (para que o tempo de espera seja de fato cumprido):

```
$
// Guardar os valores de cor anteriores nas variáve
v[1] = Screen.Red;
v[2] = Screen.Green;
v[3] = Screen.Blue;
```

```
v[4] = Screen.Chroma;

// Colorir a tela em tons de cinza em um segundo
Command.SetScreenTone(100, 100, 100, 0, 10, true)
```

```
$
// Restaurar a tela às cores anteriores em um segund
Command.SetScreenTone(v[1], v[2], v[3], v[4], 10,
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Screen`

MÉTODO

# Método: Command.SetScreenTransition()

## Sintaxe

```
Command.SetScreenTransition(Transition-Ereignis,
Transition)
```

## Descrição

Com o método SetScreenTransition é possível definir a transição de um determinado evento de transição.

## Lista de parâmetros

### Parâmetro: `Evento de transição`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 5
- **Descrição:** O evento cuja transição deve ser definida. Para isso são usadas as `constantes de transição`:

`TRANSITION_TELEPORTERASE`: a transição ao apagar a tela antes de uma teletransportação.
`TRANSITION_TELEPORTSHOW`: a transição ao exibir a tela depois de uma teletransportação.
`TRANSITION_BATTLESTARTERASE`: a transição ao apagar a tela antes de um combate.
`TRANSITION_BATTLESTARTSHOW`: a transição ao exibir a tela no início de um combate.
`TRANSITION_BATTLEENDERASE`: a transição ao apagar a tela no fim de um combate. `TRANSITION_BATTLEENDSHOW`: a transição ao exibir a tela depois de um combate.

### Parâmetro: `Transição`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 20 (Erase) / 0 ... 19 (Show)

- **Descrição:** A transição a ser usada (do tipo Erase ou Show). Em transições Erase, são usadas as `constantes de erase`:

`ERASE_FADEOUT`: a tela desaparece com fade-out `ERASE_RANDOMBLOCKS`: a tela se preenche aleatoriamente com blocos `ERASE_RANDOMBLOCKSUP`: a tela se preenche aleatoriamente com blocos, de baixo para cima `ERASE_RANDOMBLOCKSDOWN`: a tela se preenche aleatoriamente com blocos, de cima para baixo `ERASE_BLINDCLOSE`: a tela se preenche com listras horizontais `ERASE_STRIPEUPDOWN`: a tela é preenchida simultaneamente de cima e de baixo, cada uma com uma faixa preta `ERASE_STRIPELEFTRIGHT`: a tela é preenchida simultaneamente da esquerda e da direita, cada uma com uma faixa preta `ERASE_OUTSIDEINSIDE`: a tela se preenche em forma de retângulo, de fora para dentro `ERASE_INSIDEOUTSIDE`: a tela se preenche em forma de retângulo, de dentro para fora `ERASE_SCROLLUP`: a tela é deslocada para fora, para cima `ERASE_SCROLLDOWN`: a tela é deslocada para fora, para baixo `ERASE_SCROLLLEFT`: a tela é deslocada para fora, para a esquerda `ERASE_SCROLLRIGHT`: a tela é deslocada para fora, para a direita `ERASE_DIVISIONUPDOWN`: a tela se divide horizontalmente ao meio, e as metades de cima e de baixo se afastam simultaneamente `ERASE_DIVISIONLEFTRIGHT`: a tela se divide verticalmente ao meio, e as metades da esquerda e da direita se afastam simultaneamente `ERASE_DIVISION`: a tela se divide horizontal e verticalmente ao meio, e os quatro retângulos se afastam simultaneamente em direção ao seu canto da tela `ERASE_ZOOMIN`: a tela é fortemente ampliada em direção ao herói `ERASE_MOSAIC`: a tela desaparece através de um mosaico cada vez mais grosso `ERASE_ROSTERSCROLL`: a tela borra de forma ondulada `ERASE_INSTANT`: a tela fica preta imediatamente, sem transição `ERASE_DONTERASE`: a tela não é apagada

Em transições Show, são usadas as `constantes de show`:

`SHOW_FADEIN`: a tela é exibida com fade-in `SHOW_RANDOMBLOCKS`: a tela se preenche aleatoriamente com blocos do novo conteúdo `SHOW_RANDOMBLOCKSUP`: a tela se preenche aleatoriamente com blocos do novo conteúdo, de baixo para cima `SHOW_RANDOMBLOCKSDOWN`: a tela se preenche aleatoriamente com blocos do novo conteúdo, de cima para baixo `SHOW_BLINDOPEN`: a tela se preenche com listras horizontais do novo conteúdo `SHOW_STRIPEUPDOWN`: a tela é preenchida simultaneamente de cima e de baixo, cada uma com uma faixa do novo conteúdo `SHOW_STRIPELEFTRIGHT`: a tela é preenchida simultaneamente da esquerda e da direita, cada uma com uma faixa do novo conteúdo `SHOW_OUTSIDEINSIDE`: a tela se preenche em forma de retângulo, de fora para dentro `SHOW_INSIDEOUTSIDE`: a tela se preenche em forma de retângulo, de dentro para fora `SHOW_SCROLLDOWN`: a tela é deslocada para dentro vindo de cima `SHOW_SCROLLUP`: a tela é deslocada para dentro vindo de baixo `SHOW_SCROLLRIGHT`: a tela é deslocada para dentro vindo da esquerda `SHOW_SCROLLLEFT`: a tela é deslocada para dentro vindo da direita `SHOW_COMBINEUPDOWN`: a tela se forma a partir de duas partes horizontais do novo conteúdo, vindas simultaneamente de cima e de baixo `SHOW_COMBINELEFTRIGHT`: a tela se forma a partir de duas partes verticais do novo conteúdo, vindas simultaneamente da esquerda e da direita `SHOW_COMBINE`: a tela se forma a partir de quatro partes do novo conteúdo, vindas simultaneamente dos cantos da tela `SHOW_ZOOMOUT`: a tela é reduzida em direção ao herói, de um grande aumento até o tamanho normal `SHOW_MOSAIC`: a tela é exibida através de um mosaico cada vez mais fino `SHOW_ROSTERSCROLL`: a tela está borrada de forma ondulada e vai ficando cada vez mais nítida `SHOW_INSTANT`: o novo conteúdo da tela fica visível imediatamente, sem transição

Informação: dependendo da constante usada em "Evento de transição", são usadas as constantes `Erase` ou `Show` no segundo parâmetro.

## Exemplo

```
$
// A transição ao apagar a tela antes de uma teletra
Command.SetScreenTransition(TRANSITION_BATTLESTAR
// A transição ao exibir a tela no início de um comb
Command.SetScreenTransition(TRANSITION_BATTLESTAR
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`

MÉTODO

# Método: Command.SetSystemGraphic()

## Sintaxe

```
Command.SetSystemGraphic(Systemgrafik, Darstellungsart
der Nachrichten, Schriftart)
```

## Descrição

Com o método SetSystemGraphic é possível definir o gráfico do sistema.

## Lista de parâmetros

### Parâmetro: `Gráfico do sistema`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** O gráfico do sistema a ser usado (sem caminho/extensão), relativo à pasta System (também pode vir do RTP).

### Parâmetro: `Modo de exibição das mensagens`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 1
- **Descrição:** O tipo de exibição da mensagem. Para isso são usadas as `constantes de exibição`:

`DISPLAY_STRETCH`: o gráfico do sistema é exibido esticado nas caixas de mensagem. `DISPLAY_REPEAT`: o gráfico do sistema é exibido repetido lado a lado nas caixas de mensagem.

### Parâmetro: `Fonte`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 1
- **Descrição:** A fonte usada. Para isso são usadas as `constantes de fonte`:

`FONT_RMG2000`: é usada a fonte "RMG2000". `FONT_RM2000`: é usada a fonte "RM2000".

Informação: os nomes das fontes usadas podem ser alterados pelo DestinyPatcher.

## Exemplo

```
$
// Definir o gráfico do sistema como "ROYAL" (o texto
Command.SetSystemGraphic("ROYAL", DISPLAY_STRETCH
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`

MÉTODO

# Método: Command.SetSystemMusic()

## Sintaxe

```
Command.SetSystemMusic(Systemmusik-Typ, Musik, [Fade-
In-Zeit, [Lautstärke, [Tempo, [Balanço]]]])
```

## Descrição

Com o método SetSystemMusic é possível definir a música de fundo de um evento do sistema.

## Lista de parâmetros

### Parâmetro: `Tipo de música do sistema`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 6
- **Descrição:** O tipo de música do sistema a ser alterado. Para isso são usadas as `constantes de tipo de música`:

`MUSIC_BATTLE`: a música que toca durante um combate. `MUSIC_VICTORY`: a música que toca durante uma vitória. `MUSIC_INN`: a música que toca durante uma estadia em pousada. `MUSIC_BOAT`: a música que toca durante uma viagem de barco. `MUSIC_SHIP`: a música que toca durante uma viagem de navio. `MUSIC_AIRSHIP`: a música que toca durante uma viagem de aeronave. `MUSIC_GAMEOVER`: a música que toca durante a tela de "Game Over".

### Parâmetro: `Música`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** A música de fundo a ser usada (sem caminho/extensão), relativa à pasta Music (também pode vir do RTP).

### Parâmetro: `Tempo de fade-in`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 0
- **Descrição:** O tempo de fade-in da música, em milissegundos. (Com o valor 1000, a música só atinge o volume total

depois de um segundo; com 2000, depois de dois segundos, e assim por diante.)

### Parâmetro: `Volume`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 100
- **Descrição:** O volume da música de fundo, em porcentagem (100% = volume total).

### Parâmetro: `Tempo (velocidade)`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 100
- **Descrição:** O tempo (andamento) da música de fundo, em porcentagem (100% = velocidade normal).

### Parâmetro: `Balanço`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 50
- **Descrição:** A proporção do balanço da música de fundo entre os alto-falantes, em porcentagem:

0%: a música de fundo é tocada apenas pelo alto-falante esquerdo. 50%: a música de fundo é distribuída igualmente pelos dois alto-falantes. 100%: a música de fundo é tocada apenas pelo alto-falante direito.

## Exemplo

```
$
// Definir a música de fundo do combate como "Anger"
Command.SetSystemMusic(MUSIC_BATTLE, "ANGER");
// Definir a música de fundo da vitória como "Animal"
// (tempo de fade-in: 2 segundos, volume: 80%, temp
Command.SetSystemMusic(MUSIC_VICTORY, "ANIMAL", 2
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.SetSystemSound()`, `Command.PlayMusic()`

MÉTODO

# Método: Command.SetSystemSound()

## Sintaxe

```
Command.SetSystemSound(System-Sound-Typ, Sound,
[Lautstärke, [Tempo, [Balanço]]])
```

## Descrição

Com o método SetSystemSound é possível definir o som de um evento do sistema.

## Lista de parâmetros

### Parâmetro: `Tipo de som do sistema`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 11
- **Descrição:** O tipo do som do sistema a ser alterado. Para isso são usadas as `constantes de tipo de som`:

`SOUND_CURSOR`: o som que toca ao mover o cursor. `SOUND_DECISION`: o som que toca ao selecionar uma opção. `SOUND_CANCEL`: o som que toca ao cancelar uma seleção. `SOUND_BUZZER`: o som que toca ao selecionar uma opção bloqueada. `SOUND_STARTBATTLE`: o som que toca ao iniciar um combate. `SOUND_ESCAPE`: o som que toca ao fugir de um combate. `SOUND_ENEMYATTACK`: o som que toca quando um inimigo ataca. `SOUND_ENEMYDAMAGE`: o som que toca quando um inimigo sofre dano. `SOUND_ALLYDAMAGE`: o som que toca quando um jogador sofre dano. `SOUND_EVASION`: o som que toca ao esquivar de um ataque. `SOUND_ENEMYDIE`: o som que toca quando um inimigo morre. `SOUND_ITEMUSE`: o som que toca ao usar um item.

### Parâmetro: `Som`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** O som a ser usado (sem caminho/extensão), relativo

à pasta Sound (também pode vir do RTP).

### Parâmetro: `Volume`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 100
- **Descrição:** O volume do som, em porcentagem (100% = volume total).

### Parâmetro: `Tempo (velocidade)`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 100
- **Descrição:** O tempo (andamento) do som, em porcentagem (100% = velocidade normal).

### Parâmetro: `Balanço`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 50
- **Descrição:** A proporção do balanço do som entre os alto-falantes, em porcentagem:

0%: o som é tocado apenas pelo alto-falante esquerdo. 50%: o som é distribuído igualmente pelos dois alto-falantes. 100%: o som é tocado apenas pelo alto-falante direito.

## Exemplo

```
$
// Definir o som do cursor como "Cursor2"
Command.SetSystemSound(SOUND_CURSOR, "CURSOR2");
// Definir o som de seleção como "Cat"
// (volume: 100%, tempo: 80%, balanço: 55% (disc
Command.SetSystemSound(SOUND_DECISION, "CAT", 100
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.SetSystemMusic()`, `Command.PlaySound()`

MÉTODO

# Método: Command.SetTimer()

## Sintaxe

```
Command.SetTimer(Minutos, Segundos)
```

## Descrição

Com o método SetTimer é possível definir o valor atual do timer. A indicação é feita em minutos e segundos.

## Lista de parâmetros

### Parâmetro: `Minutos`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 999
- **Descrição:** A quantidade atual de minutos.

### Parâmetro: `Segundos`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 59
- **Descrição:** A quantidade atual de segundos.

## Exemplo

```
$
// Definir o valor atual do timer como 2 minutos e
Command.SetTimer(2, 15);
```

```
// Iniciar o timer (fica visível no mapa e no com
Command.StartTimer(true, true);

// ...

// Parar o timer em algum momento (por exemplo, em um
Command.StopTimer();
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.StartTimer()`, `Command.StopTimer()`

MÉTODO

# Método: Command.SetVehicleChar()

## Sintaxe

```
Command.SetVehicleChar(Fahrzeug-Event-Id, Charset,
Charframe)
```

## Descrição

Com o método SetVehicleChar é possível definir o char de um veículo.

## Lista de parâmetros

### Parâmetro: `Evento-do-veículo-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 10002 ... 10004
- **Descrição:** A Event-Id do veículo cujo char deve ser alterado. Precisa estar na faixa de 10002 a 10003. Para isso são usadas as constantes de `eventos especiais`:

`BOAT`: o char do barco deve ser alterado. `SHIP`: o char do navio deve ser alterado. `AIRSHIP`: o char da aeronave deve ser alterado.

### Parâmetro: `Charset`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** O Charset a ser usado (sem caminho/extensão), relativo à pasta Charset (também pode vir do RTP).

### Parâmetro: `Charframe`

- **Tipo de dado:** `dword`
- **Opcional:** não

- **Descrição:** O índice do `Frame` a ser usado. Esse valor deveria ficar na faixa de 0 ... 7 (veja a figura).

## Exemplo

```
$
// Transformar o char da aeronave em um dragão
Command.SetVehicleChar(AIRSHIP, "VEHICLE", 6);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Event[].Charset`, `Event[].Charframe`

MÉTODO

# Método: Command.SetVehiclePlace()

## Sintaxe

```
Command.SetVehiclePlace(Veículo-Id, Mapa-Id, X, Y)
```

## Descrição

Com o método SetVehiclePlace é possível definir a posição de um veículo.

## Lista de parâmetros

### Parâmetro: `Veículo-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 10002 ... 10004
- **Descrição:** A Id do veículo. Para isso são usadas as indicações de veículo das `constantes de eventos especiais`:

`BOAT`: o barco deve ser posicionado. `SHIP`: o navio deve ser posicionado. `AIRSHIP`: a aeronave deve ser posicionada.

### Parâmetro: `Mapa-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** A Map-Id do mapa em que o veículo deve ser posicionado. Esse valor deveria ser uma Map-Id válida.

### Parâmetro: `X`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 0
- **Descrição:** A coordenada X do veículo no mapa.

### Parâmetro: `Y`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 0
- **Descrição:** A coordenada Y do veículo no mapa.

## Exemplo

```
$
// Posicionar a aeronave na mesma posição do herói
Command.SetVehiclePlace(AIRSHIP, Map.Id, Map.Hero

// Fazer o herói entrar no veículo mais próximo (ist
Command.EnterLeaveVehicle();
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.EnterLeaveVehicle()`, `Map.Id`, `Map.HeroX`, `Map.HeroY`

MÉTODO

# Método: Command.SetWeather()

## Sintaxe

```
Command.SetWeather(Tipo de clima, Intensidade)
```

## Descrição

Com o método SetWeather é possível definir um determinado clima.

## Sinônimos

CallWeatherEffect()

## Lista de parâmetros

### Parâmetro: `Tipo de clima`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 2
- **Descrição:** O tipo de clima a definir. Para isso são usadas as `constantes de clima`:

`WEATHER_NONE`: nenhum clima (= sol) `WEATHER_RAIN`: chuva `WEATHER_SNOW`: neve

### Parâmetro: `Intensidade`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 2
- **Descrição:** A intensidade do clima. Esse valor é ignorado quando o tipo de clima é `WEATHER_NONE`. Para isso são usadas as constantes de `intensidade de clima`:

`WEATHERSTRENGTH_WEAK`: o clima tem intensidade fraca. `WEATHERSTRENGTH_NORMAL`: o clima tem intensidade normal. `WEATHERSTRENGTH_STRONG`: o clima tem intensidade forte.

## Exemplo

```
$
// Invocar uma chuva de intensidade média
Command.SetWeather(WEATHER_RAIN, WEATHERSTRENGTH_
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`

MÉTODO

# Método: Command.ShakeScreen()

## Sintaxe

```
Command.ShakeScreen(Stärke, Geschwindigkeit, Dauer,
Warten)
```

## Descrição

Com o método ShakeScreen é possível fazer a tela tremer (por exemplo, para simular um terremoto). O tremor sempre acontece na direção horizontal.

> **Informação:** O tempo de espera só é "executado" pelo RPG_RT ao final do DestinyScript, e pode ser sobrescrito por outros comandos do RPG Maker chamados pelo objeto `Command`.

## Lista de parâmetros

### Parâmetro: `Intensidade`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A intensidade do tremor (basicamente, a amplitude). Esse valor deveria ficar na faixa de 1 (fraco) a 9 (forte). Com valores maiores/menores, o tremor fica mais forte ou mais fraco, mas também pode deixar de acontecer.

### Parâmetro: `Velocidade`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A velocidade do tremor (basicamente, a frequência). Esse valor deveria ficar na faixa de 1 (lento) a 9 (rápido). Com valores maiores/menores, o tremor fica mais rápido ou mais lento, mas também pode deixar de acontecer.

### Parâmetro: `Duração`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 0
- **Descrição:** A duração, em décimos de segundo, até o tremor acabar.

### Parâmetro: `Aguardar`

- **Tipo de dado:** `bool`
- **Opcional:** não
- **Descrição:** Indica se a continuação do EventScript é interrompida até que a duração indicada se esgote.

False: não se espera. True: espera-se.

## Exemplo

O exemplo a seguir precisa estar sozinho em um comentário do RPG Maker, para que nenhum outro método do `Command` sobrescreva o tempo de espera (e, assim, o tempo de espera seja de fato cumprido):

```
$
// Fazer a tela tremer por três segundos, com uma
Command.ShakeScreen(1, 9, 30, true);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`

MÉTODO

# Método: Command.ShiftScreen()

## Sintaxe

```
Command.ShiftScreen(Richtung, Anzahl Felder,
Geschwindigkeit, Warten)
```

## Descrição

Com o método ShiftScreen é possível deslocar a tela (por exemplo, para mostrar um determinado ponto do mapa).

> **Informação:** O tempo de espera só é "executado" pelo RPG_RT ao final do DestinyScript, e pode ser sobrescrito por outros comandos do RPG Maker chamados pelo objeto `Command`.

## Lista de parâmetros

### Parâmetro: `Direção`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 3
- **Descrição:** A direção para a qual a tela deve ser deslocada. Para isso são usadas as `constantes de direção` .

`DIR_UP`: a tela deve ser deslocada para cima. `DIR_RIGHT`: a tela deve ser deslocada para a direita. `DIR_DOWN`: a tela deve ser deslocada para baixo. `DIR_LEFT`: a tela deve ser deslocada para a esquerda.

### Parâmetro: `Quantidade de campos`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A quantidade de campos que a tela deve ser deslocada.

### Parâmetro: `Velocidade`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... 6
- **Descrição:** A velocidade do deslocamento. Esse valor precisa estar na faixa de 1 (lento) a 6 (rápido). O valor 4 corresponde à velocidade normal de caminhada.

### Parâmetro: `Aguardar`

- **Tipo de dado:** `bool`
- **Opcional:** não
- **Descrição:** Indica se a continuação do EventScript é interrompida até

que a duração indicada se esgote.

False: não se espera. True: espera-se.

## Exemplo

O exemplo a seguir precisa ser inserido em dois comentários separados do RPG Maker, um após o outro (para que o tempo de espera seja de fato cumprido):

```
$
// Mover a tela três campos para a direita
Command.ShiftScreen(DIR_RIGHT, 3, 4, true);
```

```
$
// Mover a tela de volta para o herói
Command.ReturnScreen(4, true);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.ReleaseScreen()`

MÉTODO

# Método: Command.ShowBattleAnimation()

## Sintaxe

```
Command.ShowBattleAnimation(Battle-Animation-Id,
Event-Id, Vollbild, Warten)
```

## Descrição

Com o método ShowBattleAnimation é possível exibir uma animação de batalha.

> **Informação:** O tempo de espera só é "executado" pelo RPG_RT ao final do DestinyScript, e pode ser sobrescrito por outros comandos do RPG Maker chamados pelo objeto `Command`.

## Lista de parâmetros

### Parâmetro: `Animação-de-batalha-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** A Battle-Animation-Id da animação de batalha a exibir. Esse valor precisa ser uma Battle-Animation-Id válida e, portanto, fica na faixa de 1 até a quantidade de animações de batalha (esse valor atualmente não pode ser consultado por DestinyScript).

### Parâmetro: `Evento-Id`

- **Tipo de dado:** `dword`

- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** A Event-Id do respectivo evento no qual a animação de batalha deve ser exibida. Esse valor precisa ser uma Event-Id válida ou um evento especial. Aqui também podem ser usadas as constantes de `eventos especiais`:

`HERO`: o evento do herói `BOAT`: o evento do barco `SHIP`: o evento do navio `AIRSHIP`: o evento da aeronave `THIS`: o evento atual

### Parâmetro: `Tela cheia`

- **Tipo de dado:** `bool`
- **Opcional:** não
- **Descrição:** Indica se a animação de batalha é esticada para a tela inteira ou exibida apenas na área do evento.

False: a animação de batalha é exibida apenas na área do evento.

True: a animação de batalha é esticada para a tela inteira.

### Parâmetro: `Aguardar`

- **Tipo de dado:** `bool`
- **Opcional:** não
- **Descrição:** Indica se a continuação do EventScript é interrompida até que a duração indicada se esgote.

False: não se espera. True: espera-se.

## Exemplo

```
$
// Exibir a primeira animação de batalha (geralmente
Command.ShowBattleAnimation(1, HERO, true, false)
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`

MÉTODO

# Método: Command.ShowFace()

## Sintaxe

```
Command.ShowFace(Faceset, Frame, Position des Faces,
Gespiegelt)
```

## Descrição

Com o método ShowFace é possível definir um face para as mensagens.

## Lista de parâmetros

### Parâmetro: `Faceset`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** O nome do arquivo do Faceset (sem caminho/extensão). Se esse parâmetro for uma string vazia, o face é removido.

### Parâmetro: `Frame`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 15
- **Descrição:** O índice do face usado. Esse valor fica na faixa de 0 ... 15 (veja a figura).

### Parâmetro: `Posição do face`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 1
- **Descrição:** Indica a posição do face. Para isso podem ser usadas as `constantes de posição de face`.

`FACEPOSITION_LEFT`: o face é exibido na borda esquerda da caixa de mensagem. `FACEPOSITION_RIGHT`: o face é exibido na borda direita da caixa de mensagem.

### Parâmetro: `Espelhado`

- **Tipo de dado:** `bool`
- **Opcional:** não
- **Descrição:** Indica se o face deve ser exibido espelhado.

False: o rosto é exibido normalmente. True: o rosto é exibido espelhado.

## Exemplo

```
$
// Colocar o rosto do Alex na caixa de mensagem
Command.ShowFace("Chara1", 0, FACEPOSITION_RIGHT,

// Remover novamente o face da caixa de mensagem
Command.ShowFace("", 0, FACEPOSITION_LEFT, false)
```

## Erros possíveis

`DESTINYERROR_VALUE`, `DESTINYERROR_EXCEPTION`

## Veja também

`Command`, `Faceset.Filename`, `Faceset.Frame`, `Faceset.Position`, `Faceset.Flipped`

MÉTODO

# Método: Command.ShowPicture()

## Sintaxe

```
Command.ShowPicture(Picture-Id, Picture, X, Y, [Mit
Karte bewegen, [Vergrößerung, [Transparenz,
[Maskenfarbe verwenden, [Rot, [Grün, [Blau,
[Chrominanz, [Aktion, [Intensidade da ação]]]]]]]]]])
```

## Descrição

Com o método ShowPicture é possível carregar e exibir um gráfico como Picture.

## Lista de parâmetros

### Parâmetro: `Picture-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... Picture.Count
- **Descrição:** A Picture-Id da Picture a exibir. Esse valor precisa ser uma Picture-Id válida e, portanto, fica na faixa de 1 a `Picture.Count`.

### Parâmetro: `Picture`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** O nome do arquivo da Picture (sem caminho/extensão). A Picture é buscada relativa à pasta Picture, e também pode vir do RTP.

### Parâmetro: `X`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A coordenada X da Picture. Ela se refere ao centro da Picture.

### Parâmetro: `Y`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A coordenada Y da Picture. Ela se refere ao centro da Picture.

### Parâmetro: `Mover junto com o mapa`

- **Tipo de dado:** `bool`
- **Opcional:** sim
- **Valor padrão:** false
- **Descrição:** Indica se a Picture se move junto com o mapa.

False: a Picture não se move junto. True: a Picture se move junto.

### Parâmetro: `Ampliação`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 100
- **Faixa de valores:** >= 1
- **Descrição:** O fator de ampliação da Picture, em porcentagem. O valor padrão 100 indica que a Picture é exibida normalmente.

### Parâmetro: `Transparência`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 0
- **Descrição:** A transparência da Picture, em porcentagem. Esse valor deveria ficar na faixa de 0 a 100. Com valores maiores que 100, ocorrem erros de cálculo divertidos.

### Parâmetro: `Usar cor da máscara`

- **Tipo de dado:** `bool`
- **Opcional:** sim
- **Valor padrão:** true
- **Descrição:** Indica se a cor de máscara é usada.

False: a cor de máscara não é usada. True: a cor de máscara é usada (valor padrão).

### Parâmetro: `Vermelho`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 100
- **Descrição:** A coloração vermelha da Picture, em porcentagem. Esse valor deveria ficar na faixa de 0 a 200.

O valor 0 significa que a Picture não apresenta a cor vermelha. O valor 100 significa que a Picture exibe o vermelho normalmente. O valor 200 significa que a Picture é exibida com tons de vermelho especialmente fortes.

### Parâmetro: `Verde`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 100
- **Descrição:** A coloração verde da Picture, em porcentagem. Esse valor deveria ficar na faixa de 0 a 200.

O valor 0 significa que a Picture não apresenta a cor verde. O valor 100 significa que a Picture exibe o verde normalmente. O valor 200 significa que a Picture é exibida com tons de verde especialmente fortes.

### Parâmetro: `Azul`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 100
- **Descrição:** A coloração azul da Picture, em porcentagem. Esse valor deveria ficar na faixa de 0 a 200.

O valor 0 significa que a Picture não apresenta a cor azul. O valor 100 significa que a Picture exibe o azul normalmente. O valor 200 significa que a Picture é exibida com tons de azul especialmente fortes.

### Parâmetro: `Crominância`

- **Tipo de dado:** `dword`
- **Opcional:** ja Standardwert: 100
- **Descrição:** A crominância (saturação de cor) da Picture, em porcentagem. Esse valor deveria ficar na faixa de 0 a 200.

O valor 0 significa que a Picture é exibida dessaturada. O valor 100 significa que a Picture é exibida normalmente. O valor 200 significa que a Picture é exibida com diferenças de cor especialmente grandes.

### Parâmetro: `Ação`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 0
- **Faixa de valores:** 0 ... 2
- **Descrição:** A ação da Picture. Para isso são usadas as `constantes de ação`:

`ACTION_NONE`: a Picture não realiza nenhuma ação. `ACTION_ROTATION`: a Picture gira em torno do próprio eixo. `ACTION_RIPPLE`: a Picture realiza um efeito de onda.

### Parâmetro: `Intensidade da ação`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 0
- **Descrição:** A intensidade de ação da Picture. O significado desse valor depende da ação escolhida para a Picture. Na ação `ACTION_NONE`, o valor deste parâmetro é ignorado.

## Exemplo

```
$
// Carregar o gráfico de nuvem do RTP como Picture 1 e exi
Command.ShowPicture(1, "CLOUD", 160, 120);
```

## Erros possíveis

`DESTINYERROR_VALUE`, `DESTINYERROR_EXCEPTION`

## Veja também

`Command`, `Command.MovePicture()`, `Command.ErasePicture()`, `Picture`

MÉTODO

# Método: Command.ShowScreen()

## Sintaxe

```
Command.ShowScreen(Transição)
```

## Descrição

Com o método ShowScreen é possível exibir a tela por meio de fade-in ("mostrar").

## Lista de parâmetros

### Parâmetro: `Transição`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 19
- **Descrição:** A transição a ser usada (do tipo Show). Para isso são usadas as `constantes de show`:

`SHOW_FADEIN`: a tela é exibida com fade-in `SHOW_RANDOMBLOCKS`: a tela se preenche aleatoriamente com blocos do novo conteúdo `SHOW_RANDOMBLOCKSUP`: a tela se preenche aleatoriamente com blocos do novo conteúdo, de baixo para cima `SHOW_RANDOMBLOCKSDOWN`: a tela se preenche aleatoriamente com blocos do novo conteúdo, de cima para baixo `SHOW_BLINDOPEN`: a tela se preenche com listras horizontais do novo conteúdo `SHOW_STRIPEUPDOWN`: a tela é preenchida simultaneamente de cima e de baixo, cada uma com uma faixa do novo conteúdo `SHOW_STRIPELEFTRIGHT`: a tela é preenchida simultaneamente da esquerda e da direita, cada uma com uma faixa do novo conteúdo `SHOW_OUTSIDEINSIDE`: a tela se preenche em forma de retângulo, de fora para dentro `SHOW_INSIDEOUTSIDE`: a tela se preenche em forma de retângulo, de dentro para fora `SHOW_SCROLLDOWN`: a tela é deslocada para dentro vindo de cima `SHOW_SCROLLUP`: a tela é deslocada para dentro vindo de baixo `SHOW_SCROLLRIGHT`: a tela é deslocada para dentro vindo da esquerda `SHOW_SCROLLLEFT`: a tela é deslocada para dentro vindo da direita `SHOW_COMBINEUPDOWN`: a tela se forma a partir de duas partes horizontais do novo conteúdo, vindas simultaneamente de cima e de baixo `SHOW_COMBINELEFTRIGHT`: a tela se forma a partir de duas partes verticais do novo conteúdo, vindas simultaneamente da esquerda e da direita `SHOW_COMBINE`: a tela se forma a partir de quatro partes do novo conteúdo, vindas simultaneamente dos cantos da tela `SHOW_ZOOMOUT`: a tela é reduzida em direção ao herói, de um grande aumento até o tamanho normal `SHOW_MOSAIC`: a tela é exibida através de um mosaico cada vez mais fino `SHOW_ROSTERSCROLL`: a tela está borrada de forma ondulada e vai ficando cada vez mais nítida `SHOW_INSTANT`: o novo conteúdo da tela fica visível imediatamente, sem transição

## Exemplo

```
$
// Apagar a tela de fora para dentro
Command.EraseScreen(ERASE_OUTSIDEINSIDE);
// Teletransportar o herói para o canto inferior direito
Command.Teleport(Map.Id, Map.Width - 1, Map.Heigh
// Exibir a tela de dentro para fora
Command.ShowScreen(SHOW_INSIDEOUTSIDE);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.Teleport()`, `Command.EraseScreen()`, `Map.Id`, `Map.HeroX`, `Map.HeroY`

MÉTODO

# Método: Command.StartTimer()

## Sintaxe

```
Command.StartTimer(Timer anzeigen, Timer im Kampf
verfügbar)
```

## Descrição

Com o método StartTimer é possível iniciar o timer.

## Lista de parâmetros

### Parâmetro: `Exibir timer`

- **Tipo de dado:** `bool`
- **Opcional:** não
- **Descrição:** Indica se o timer é exibido.

False: o timer não é exibido. True: o timer é exibido.

### Parâmetro: `Timer disponível em batalha`

- **Tipo de dado:** `bool`
- **Opcional:** não
- **Descrição:** Indica se o timer fica disponível em combate. Se o timer for exibido, ele também é exibido em combate nesse caso.

False: o timer não continua correndo durante o combate, e também não é exibido lá. True: o timer continua correndo durante o combate, e é exibido lá, se for o caso.

## Exemplo

```
$
// Definir o valor atual do timer como 2 minutos e
Command.SetTimer(2, 15);

// Iniciar o timer (fica visível no mapa e no com
Command.StartTimer(true, true);

// ...

// Parar o timer em algum momento (por exemplo, em um
Command.StopTimer();
```

## Veja também

`Command`, `Command.SetTimer()`, `Command.StopTimer()`

MÉTODO

# Método: Command.StopAll()

## Sintaxe

```
Command.StopAll()
```

## Descrição

Com o método StopAll, os passos de movimento a executar são cancelados. Os passos de movimento a cancelar precisam ter sido executados antes em um comando MoveEvent (EventScript).

## Exemplo

Antes do exemplo a seguir, um comando MoveEvent (EventScript) deveria ter sido executado, para que se note algum efeito:

```
$
// Cancela os comandos MoveEvent (EventScript) exec
Command.StopAll();
```

## Veja também

`Command`, `Command.MoveEvent()`, `Command.MoveAll()`

MÉTODO

# Método: Command.StopMusic()

## Sintaxe

```
Command.StopMusic()
```

## Descrição

Com o método StopMusic é possível parar a música de fundo atual. Uma música de fundo pode ser iniciada pelo método `PlayMusic`.

> **Informação:** Não é necessário parar a música de fundo atual antes para tocar outra música de fundo. A outra música de fundo pode ser iniciada diretamente por `Command.PlayMusic()`. Se já estivesse tocando outra música de fundo antes, ela é parada automaticamente.

## Exemplo

```
$
// Tocar a música de fundo "ANGER"
Command.PlayMusic("ANGER");
// Parar imediatamente a música de fundo atual
Command.StopMusic();
```

## Veja também

`Command`, `Command.PlayMusic()`, `Command.FadeOutMusic()`

MÉTODO

# Método: Command.StopParallelProcesses()

## Sintaxe

```
Command.StopParallelProcesses()
```

## Descrição

Com o método StopParallelProcesses é possível interromper eventos em execução (no mapa ou CommonEvent) do tipo "Parallel Process".

## Exemplo

```
$
// Interrompe processos paralelos
Command.StopParallelProcesses();
```

## Veja também

`Command`

MÉTODO

# Método: Command.StopTimer()

## Sintaxe

```
Command.StopTimer()
```

## Descrição

Com o método StopTimer é possível parar o timer. Se o timer for iniciado novamente depois, o tempo anterior continua de onde parou.

## Exemplo

```
$
// Definir o valor atual do timer como 2 minutos e
Command.SetTimer(2, 15);

// Iniciar o timer (fica visível no mapa e no com
Command.StartTimer(true, true);

// ...
```

```
// Parar o timer em algum momento (por exemplo, em um
Command.StopTimer();
```

## Veja também

`Command`, `Command.SetTimer()`, `Command.StartTimer()`

MÉTODO

# Método: Command.SubtractAbility()

## Sintaxe

```
Command.SubtractAbility(Herói-Id, Tipo, Quantidade)
```

## Descrição

Com o método SubtractAbility é possível retirar pontos de habilidade de um herói.

## Lista de parâmetros

### Parâmetro: `Herói-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... Actor.Count
- **Descrição:** A Actor-Id do herói ao qual os pontos de habilidade devem ser retirados. Precisa estar na faixa de 1 a `Actor.Count`. Alternativamente, pode-se usar a constante `STATE_WHOLEPARTY`, para que todo o grupo perca pontos de habilidade.

### Parâmetro: `Tipo`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 5
- **Descrição:** O tipo dos pontos de habilidade. Para isso são usadas as `constantes de valores de status`:

`ABILITY_MAXHP`: o herói perde pontos de vida máximos (equivale a `Actor[].MaxHpDiff` -= Valor). `ABILITY_MAXMP` : o herói perde pontos de magia máximos (equivale a `Actor[].MaxMpDiff` -= Valor). `ABILITY_ATTACK`: o herói perde pontos de ataque (equivale a `Actor[].AttackDiff` -= Valor). `ABILITY_DEFENSE`: o herói perde pontos de defesa (equivale a `Actor[].DefenseDiff` -= Valor). `ABILITY_MIND`: o herói perde pontos de inteligência (equivale a `Actor[].MindDiff` -= Valor). `ABILITY_AGILITY`: o herói perde pontos de agilidade (equivale a `Actor[].AgilityDiff` -= Valor).

### Parâmetro: `Quantidade`

- **Tipo de dado:** `dword`
- **Opcional:** nein Beschreibung: Die Anzahl der Fähigkeitspunkte, die der Held verlieren soll.

## Exemplo

```
$
// Aumentar em 50 os pontos de ataque do primeiro he
Command.AddAbility(1, ABILITY_ATTACK, 50);

// Dar a todo o grupo (= por herói) dez pontos de defe
Command.SubtractLevel(STATE_WHOLEPARTY, ABILITY_D
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.AddAbility()`

MÉTODO

# Método: Command.SubtractExp()

## Sintaxe

```
Command.SubtractExp(Herói-Id, Quantidade)
```

## Descrição

Com o método SubtractExp é possível retirar pontos de experiência de um herói.

## Lista de parâmetros

### Parâmetro: `Herói-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... Actor.Count
- **Descrição:** A Actor-Id do herói ao qual os pontos de experiência devem ser retirados. Precisa estar na faixa de 1 a `Actor.Count`. Alternativamente, pode-se usar a constante `STATE_WHOLEPARTY`, para que todo o grupo perca pontos de experiência.

### Parâmetro: `Quantidade`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A quantidade de pontos de experiência que o herói deve perder.

## Exemplo

```
$
// Adicionar 100 pontos de experiência ao primeiro he
// (aqui, se for o caso, uma mensagem de aumento de
```

```
Command.AddExp(1, 100, true);

// Adicionar 50 pontos de experiência a todo o grupo
Command.SubtractExp(STATE_WHOLEPARTY, 50);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.AddExp()`

MÉTODO

# Método: Command.SubtractGold()

## Sintaxe

```
Command.SubtractGold(Quantidade)
```

## Descrição

Com o método SubtractGold é possível retirar dinheiro do grupo.

## Lista de parâmetros

### Parâmetro: `Quantidade`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A quantidade de dinheiro a ser retirada do grupo.

## Exemplo

```
$
// Dar 100 de dinheiro ao grupo
Command.AddGold(100);

// Tirar 100 de dinheiro do grupo (assim como veio,
Command.SubtractGold(100);
```

## Veja também

`Command`, `Command.AddGold()`

MÉTODO

# Método: Command.SubtractHp()

## Sintaxe

```
Command.SubtractHp(Actor-Id, Anzahl, [Sterben
möglich?])
```

## Descrição

Com o método SubtractHp é possível fazer um herói perder pontos de vida.

## Lista de parâmetros

### Parâmetro: `Herói-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... Actor.Count
- **Descrição:** A Actor-Id do herói que deve perder pontos de vida. Precisa estar na faixa de 1 a `Actor.Count`. Alternativamente, pode-se usar a constante `STATE_WHOLEPARTY`, para que todo o grupo perca pontos de vida.

### Parâmetro: `Quantidade`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A quantidade de pontos de vida que o herói deve perder.

Se não for possível morrer, os pontos de vida do herói não podem cair abaixo de 1. Se for possível morrer, os pontos de vida do herói não podem cair abaixo de 0. Se todos os heróis do grupo tiverem 0 pontos de vida, o jogo é perdido (Game Over).

### Parâmetro: `Pode morrer?`

- **Tipo de dado:** `bool`
- **Opcional:** sim
- **Valor padrão:** true
- **Descrição:** Indica se um herói pode morrer por perda de pontos de vida.

False: não é possível morrer por perda de pontos de vida (por isso, os pontos de vida não caem abaixo de 1 com essa ação). True: é possível morrer por perda de pontos de vida. Se todos os heróis estiverem mortos, o jogo é perdido.

## Exemplo

```
$
// Dar 100 pontos de vida ao primeiro herói
Command.AddHp(1, 100);

// Dar 50 pontos de vida a todo o grupo (= por herói)
Command.SubtractHp(STATE_WHOLEPARTY, 50, false);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.AddHp()`

MÉTODO

# Método: Command.SubtractLevel()

## Sintaxe

```
Command.SubtractLevel(Herói-Id, Quantidade)
```

## Descrição

Com o método SubtractLevel é possível retirar níveis de um herói.

## Lista de parâmetros

### Parâmetro: `Herói-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... Actor.Count
- **Descrição:** A Actor-Id do herói ao qual os níveis devem ser retirados. Precisa estar na faixa de 1 a `Actor.Count`. Alternativamente, pode-se usar a constante `STATE_WHOLEPARTY`, para que todo o grupo perca níveis.

### Parâmetro: `Quantidade`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A quantidade de níveis que o herói deve perder.

## Exemplo

```
$
// Fazer o primeiro herói subir dois níveis
// (aqui, se for o caso, uma mensagem de aumento de
Command.AddLevel(1, 2, true);

// Fazer todo o grupo (= por herói) descer um nível
Command.SubtractLevel(STATE_WHOLEPARTY, 1);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.AddLevel()`

MÉTODO

# Método: Command.SubtractMp()

## Sintaxe

```
Command.SubtractMp(Herói-Id, Quantidade)
```

## Descrição

Com o método SubtractMp é possível fazer um herói perder pontos de magia.

## Lista de parâmetros

### Parâmetro: `Herói-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... Actor.Count
- **Descrição:** A Actor-Id do herói que deve perder pontos de magia. Precisa estar na faixa de 1 a `Actor.Count`. Alternativamente, pode-se usar a constante `STATE_WHOLEPARTY`, para que todo o grupo perca pontos de magia.

### Parâmetro: `Quantidade`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A quantidade de pontos de magia que o herói deve perder. Os pontos de magia do herói não podem cair abaixo de 0.

## Exemplo

```
$
// Dar 100 pontos de magia ao primeiro herói
Command.AddMp(1, 100);

// Dar 50 pontos de magia a todo o grupo (= por herói)
Command.SubtractMp(STATE_WHOLEPARTY, 50);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.AddMp()`

MÉTODO

# Método: Command.SwapTwoEvents()

## Sintaxe

```
Command.SwapTwoEvents(Evento-Id 1, Evento-Id 2)
```

## Descrição

Com o método SwapTwoEvents é possível trocar a posição de dois eventos.

## Lista de parâmetros

### Parâmetro: `Evento-Id 1`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A Event-Id do primeiro evento a trocar. Esse valor precisa ser uma Event-Id válida. Também pode ser usada a constante `THIS`, para trocar a posição do evento atual.

### Parâmetro: `Evento-Id 2`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A Event-Id do segundo evento a trocar. Esse valor precisa ser uma Event-Id válida. Também pode ser usada a constante `THIS`, para trocar a posição do evento atual.

## Exemplo

```
$
// Trocar a posição do primeiro evento com a posição
Command.SwapTwoEvents(1, 2);
```

## Observações adicionais

O exemplo acima também pode ser feito pelo objeto `Event[]`.

```
$
// Guardar temporariamente as coordenadas do primeiro
d[1] = Event[1].X;
d[2] = Event[1].Y;
// Definir as coordenadas do primeiro evento com as coor
Event[1].X = Event[2].X;
Event[1].Y = Event[2].Y;
// Definir as coordenadas do segundo evento com as coor
Event[2].X = d[1];
Event[2].Y = d[2];
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.SetEventPlace()`, `Map.Id`, `Map.HeroX`, `Map.HeroY`, `Event[]`

MÉTODO

# Método: Command.Teleport()

## Sintaxe

```
Command.Teleport(Mapa-Id, X, Y)
```

## Descrição

Com o método Teleport é possível teletransportar o herói para uma determinada posição de um determinado mapa.

> **Informação:** Os parâmetros deste método precisam obrigatoriamente ser válidos, caso contrário o RPG_RT trava.

## Lista de parâmetros

### Parâmetro: `Mapa-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** A Id do mapa de destino. Esse valor precisa ser uma Map-Id válida e precisa ser maior/igual a 1. Se aqui for informada uma Map-Id inválida, o jogo trava. Isso não pode ser evitado.

### Parâmetro: `X`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 0
- **Descrição:** A coordenada X do herói no mapa. Esse valor precisa ser maior/igual a 0.

### Parâmetro: `Y`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 0
- **Descrição:** A coordenada Y do herói no mapa. Esse valor precisa ser maior/igual a 0.

## Exemplo

```
$
// Teletransportar o herói para a borda direita do
Command.Teleport(Map.Id, Map.Width - 1, Map.HeroY
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`, `Command.MemorizePlace()`, `Command.GoToMemorizedPlace()`, `Map.Id`, `Map.Width`, `Map.Height`, `Map.HeroX`, `Map.HeroY`

MÉTODO

# Método: Command.TitleScreen()

## Sintaxe

```
Command.TitleScreen()
```

## Descrição

Com o método TitleScreen é possível encerrar a partida atual pela tela de título. A partida é encerrada imediatamente e o jogo entra no modo de tela de título; o DestinyScript não continua rodando e também é encerrado imediatamente.

## Exemplo

```
$
// Encerrar a partida atual pela tela de título
Command.TitleScreen();
```

## Veja também

`Command`, `Command.GameOver()`

MÉTODO

# Método: Command.Wait()

## Sintaxe

```
Command.Wait(Duração)
```

## Descrição

Com o método Wait é possível interromper a execução do EventScript por um determinado tempo.

> **Informação:** O tempo de espera só é "executado" pelo RPG_RT ao final do DestinyScript, e pode ser sobrescrito por outros comandos do RPG Maker chamados pelo objeto `Command`.

## Lista de parâmetros

### Parâmetro: `Duração`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 0
- **Descrição:** A duração, em décimos de segundo, até que a execução do EventScript continue.

## Exemplo

O exemplo a seguir precisa estar sozinho em um comentário do RPG Maker, para que nenhum outro método do `Command` sobrescreva o tempo de espera (e, assim, o tempo de espera seja de fato cumprido):

```
$
// Interromper a execução do evento atual por 1 segu
Command.Wait(10);
```

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Command`

OBJETO

# Objeto: CommonEvent

## Sintaxe

```
CommonEvent
```

## Descrição

O objeto CommonEvent serve para tornar acessíveis todos os CommonEvents do banco de dados do jogo.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Count` | Propriedade | Devolve a quantidade de CommonEvents. |
| `CommonEvent[]` | Objeto | Permite o acesso a um determinado elemento do objeto CommonEvent. |

PROPRIEDADE

# Propriedade: CommonEvent.Count

## Sintaxe

```
CommonEvent.Count
```

## Descrição

Devolve a quantidade de CommonEvents. Isso pode ser usado para fazer um laço sobre todos os CommonEvents.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

## Exemplo

```
$
// Montar uma lista de todos os CommonEvents e
a[1] = "";
for(v[1] = 1; v[1] <= CommonEvent.Count; v[1]++)
  a[1] ..= CommonEvent[v[1]].Name .. CRLF;
next;
```

## Veja também

`CommonEvent`, `CommonEvent[]`, `CommonEvent[].Name`

OBJETO

# Objeto: CommonEvent[]

## Sintaxe

```
CommonEvent[CommonEvent-Id]
```

## Descrição

Permite o acesso a um determinado elemento do objeto `CommonEvent`.

## Lista de parâmetros

### Parâmetro: `CommonEvent-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... CommonEvent.Count

- **Descrição:** A CommonEvent-Id do respectivo CommonEvent. Esse valor precisa estar na faixa de 1 ... `CommonEvent.Count`.

## Erros possíveis

`DESTINYERROR_ARRAYBOUND`

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `CallMode` | Propriedade | Com a propriedade CallMode é possível descobrir o modo de chamada de um CommonEvent. |
| `Id` | Propriedade | Com a propriedade Id é possível descobrir a CommonEvent-Id de um CommonEvent. |
| `Name` | Propriedade | Com a propriedade Name é possível descobrir o nome de um CommonEvent. |
| `Switch` | Propriedade | Com a propriedade Switch é possível descobrir a Switch-Id do switch que precisa estar True para ativar o CommonEvent. |

PROPRIEDADE

# Propriedade: CommonEvent[].CallMode

## Sintaxe

```
CommonEvent[CommonEvent-Id].CallMode
```

## Descrição

Com a propriedade CallMode é possível descobrir o modo de chamada de um CommonEvent. Para identificar esse valor, podem ser usadas as constantes `CallMode`.

`CALLMODE_CALL`: o CommonEvent é iniciado por chamadas. `CALLMODE_PARALLELPROCESS`: o CommonEvent roda como processo paralelo. `CALLMODE_AUTOSTART`: o CommonEvent roda como processo paralelo que bloqueia o controle do jogo e os movimentos livres.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

## Exemplo

```
$
// Verificar se o primeiro CommonEvent é iniciado por
if(CommonEvent[1].CallMode == CALLMODE_CALL)
  // Chamar o primeiro CommonEvent
  Command.CallCommonEvent(1);
endif;
```

## Veja também

`CommonEvent`, `CommonEvent[]`, `Command.CallCommonEvent()`

PROPRIEDADE

# Propriedade: CommonEvent[].Id

## Sintaxe

```
CommonEvent[CommonEvent-Id].Id
```

## Descrição

Com a propriedade Id é possível descobrir a CommonEvent-Id de um CommonEvent. Esse valor corresponde ao índice usado no objeto `CommonEvent[]`.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

## Exemplo

```
$
// Descobrir a CommonEvent-Id do primeiro CommonEvent e
v[1] = CommonEvent[1].Id;  // 1 (= o valor no
```

## Veja também

`CommonEvent`, `CommonEvent[]`, `CommonEvent.Count`

PROPRIEDADE

# Propriedade: CommonEvent[].Name

## Sintaxe

```
CommonEvent[CommonEvent-Id].Name
```

## Descrição

Com a propriedade Name é possível descobrir o nome de um CommonEvent.

## Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** sim

## Exemplo

```
$
// Ler o nome do primeiro CommonEvent e guardar em
a[1] = CommonEvent[1].Name;
```

O exemplo a seguir pode ser usado para chamar um CommonEvent que tenha o mesmo nome do evento atual.

```
$
// Guardar o nome do evento atual
a[1] = Event[THIS].Name;
// Procurar um CommonEvent com o mesmo nome
for(v[1] = 1; v[1] <= CommonEvent.Count; v[1]++)
  if(CommonEvent[v[1]].Name == a[1])
    // O nome é igual -> chamar o CommonEvent
    Command.CallCommonEvent(v[1]);
    break;
  endif;
next;
```

## Veja também

`CommonEvent`, `CommonEvent[]`, `CommonEvent.Count`, `CommonEvent[].Name`, `Event[].Name`, `Command.CallCommonEvent()`

PROPRIEDADE

# Propriedade: CommonEvent[].Switch

## Sintaxe

```
CommonEvent[CommonEvent-Id].Switch
```

## Descrição

Com a propriedade Switch é possível descobrir a Switch-Id do switch que precisa estar True para ativar o CommonEvent. Esse valor só é válido quando o `CallMode` não é `CALLMODE_CALL`. Se esse valor for 0, nenhum switch é necessário para ativar o CommonEvent.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** >= 0

## Exemplo

```
$
// Verificar se o primeiro CommonEvent tem um switch
if(CommonEvent[1].Switch > 0)
  // Ativar o respectivo switch
  s[CommonEvent[1].Switch] = true;
endif;
```

## Veja também

`CommonEvent`, `CommonEvent[]`, `CommonEvent[].CallMode`

OBJETO

# Objeto: Condition

## Sintaxe

```
Condition
```

## Descrição

O objeto Condition serve para tornar acessíveis todas as condições dos heróis e inimigos do banco de dados do jogo.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Count` | Propriedade | Devolve a quantidade de condições. |
| `Condition[]` | Objeto | Permite o acesso a um determinado elemento do objeto Condition. |

PROPRIEDADE

# Propriedade: Condition.Count

## Sintaxe

```
Condition.Count
```

## Descrição

Devolve a quantidade de condições. Isso pode ser usado para fazer um laço sobre todas as condições.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

## Exemplo

```
$
// Montar uma lista de todas as condições e
a[1] = "";
for(v[1] = 1; v[1] <= Condition.Count; v[1]++)
  a[1] ..= Condition[v[1]].Name .. CRLF;
next;
```

## Veja também

`Condition`, `Condition[]`, `Condition[].Name`

OBJETO

# Objeto: Condition[]

## Sintaxe

```
Condition[Condição-Id]
```

## Descrição

Permite o acesso a um determinado elemento do objeto `Condition`.

## Lista de parâmetros

### Parâmetro: `Condição-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... Condition.Count
- **Descrição:** A Condition-Id da respectiva condição. Esse valor precisa estar na faixa de 1 ... `Condition.Count`.

## Erros possíveis

`DESTINYERROR_ARRAYBOUND`

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `ActionLimitation` | Propriedade | Com a propriedade ActionLimitation é possível descobrir a limitação de ação de uma condição. |
| `AllyStateMessage` | Propriedade | Com a propriedade AllyStateMessage é possível descobrir o texto exibido quando um herói recebe essa condição. |
| `AlreadyInStateMessage` | Propriedade | Com a propriedade AlreadyInStateMessage é possível descobrir o texto exibido quando se tenta dar essa condição a um herói/inimigo que já tem essa condição. |
| `Chance` | Objeto | Com o objeto Chance é possível descobrir as chances de adquirir uma condição. |
| `Chance[]` | Propriedade | Com a propriedade Chance é possível descobrir a chance de adquirir a condição, em porcentagem (0 ... |
| `Color` | Propriedade | Com a propriedade Color é possível descobrir a cor de uma condição. |
| `EnemyStateMessage` | Propriedade | Com a propriedade EnemyStateMessage é possível descobrir o texto exibido quando um inimigo recebe essa condição. |
| `HalfAgility` | Propriedade | Com a propriedade HalfAgility é possível descobrir se um herói fica com apenas metade dos seus pontos de agilidade disponíveis quando tem essa condição. |
| `HalfAttack` | Propriedade | Com a propriedade HalfAttack é possível descobrir se um herói fica com apenas metade dos seus pontos de ataque disponíveis quando tem essa condição. |
| `HalfDefense` | Propriedade | Com a propriedade HalfDefense é possível descobrir se um herói fica com apenas metade dos seus pontos de defesa disponíveis quando tem essa condição. |
| `HalfMind` | Propriedade | Com a propriedade HalfMind é possível descobrir se um herói fica com apenas metade dos seus pontos de inteligência disponíveis quando tem essa condição. |

| Nome | Tipo | Resumo |
| --- | --- | --- |
| `HitChanceLimitation` | Propriedade | Com a propriedade HitChanceLimitation é possível descobrir quais técnicas não podem ser usadas quando um herói/inimigo está sob efeito dessa condição. |
| `HitRateChange` | Propriedade | Com a propriedade HitRateChange é possível descobrir em que medida a chance de acerto de um herói/inimigo (em porcentagem) é alterada quando ele tem essa condição. |
| `MindChanceLimitation` | Propriedade | Com a propriedade MindChanceLimitation é possível descobrir quais técnicas não podem ser usadas quando um herói/inimigo está sob efeito dessa condição. |
| `Name` | Propriedade | Com a propriedade Name é possível descobrir o nome de uma condição. |
| `NaturalHealingChance` | Propriedade | Com a propriedade NaturalHealingChance é possível descobrir a chance (em porcentagem) de uma condição parar de fazer efeito sozinha no início de um turno. |
| `NaturalHealingTurns` | Propriedade | Com a propriedade NaturalHealingTurns é possível descobrir a quantidade de turnos (em combate) necessária até que uma condição possa parar de fazer efeito sozinha. |
| `PercentDownPerTurnHp` | Propriedade | Com a propriedade PercentDownPerTurnHp é possível descobrir a quantidade de pontos de vida (em porcentagem) que o herói/inimigo perde por turno devido a essa condição. |
| `PercentDownPerTurnMp` | Propriedade | Com a propriedade PercentDownPerTurnMp é possível descobrir a quantidade de pontos de magia (em porcentagem) que o herói/inimigo perde por turno devido a essa condição. |
| `PointDownPerStepHp` | Propriedade | Com a propriedade PointDownPerStepHp é possível descobrir a quantidade de pontos de vida (em pontos) que o herói perde ao se mover pelo mapa devido a essa condição. |
| `PointDownPerStepMp` | Propriedade | Com a propriedade PointDownPerStepMp é possível descobrir a quantidade de pontos de magia (em pontos) que o herói perde ao se mover pelo mapa devido a essa condição. |
| `PointDownPerTurnHp` | Propriedade | Com a propriedade PointDownPerTurnHp é possível descobrir a quantidade de pontos de vida (em pontos) que o herói/inimigo perde por turno devido a essa condição. |
| `PointDownPerTurnMp` | Propriedade | Com a propriedade PointDownPerTurnMp é possível descobrir a quantidade de pontos de magia (em pontos) que o herói/inimigo perde por turno devido a essa condição. |
| `Priority` | Propriedade | Com a propriedade Priority é possível descobrir a prioridade de uma condição. |
| `RecoveryStateMessage` | Propriedade | Com a propriedade RecoveryStateMessage é possível descobrir o texto exibido quando um herói/inimigo é curado dessa condição. |
| `ShockHealingChance` | Propriedade | Com a propriedade ShockHealingChance é possível descobrir a chance (em porcentagem) de a condição indicada ser curada por um ataque físico. |
| `StepsDownHp` | Propriedade | Com a propriedade StepsDownHp é possível descobrir a quantidade de passos necessária para que o herói perca pontos de vida ao se mover pelo mapa. |
| `StepsDownMp` | Propriedade | Com a propriedade StepsDownMp é possível descobrir a quantidade de passos necessária para que o herói perca pontos de magia ao se mover pelo mapa. |
| `StillInStateMessage` | Propriedade | Com a propriedade StillInStateMessage é possível descobrir o texto exibido quando um herói/inimigo tem, em combate, mais um turno sob efeito dessa condição. |
| `Type` | Propriedade | Com a propriedade Type é possível descobrir o tipo de uma condição. |

PROPRIEDADE

## Propriedade: Condition[].ActionLimitation

### Sintaxe

```
Condition[Condição-Id].ActionLimitation
```

### Descrição

Com a propriedade ActionLimitation é possível descobrir a limitação de ação de uma condição. Para identificar esse valor, podem ser usadas as constantes `ConditionLimitation`.

`CONDITIONLIMITATION_NONE`: as ações possíveis do herói não têm restrição no sistema de batalha
`CONDITIONLIMITATION_NOACTION`: o herói não pode mais executar ações em combate
`CONDITIONLIMITATION_ATTACKENEMY`: o herói ataca automaticamente um inimigo aleatório em combate
`CONDITIONLIMITATION_ATTACKALLY`: o herói ataca automaticamente um aliado aleatório em combate

### Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** 0 ... 3

PROPRIEDADE

## Propriedade: Condition[].AllyStateMessage

### Sintaxe

```
Condition[Condição-Id].AllyStateMessage
```

### Descrição

Com a propriedade AllyStateMessage é possível descobrir o texto exibido quando um herói recebe essa condição.

### Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Condition[].AlreadyInStateMessage

### Sintaxe

```
Condition[Condição-Id].AlreadyInStateMessage
```

## Descrição

Com a propriedade AlreadyInStateMessage é possível descobrir o texto exibido quando se tenta dar essa condição a um herói/inimigo que já tem essa condição.

## Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** sim

OBJETO

# Objeto: Condition[].Chance

## Sintaxe

```
Condition[Condição-Id].Chance
```

## Descrição

Com o objeto Chance é possível descobrir as chances de adquirir uma condição.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Count` | Propriedade | Devolve sempre 5, já que o RPG Maker prevê exatamente 5 chances. |

PROPRIEDADE

# Propriedade: Condition[].Chance.Count

## Sintaxe

```
Condition[Condição-Id].Chance.Count
```

## Descrição

Devolve sempre 5, já que o RPG Maker prevê exatamente 5 chances.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** 5

PROPRIEDADE

# Propriedade: Condition[].Chance[]

## Sintaxe

```
Condition[Condição-Id].Chance[Nível]
```

## Descrição

Com a propriedade Chance é possível descobrir a chance de adquirir a condição, em porcentagem (0 ... 100), em um dos cinco níveis (A ... E).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** 0 ... 100

## Lista de parâmetros

### Parâmetro: `Nível`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** Indica o respectivo nível (A = 0, B = 1, C = 2, D = 3, E = 4). Para isso podem ser usadas as constantes `Effect`.

## Erros possíveis

`DESTINYERROR_ARRAYBOUND`

## Veja também

`Actor[].ConditionEffect[]`

PROPRIEDADE

# Propriedade: Condition[].Color

## Sintaxe

```
Condition[Condição-Id].Color
```

## Descrição

Com a propriedade Color é possível descobrir a cor de uma condição. Para identificar esse valor, podem ser usadas as constantes de `cores de texto`.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** 0 ... 19

PROPRIEDADE

# Propriedade: Condition[].EnemyStateMessage

## Sintaxe

```
Condition[Condição-Id].EnemyStateMessage
```

## Descrição

Com a propriedade EnemyStateMessage é possível descobrir o texto exibido quando um inimigo recebe essa condição.

## Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Condition[].HalfAgility

## Sintaxe

```
Condition[Condição-Id].HalfAgility
```

## Descrição

Com a propriedade HalfAgility é possível descobrir se um herói fica com apenas metade dos seus pontos de agilidade disponíveis quando tem essa condição. (False = os pontos de agilidade permanecem inalterados; True = os pontos de agilidade ficam pela metade)

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

## Veja também

`Condition[].HalfAttack`, `Condition[].HalfDefense`, `Condition[].HalfMind`

PROPRIEDADE

# Propriedade: Condition[].HalfAttack

## Sintaxe

```
Condition[Condição-Id].HalfAttack
```

## Descrição

Com a propriedade HalfAttack é possível descobrir se um herói fica com apenas metade dos seus pontos de ataque disponíveis quando tem essa condição. (False = os pontos de ataque permanecem inalterados; True = os pontos de ataque ficam pela metade)

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

## Veja também

`Condition[].HalfDefense`, `Condition[].HalfMind`, `Condition[].HalfAgility`

PROPRIEDADE

# Propriedade: Condition[].HalfDefense

## Sintaxe

```
Condition[Condição-Id].HalfDefense
```

## Descrição

Com a propriedade HalfDefense é possível descobrir se um herói fica com apenas metade dos seus pontos de defesa disponíveis quando tem essa condição. (False = os pontos de defesa permanecem inalterados; True = os pontos de defesa ficam pela metade)

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

## Veja também

`Condition[].HalfAttack`, `Condition[].HalfMind`, `Condition[].HalfAgility`

PROPRIEDADE

# Propriedade: Condition[].HalfMind

## Sintaxe

```
Condition[Condição-Id].HalfMind
```

## Descrição

Com a propriedade HalfMind é possível descobrir se um herói fica com apenas metade dos seus pontos de inteligência disponíveis quando tem essa condição. (False = os pontos de inteligência permanecem inalterados; True = os pontos de inteligência ficam pela metade)

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

## Veja também

`Condition[].HalfAttack`, `Condition[].HalfDefense`, `Condition[].HalfAgility`

PROPRIEDADE

# Propriedade: Condition[].HitChanceLimitation

## Sintaxe

```
Condition[Condição-Id].HitChanceLimitation
```

## Descrição

Com a propriedade HitChanceLimitation é possível descobrir quais técnicas não podem ser usadas quando um herói/inimigo está sob efeito dessa condição. Todas as técnicas com um valor de HitChance maior/igual ao valor aqui indicado deixam de poder ser usadas em combate pelo herói/inimigo. Se essa opção não estiver ativada no banco de dados do jogo, é devolvido 0.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** >= 0

PROPRIEDADE

## Propriedade: Condition[].HitRateChange

### Sintaxe

```
Condition[Condição-Id].HitRateChange
```

### Descrição

Com a propriedade HitRateChange é possível descobrir em que medida a chance de acerto de um herói/inimigo (em porcentagem) é alterada quando ele tem essa condição. O valor da condição padrão é 100 (por padrão, todos os acertos são bem-sucedidos, a menos que o inimigo esquive). (0 = nenhum acerto é bem-sucedido, 50 = metade dos acertos é bem-sucedida, 100 = todos os acertos são bem-sucedidos (= caso normal))

### Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** 0 ... 100

PROPRIEDADE

## Propriedade: Condition[].MindChanceLimitation

### Sintaxe

```
Condition[Condição-Id].MindChanceLimitation
```

### Descrição

Com a propriedade MindChanceLimitation é possível descobrir quais técnicas não podem ser usadas quando um herói/inimigo está sob efeito dessa condição. Todas as técnicas com um valor de MindChance maior/igual ao valor aqui indicado deixam de poder ser usadas em combate pelo herói/inimigo. Se essa opção não estiver ativada no banco de dados do jogo, é devolvido 0.

### Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** >= 0

PROPRIEDADE

## Propriedade: Condition[].Name

### Sintaxe

```
Condition[Condição-Id].Name
```

## Descrição

Com a propriedade Name é possível descobrir o nome de uma condição.

## Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** sim

## Exemplo

No exemplo a seguir, partimos do princípio de que o jogo acabou de ser iniciado e que o banco de dados padrão do jogo está sendo usado.

```
$
// Ler o nome da primeira condição e guardar em
a[1] = Condition[1].Name;    // "Normal"
// Dar ao primeiro herói as condições "Poison" e
Command.AddCondition(1, 2);  // 2 = Poison no banco de
Command.AddCondition(1, 8);  // 8 = Numb no banco de
// Listar todas as condições do primeiro herói e
// Isso resultaria, ao final, em "Condições do herói "Al
a[2] = "Konditionen des Helden """ .. Actor[1].Na
for(v[1] = 1; v[1] <= Condition.Count; v[1]++)
  if(Actor[1].HasCondition(v[1]))
    a[2] ..= Condition[v[1]].Name .. CRLF;
  endif;
next;
```

## Veja também

`Condition`, `Condition[]`, `Condition.Count`, `Actor[].Name`, `Actor[].HasCondition()`

PROPRIEDADE

# Propriedade: Condition[].NaturalHealingChance

## Sintaxe

```
Condition[Condição-Id].NaturalHealingChance
```

## Descrição

Com a propriedade NaturalHealingChance é possível descobrir a chance (em porcentagem) de uma condição parar de fazer efeito sozinha no início de um turno. Para isso, porém, é preciso primeiro ultrapassar a quantidade de turnos (desde a aquisição da condição) indicada pela propriedade `NaturalHealingTurns`. Depois disso, em cada turno seguinte, verifica-se se a condição foi curada.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** 0 ... 100

## Veja também

`Condition[].NaturalHealingTurns`, `Condition[].ShockHealingChance`

PROPRIEDADE

# Propriedade: Condition[].NaturalHealingTurns

## Sintaxe

```
Condition[Condição-Id].NaturalHealingTurns
```

## Descrição

Com a propriedade NaturalHealingTurns é possível descobrir a quantidade de turnos (em combate) necessária até que uma condição possa parar de fazer efeito sozinha. Uma vez ultrapassada a quantidade de turnos indicada (desde a aquisição da condição), o valor da propriedade `NaturalHealingChance` decide, em cada turno seguinte, se a condição foi curada.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** >= 0

## Veja também

`Condition[].NaturalHealingChance`, `Condition[].ShockHealingChance`

PROPRIEDADE

# Propriedade: Condition[].PercentDownPerTurnHp

## Sintaxe

```
Condition[Condição-Id].PercentDownPerTurnHp
```

## Descrição

Com a propriedade PercentDownPerTurnHp é possível descobrir a quantidade de pontos de vida (em porcentagem) que o herói/inimigo perde por turno devido a essa condição.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** 0 ... 100

PROPRIEDADE

# Propriedade: Condition[].PercentDownPerTurnMp

## Sintaxe

```
Condition[Condição-Id].PercentDownPerTurnMp
```

## Descrição

Com a propriedade PercentDownPerTurnMp é possível descobrir a quantidade de pontos de magia (em porcentagem) que o herói/inimigo perde por turno devido a essa condição.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** 0 ... 100

PROPRIEDADE

# Propriedade: Condition[].PointDownPerStepHp

## Sintaxe

```
Condition[Condição-Id].PointDownPerStepHp
```

## Descrição

Com a propriedade PointDownPerStepHp é possível descobrir a quantidade de pontos de vida (em pontos) que o herói perde ao se mover pelo mapa devido a essa condição. A quantidade de passos necessária para perder pontos de vida é definida pela propriedade `StepsDownHp`. O valor desta propriedade só é válido quando se trata de uma condição de movimento.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** >= 0

PROPRIEDADE

# Propriedade: Condition[].PointDownPerStepMp

## Sintaxe

```
Condition[Condição-Id].PointDownPerStepMp
```

## Descrição

Com a propriedade PointDownPerStepMp é possível descobrir a quantidade de pontos de magia (em pontos) que o herói perde ao se mover pelo mapa devido a essa condição. A quantidade de passos necessária para perder pontos de magia é definida pela propriedade `StepsDownMp`. O valor desta propriedade só é válido quando se trata de uma condição de movimento.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** >= 0

PROPRIEDADE

# Propriedade: Condition[].PointDownPerTurnHp

## Sintaxe

```
Condition[Condição-Id].PointDownPerTurnHp
```

## Descrição

Com a propriedade PointDownPerTurnHp é possível descobrir a quantidade de pontos de vida (em pontos) que o herói/inimigo perde por turno devido a essa condição.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** >= 0

PROPRIEDADE

# Propriedade: Condition[].PointDownPerTurnMp

## Sintaxe

```
Condition[Condição-Id].PointDownPerTurnMp
```

## Descrição

Com a propriedade PointDownPerTurnMp é possível descobrir a quantidade de pontos de magia (em pontos) que o herói/inimigo perde por turno devido a essa condição.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

- **Faixa de valores:** >= 0

PROPRIEDADE

## Propriedade: Condition[].Priority

### Sintaxe

```
Condition[Condição-Id].Priority
```

### Descrição

Com a propriedade Priority é possível descobrir a prioridade de uma condição. Esse valor fica na faixa de 0 a 100, sendo que 100 representa a prioridade mais alta.

### Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** 0 ... 100

PROPRIEDADE

## Propriedade: Condition[].RecoveryStateMessage

### Sintaxe

```
Condition[Condição-Id].RecoveryStateMessage
```

### Descrição

Com a propriedade RecoveryStateMessage é possível descobrir o texto exibido quando um herói/inimigo é curado dessa condição.

### Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Condition[].ShockHealingChance

### Sintaxe

```
Condition[Condição-Id].ShockHealingChance
```

## Descrição

Com a propriedade ShockHealingChance é possível descobrir a chance (em porcentagem) de a condição indicada ser curada por um ataque físico.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** 0 ... 100

## Veja também

`Condition[].NaturalHealingTurns`, `Condition[].NaturalHealingChance`

PROPRIEDADE

# Propriedade: Condition[].StepsDownHp

## Sintaxe

```
Condition[Condição-Id].StepsDownHp
```

## Descrição

Com a propriedade StepsDownHp é possível descobrir a quantidade de passos necessária para que o herói perca pontos de vida ao se mover pelo mapa. Se esse valor for igual a 1, o jogador perde pontos de vida a cada passo dado no mapa. A quantidade de pontos de vida que o jogador perde é definida pela propriedade `PointDownPerStepHp`. O valor desta propriedade só é válido quando se trata de uma condição de movimento.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** >= 0

PROPRIEDADE

# Propriedade: Condition[].StepsDownMp

## Sintaxe

```
Condition[Condição-Id].StepsDownMp
```

## Descrição

Com a propriedade StepsDownMp é possível descobrir a quantidade de passos necessária para que o herói perca pontos de magia ao se mover pelo mapa. Se esse valor for igual a 1, o jogador perde pontos de magia a cada passo dado no mapa. A quantidade de pontos de magia que o jogador perde é definida pela propriedade `PointDownPerStepMp`. O valor desta propriedade só é válido quando se trata de uma condição de movimento.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** >= 0

PROPRIEDADE

# Propriedade: Condition[].StillInStateMessage

## Sintaxe

```
Condition[Condição-Id].StillInStateMessage
```

## Descrição

Com a propriedade StillInStateMessage é possível descobrir o texto exibido quando um herói/inimigo tem, em combate, mais um turno sob efeito dessa condição.

## Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Condition[].Type

## Sintaxe

```
Condition[Condição-Id].Type
```

## Descrição

Com a propriedade Type é possível descobrir o tipo de uma condição. Para identificar esse valor, podem ser usadas as constantes `ConditionType`.

`CONDITIONTYPE_COMBAT`: a condição é uma condição de combate (ou seja, termina no fim do combate).
`CONDITIONTYPE_MOVEMENT`: a condição é uma condição de movimento (ou seja, permanece depois do fim do combate).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

OBJETO

# Objeto: Convert

## Sintaxe

```
Convert
```

## Descrição

Com o objeto Convert é possível converter tipos de dado em outros tipos de dado. Também é possível converter ângulos, além de alterar as configurações de conversão para o separador decimal.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Angle()` | Método | Converte um ângulo de um formato de ângulo para outro. |
| `Bool()` | Método | Converte um valor para o tipo de dado Bool (= Switch). |
| `Byte()` | Método | Converte um valor para o tipo de dado Byte. |
| `DecimalComma` | Propriedade | Com a propriedade DecimalComma é possível descobrir/alterar o separador decimal usado nas conversões entre String e número. |
| `Dword()` | Método | Converte um valor para o tipo de dado Dword. |
| `Float()` | Método | Converte um valor para o tipo de dado Float (= Double). |
| `String()` | Método | Converte um valor para o tipo de dado String. |
| `Word()` | Método | Converte um valor para o tipo de dado Word. |

MÉTODO

# Método: Convert.Angle()

## Sintaxe

```
Convert.Angle(Ângulo, FormatoAtual, FormatoDesejado)
```

## Descrição

Converte um ângulo de um formato de ângulo para outro. Para indicar os formatos de ângulo, são usadas as `constantes de formato de ângulo`.

## Lista de parâmetros

### Parâmetro: `Ângulo`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** O ângulo a converter

### Parâmetro: `FormatoAtual`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... 4
- **Descrição:** O formato de ângulo em que o ângulo indicado se encontra atualmente. Para isso são usadas as `constantes de formato de ângulo`. Os seguintes formatos estão definidos:

1. `DEG`: significa Degree e representa o ângulo de 0 a 360° (360° = um círculo completo) 2. `RAD`: significa Radiant (= radiano) e representa o ângulo de 0 a 2 * π (2 * π = um círculo completo) 3. `GRAD`: significa grado e representa o ângulo de 0 a 400 (400 = um círculo completo) 4. `RPG`: significa um formato específico do RPG Maker e representa o ângulo de 0 a 256 (256 = um círculo completo)

### Parâmetro: `FormatoDesejado`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... 4
- **Descrição:** O formato de ângulo para o qual o ângulo deve ser convertido. Para isso são usadas as `constantes de formato de ângulo`. Os seguintes formatos estão definidos:

1. `DEG`: significa Degree e representa o ângulo de 0 a 360° (360° = um círculo completo)

1. `RAD`: significa Radiant (= radiano) e representa o ângulo de 0 a 2 * π (2 * π = um círculo completo) 3. `GRAD`: significa grado e representa o ângulo de 0 a 400 (400 = um círculo completo) 4. `RPG`: significa um formato específico do RPG Maker e representa o ângulo de 0 a 256 (256 = um círculo completo)

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** O ângulo convertido

## Exemplo

```
$
// Converter o ângulo 90.0 do formato de ângulo DEG
f[1] = Convert.Angle(90.0, DEG, GRAD);  // = 100.
```

## Observações adicionais

Este método pode ser bem usado para converter ângulos para o formato de ângulo do RPG Maker. (por exemplo, em `Picture[].ActionValue`, para imagens rotativas)

## Erros possíveis

`DESTINYERROR_VALUE`, `DESTINYERROR_FLOATERROR`

## Veja também

`Convert`, `Picture[].ActionValue`

MÉTODO

# Método: Convert.Bool()

## Sintaxe

```
Convert.Bool(Valor)
```

## Descrição

Converte um valor para o tipo de dado Bool (= Switch). Para isso, o valor precisa estar na faixa de 0 a 1.

> **Informação:** As conversões no DestinyScript normalmente acontecem automaticamente, por isso uma chamada explícita dos métodos de conversão normalmente não é necessária.

## Sinônimos

Switch()

## Lista de parâmetros

### Parâmetro: `Valor`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Faixa de valores:** 0 ... 1
- **Descrição:** O valor a converter

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** O valor convertido como Bool

## Exemplo

```
$
// Converter o número 1 em um Bool e guardar em s[1
s[1] = Convert.Bool(1);
```

## Erros possíveis

`DESTINYERROR_RANGE`

## Veja também

`Convert`, `Convert.Byte()`, `Convert.Word()`, `Convert.Dword()`, `Convert.Float()`, `Convert.String()`

MÉTODO

# Método: Convert.Byte()

## Sintaxe

```
Convert.Byte(Valor)
```

## Descrição

Converte um valor para o tipo de dado Byte. Para isso, o valor precisa estar na faixa de 0 a 255.

> **Informação:** As conversões no DestinyScript normalmente acontecem automaticamente, por isso uma chamada explícita dos métodos de conversão normalmente não é necessária.

## Lista de parâmetros

### Parâmetro: `Valor`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Faixa de valores:** 0 ... 255
- **Descrição:** O valor a converter

### Valor de retorno

- **Tipo de dado:** `byte`
- **Descrição:** O valor convertido como Byte

## Exemplo

```
$
// Converter o número 10 em um Byte e guardar em v[
// (aqui o tipo de dado é automaticamente convertido de volta)
v[1] = Convert.Byte(10);
```

## Erros possíveis

`DESTINYERROR_RANGE`

## Veja também

`Convert`, `Convert.Word()`, `Convert.Dword()`, `Convert.Float()`, `Convert.String()`, `Convert.Bool()`

PROPRIEDADE

# Propriedade: Convert.DecimalComma

## Sintaxe

```
Convert.DecimalComma
```

## Descrição

Com a propriedade DecimalComma é possível descobrir/alterar o separador decimal usado nas conversões entre String e número. O valor é True ou False, dependendo do idioma configurado no DestinyPatch ao iniciar o jogo.

Para o valor dessa propriedade, vale o seguinte:

Valor | Descrição

False O separador decimal é o ponto

True O separador decimal é a vírgula

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** não
- **Depende do idioma:** sim

## Exemplo

```
$
// Alterar o separador decimal para ponto
Convert.DecimalComma = False;
// Converter o número 1.5 em uma String e
a[1] = Convert.String(1.5);      // = "1.5"
// Alterar o separador decimal para vírgula
Convert.DecimalComma = True;
// Converter o número 1.5 em uma String e
a[2] = Convert.String(1.5);      // = "1,5"
```

`Convert`, `Convert.String()`

MÉTODO

# Método: Convert.Dword()

## Sintaxe

```
Convert.Dword(Valor)
```

## Descrição

Converte um valor para o tipo de dado Dword. Para isso, o valor precisa estar na faixa de -2147483648 a 2147483647.

> **Informação:** As conversões no DestinyScript normalmente acontecem automaticamente, por isso uma chamada explícita dos métodos de conversão normalmente não é necessária.

## Lista de parâmetros

### Parâmetro: `Valor`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Faixa de valores:** -2147483648 ... 2147483647
- **Descrição:** O valor a converter

### Valor de retorno

- **Tipo de dado:** `dword`
- **Descrição:** O valor convertido como Dword

## Exemplo

```
$
// Converter o número 10 em um Dword e guardar em v
v[1] = Convert.Dword(10);
```

## Erros possíveis

`DESTINYERROR_RANGE`

## Veja também

`Convert`, `Convert.Byte()`, `Convert.Word()`, `Convert.Float()`, `Convert.String()`, `Convert.Bool()`

MÉTODO

# Método: Convert.Float()

## Sintaxe

```
Convert.Float(Valor)
```

## Descrição

Converte um valor para o tipo de dado Float (= Double). Para isso, o valor precisa estar na faixa de -1.7E+308 a +1.7E+308.

> **Informação:** As conversões no DestinyScript normalmente acontecem automaticamente, por isso uma chamada explícita dos métodos de conversão normalmente não é necessária.

## Sinônimos

Double()

## Lista de parâmetros

### Parâmetro: `Valor`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Faixa de valores:** -1.7E+308 ... +1.7E+308
- **Descrição:** O valor a converter

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** O valor convertido como Float

## Exemplo

```
$
// Converter o número 10 em um Float e guardar em f
f[1] = Convert.Float(10);
```

## Erros possíveis

`DESTINYERROR_RANGE`

## Veja também

`Convert`, `Convert.Byte()`, `Convert.Word()`, `Convert.Dword()`, `Convert.String()`, `Convert.Bool()`

MÉTODO

# Método: Convert.String()

## Sintaxe

```
Convert.String(Valor)
```

## Descrição

Converte um valor para o tipo de dado String.

> **Informação:** As conversões no DestinyScript normalmente acontecem automaticamente, por isso uma chamada explícita dos métodos de conversão normalmente não é necessária.

## Lista de parâmetros

### Parâmetro: `Valor`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** O valor a converter

### Valor de retorno

- **Tipo de dado:** `string`
- **Descrição:** O valor convertido como String

## Exemplo

```
$
// Converter o número 10 em uma String e guardar em
a[1] = Convert.String(10);    // "10"
```

## Veja também

`Convert`, `Convert.Byte()`, `Convert.Word()`, `Convert.Dword()`, `Convert.Float()`, `Convert.Bool()`, `Convert.DecimalComma`

MÉTODO

# Método: Convert.Word()

## Sintaxe

```
Convert.Word(Valor)
```

## Descrição

Converte um valor para o tipo de dado Word. Para isso, o valor precisa estar na faixa de -32768 a 32767.

> **Informação:** As conversões no DestinyScript normalmente acontecem automaticamente, por isso uma chamada explícita dos métodos de conversão normalmente não é necessária.

## Lista de parâmetros

### Parâmetro: `Valor`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Faixa de valores:** -32768 ... 32767
- **Descrição:** O valor a converter

### Valor de retorno

- **Tipo de dado:** `word`
- **Descrição:** O valor convertido como Word

## Exemplo

```
$
// Converter o número 10 em um Word e guardar em v[
// (aqui o tipo de dado é automaticamente convertido de volta)
v[1] = Convert.Word(10);
```

## Erros possíveis

`DESTINYERROR_RANGE`

## Veja também

`Convert`, `Convert.Byte()`, `Convert.Dword()`, `Convert.Float()`, `Convert.String()`, `Convert.Bool()`

OBJETO

# Objeto: d

## Sintaxe

```
d
```

## Descrição

O objeto d serve para tornar acessíveis os Dwords (números inteiros, como variáveis) da Destiny.dll. A diferença em relação ao objeto v é que as variáveis (Dwords) do objeto v são gerenciadas pelo RPG_RT, enquanto os Dwords do objeto d são gerenciados pela Destiny.dll.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Count` | Propriedade | Devolve a quantidade de Dwords no objeto d. |
| `ReAlloc()` | Método | Com o método ReAlloc é possível aumentar/diminuir a quantidade de Dwords no objeto d. |
| `d[]` | Propriedade | Permite o acesso a um determinado Dword do objeto d. |

## Veja também

`a`, `f`, `v`, `s`

PROPRIEDADE

# Propriedade: d.Count

## Sintaxe

```
d.Count
```

## Descrição

Devolve a quantidade de Dwords no objeto `d`. Isso pode ser usado para fazer um laço sobre todos os Dwords.

> **Informação:** A quantidade de Dwords disponível ao iniciar o jogo é definida pelo DestinyPatcher.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** >= 0

## Exemplo

```
$
// Definir todos os Dwords como 0
for(v[1] = 1; v[1] <= d.Count; v[1]++)
  d[v[1]] = 0;
next;
```

## Veja também

`d`, `d[]`, `d.ReAlloc()`

MÉTODO

# Método: d.ReAlloc()

## Sintaxe

```
d.ReAlloc(Quantidade)
```

## Descrição

Com o método ReAlloc é possível aumentar/diminuir a quantidade de Dwords no objeto `d`.

> **Informação:** Se a quantidade de Dwords informada for grande demais, é lançado o erro `DESTINYERROR_MEMORY`. O conteúdo original do objeto `d` é mantido.

> **Informação:** Ao aumentar a quantidade de Dwords, os novos Dwords são definidos como 0.

## Lista de parâmetros

### Parâmetro: `Quantidade`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 0
- **Descrição:** A nova quantidade de Dwords no objeto `d`. Esse valor precisa ser maior/igual a 0. Com um número grande demais, a chamada deste método falha.

## Exemplo

```
$
// Aumentar em 1 a quantidade máxima de Dwords
d.ReAlloc(d.Count + 1);
// Atribuir ao novo Dword o valor 5 (depois do
d[d.Count] = 5;
```

## Erros possíveis

`DESTINYERROR_VALUE`, `DESTINYERROR_MEMORY`

## Veja também

`d`, `d[]`, `d.Count`

PROPRIEDADE

# Propriedade: d[]

## Sintaxe

```
d[Índice]
```

## Descrição

Permite o acesso a um determinado Dword do objeto `d`.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

## Lista de parâmetros

### Parâmetro: `Índice`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... d.Count

- **Descrição:** O índice do Dword. Precisa estar na faixa de 1 ... `d.Count`.

## Exemplo

```
$
// Definir o valor do primeiro Dword como 1
d[1] = 1;
```

## Erros possíveis

`DESTINYERROR_ARRAYBOUND`

## Veja também

`d`, `d.Count`, `d.ReAlloc()`

OBJETO

# Objeto: Destiny

## Sintaxe

```
Destiny
```

## Descrição

Com o objeto Destiny é possível acessar a própria Destiny.dll. É possível consultar as informações de versão da Destiny.dll e definir o `idioma`. Também é possível `salvar/carregar` as áreas de memória da Destiny.dll (`a`, `d`, `f`). Além disso, é possível `chamar o DestinyDebugger` e verificar se determinados métodos/propriedades estão `implementados`.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Debugger` | Objeto | Com o objeto Debugger é possível acessar o DestinyDebugger. |
| `DllVersionMajor` | Propriedade | Devolve a parte inteira do número de versão disponível da Destiny.dll. |
| `DllVersionMinor` | Propriedade | Devolve a parte decimal do número de versão disponível da Destiny.dll. |
| `Implemented()` | Método | Com o método Implemented é possível verificar se um determinado método ou propriedade está implementado. |
| `Language` | Propriedade | Com a propriedade Language é possível ler/gravar o idioma configurado da Destiny.dll. |
| `Load()` | Método | Com o método Load, as áreas de memória da Destiny.dll são carregadas de um determinado slot de save (= arquivo no disco). |
| `Save()` | Método | Com o método Save, as áreas de memória (indicadas) da Destiny.dll são salvas em um determinado slot de save (= arquivo no disco). |
| `VersionMajor` | Propriedade | Devolve a parte inteira do número de versão da Destiny.dll em uso. |
| `VersionMinor` | Propriedade | Devolve a parte decimal do número de versão da Destiny.dll em uso. |

OBJETO

# Objeto: Destiny.Debugger

## Sintaxe

```
Destiny.Debugger
```

## Descrição

Com o objeto Debugger é possível acessar o DestinyDebugger.

## Exemplo

```
$
// Abrir o DestinyDebugger
Destiny.Debugger.Open();
```

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Open()` | Método | Com o método Open é possível abrir o DestinyDebugger. |
| `Output` | Objeto | Através do objeto Output é possível emitir saídas de depuração. |

## Veja também

`Destiny.Debugger.Open()`

MÉTODO

# Método: Destiny.Debugger.Open()

## Sintaxe

```
Destiny.Debugger.Open()
```

## Descrição

Com o método Open é possível abrir o DestinyDebugger. No DestinyDebugger é possível visualizar em tempo de execução todas as propriedades e constantes do DestinyScript.

## Exemplo

```
$
// Abrir o DestinyDebugger
Destiny.Debugger.Open();
```

## Veja também

`Destiny`, `Destiny.Debugger`, `Beschreibung des DestinyDebuggers`

OBJETO

# Objeto: Destiny.Debugger.Output

## Sintaxe

```
Destiny.Debugger.Output
```

## Descrição

Através do objeto Output é possível emitir saídas de depuração.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Clear()` | Método | Com o método Clear é possível limpar o conteúdo da saída de depuração. |
| `Print()` | Método | Com o método Print é possível anexar uma linha à saída de depuração. |
| `Show()` | Método | Com o método Show é possível exibir a saída de depuração. |
| `Text` | Propriedade | Com a propriedade Text é possível ler/gravar o conteúdo atual da saída de depuração. |

## Veja também

`Destiny`, `Destiny.Debugger`, `Beschreibung der Debug-Ausgabe`

MÉTODO

# Método: Destiny.Debugger.Output.Clear()

## Sintaxe

```
Destiny.Debugger.Output.Clear()
```

## Descrição

Com o método Clear é possível limpar o conteúdo da saída de depuração.

## Exemplo

```
$

// Limpar o conteúdo da saída de depuração
Destiny.Debugger.Output.Clear();

// Inserir algumas linhas na saída de depuração
Destiny.Debugger.Output.Print("Zeile 1");
Destiny.Debugger.Output.Print("Zeile 2");
```

```
Destiny.Debugger.Output.Print("Zeile 3");

// Exibir a saída de depuração
Destiny.Debugger.Output.Show();
```

## Veja também

`Destiny`, `Destiny.Debugger`, `Destiny.Debugger.Output`, `Destiny.Debugger.Output.Print()`, `Destiny.Debugger.Output.Show()`, `Beschreibung der Debug-Ausgabe`

MÉTODO

# Método: Destiny.Debugger.Output.Print()

## Sintaxe

```
Destiny.Debugger.Output.Print(Linha)
```

## Descrição

Com o método Print é possível anexar uma linha à saída de depuração. Uma quebra de linha é adicionada automaticamente.

## Lista de parâmetros

### Parâmetro: `Linha`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** O texto a ser anexado.

## Exemplo

```
$

// Limpar o conteúdo da saída de depuração
Destiny.Debugger.Output.Clear();

// Inserir algumas linhas na saída de depuração
Destiny.Debugger.Output.Print("Zeile 1");
Destiny.Debugger.Output.Print("Zeile 2");
Destiny.Debugger.Output.Print("Zeile 3");

// Exibir a saída de depuração
Destiny.Debugger.Output.Show();
```

## Veja também

`Destiny`, `Destiny.Debugger`, `Destiny.Debugger.Output`, `Destiny.Debugger.Output.Clear()`, `Destiny.Debugger.Output.Show()`, `Beschreibung der Debug-Ausgabe`

MÉTODO

# Método: Destiny.Debugger.Output.Show()

## Sintaxe

```
Destiny.Debugger.Output.Show()
```

## Descrição

Com o método Show é possível exibir a saída de depuração. Para isso, a janela do RPG_RT é minimizada. Depois de fechar a saída de depuração, a janela do RPG_RT é restaurada.

## Exemplo

```
$

// Limpar o conteúdo da saída de depuração
Destiny.Debugger.Output.Clear();

// Inserir algumas linhas na saída de depuração
Destiny.Debugger.Output.Print("Zeile 1");
Destiny.Debugger.Output.Print("Zeile 2");
Destiny.Debugger.Output.Print("Zeile 3");

// Exibir a saída de depuração
Destiny.Debugger.Output.Show();
```

## Veja também

`Destiny`, `Destiny.Debugger`, `Destiny.Debugger.Output`, `Destiny.Debugger.Output.Clear()`, `Destiny.Debugger.Output.Print()`, `Beschreibung der Debug-Ausgabe`

PROPRIEDADE

# Propriedade: Destiny.Debugger.Output.Text

## Sintaxe

```
Destiny.Debugger.Output.Text
```

## Descrição

Com a propriedade Text é possível ler/gravar o conteúdo atual da saída de depuração. O conteúdo costuma ser anexado pelo método `Print`.

## Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** não

## Exemplo

```
$

// Definir o conteúdo da saída de depuração como "Ol
Destiny.Debugger.Output.Text = "Hallo";

// Exibir a saída de depuração
Destiny.Debugger.Output.Show();
```

## Veja também

`Destiny`, `Destiny.Debugger`, `Destiny.Debugger.Output`, `Destiny.Debugger.Output.Show()`, `Beschreibung der Debug-Ausgabe`

PROPRIEDADE

# Propriedade: Destiny.DllVersionMajor

## Sintaxe

```
Destiny.DllVersionMajor
```

## Descrição

Devolve a parte inteira do número de versão disponível da Destiny.dll.

> **Informação:** A distinção entre o número de versão da Destiny.dll em uso e o número de versão da Destiny.dll disponível foi pensada para modos de compatibilidade, mas ainda não é usada. Para verificar o número de versão da Destiny.dll, o ideal é sempre usar `Destiny.VersionMajor` e `Destiny.VersionMinor`.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

## Exemplo

```
$
// O número de versão disponível da Destiny.dll
a[1] = Destiny.DllVersionMajor .. "." .. Destiny.
```

## Veja também

`Destiny`, `Destiny.DllVersionMinor`

PROPRIEDADE

# Propriedade: Destiny.DllVersionMinor

## Sintaxe

```
Destiny.DllVersionMinor
```

## Descrição

Devolve a parte decimal do número de versão disponível da Destiny.dll.

> **Informação:** A distinção entre o número de versão da Destiny.dll em uso e o número de versão da Destiny.dll disponível foi pensada para modos de compatibilidade, mas ainda não é usada. Para verificar o número de versão da Destiny.dll, o ideal é sempre usar `Destiny.VersionMajor` e `Destiny.VersionMinor`.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

## Exemplo

```
$
// O número de versão disponível da Destiny.dll
a[1] = Destiny.DllVersionMajor .. "." .. Destiny.
```

## Veja também

`Destiny`, `Destiny.DllVersionMajor`

MÉTODO

# Método: Destiny.Implemented()

## Sintaxe

```
Destiny.Implemented(Chamada)
```

## Descrição

Com o método Implemented é possível verificar se um determinado método ou propriedade está implementado. Isso é útil para scripts que precisam obrigatoriamente de determinados métodos/propriedades e, caso contrário, não devem rodar, ou devem executar outra coisa.

## Lista de parâmetros

### Parâmetro: `Chamada`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** Indica a chamada a verificar, sem parâmetros. Os parênteses (colchetes e parênteses comuns) precisam ser informados nesta chamada.

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** Indica se o método/propriedade está disponível (True) ou não (False).

## Exemplo

```
$
// Verificar se o método Destiny.Implemented() est
if(Destiny.Implemented("Destiny.Implemented()"))
  // ... (é executado)
endif;

// Verificar se a propriedade Destiny.Language est
if(Destiny.Implemented("Destiny.Language"))
  // ... (é executado)
endif;

// Verificar se a propriedade Error[].Message está
if(Destiny.Implemented("Error[].Message"))
  // ... (é executado)
endif;

// Verificar se a propriedade Error.Message está d
if(Destiny.Implemented("Error.Message"))
  // ... (não é executado, pois o objeto Error não
endif;
```

## Veja também

`Destiny`, `Destiny.Language`, `Error[].Message`, `Destiny.VersionMajor`, `Destiny.VersionMinor`

PROPRIEDADE

# Propriedade: Destiny.Language

## Sintaxe

```
Destiny.Language
```

## Descrição

Com a propriedade Language é possível ler/gravar o idioma configurado da Destiny.dll. Ao alterar essa propriedade, `Convert.DecimalComma` também é ajustada.

> **Informação:** Para alterar/comparar essa propriedade, deveriam ser usadas as `constantes de idioma` (veja o exemplo).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não
- **Faixa de valores:** 0 ... 1

## Exemplo

```
$
// Definir o idioma como inglês
Destiny.Language = LANGUAGE_ENGLISH;
// Consultar a primeira mensagem de erro e guardar em
a[1] = Error[1].Message;  // "The syntax is inval
// Definir o idioma como alemão
Destiny.Language = LANGUAGE_GERMAN;
// Consultar a primeira mensagem de erro e guardar em
a[2] = Error[1].Message;  // "Die Syntax ist ungü
```

## Veja também

`Destiny`, `Error[].Message`

MÉTODO

# Método: Destiny.Load()

## Sintaxe

```
Destiny.Load(Slot)
```

## Descrição

Com o método Load, as áreas de memória da Destiny.dll são carregadas de um determinado slot de save (= arquivo no disco). O arquivo do slot fica na pasta do jogo e, por padrão, tem o nome SaveXX.dsd, sendo que XX deve ser substituído pelo slot no formato de dois dígitos. Quais áreas de memória são carregadas depende de quais saves foram salvos anteriormente. Um save salvo do RPG_RT não é carregado com isso. Para carregar um save salvo do RPG_RT, pode-se usar o método `Load()` do objeto `Game`. Se as áreas de memória não puderem ser carregadas, um erro é lançado; as áreas de memória anteriores não são substituídas nesse caso.

> **Informação:** Se tanto o save do jogo quanto as áreas de memória da Destiny.dll devem ser carregados, primeiro devem ser carregadas as áreas de memória da Destiny.dll e depois o save do jogo, pois, caso contrário, depois de carregar o save do jogo, os comandos de DestinyScript seguintes talvez não sejam mais executados.

## Lista de parâmetros

### Parâmetro: `Slot`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** Indica o número do slot de save

## Exemplo

```
$
// Guardar o número 5.0 em f[1]
f[1] = 5.0;

// Salvar as áreas de memória da Destiny.dll no primeir
Destiny.Save(1);
// Salvar apenas as Strings e os Dwords da Destiny.dll
Destiny.Save(2, "AD");

// Guardar o número 0.0 em f[1]
f[1] = 0.0;

// Carregar as áreas de memória do segundo slot
Destiny.Load(2);     // f[1] = 0.0
// Carregar as áreas de memória do primeiro slot
Destiny.Load(1);     // f[1] = 5.0
```

## Erros possíveis

`DESTINYERROR_VALUE`, `DESTINYERROR_MEMORY`, `DESTINYERROR_CANTACCESSFILE`

## Veja também

`Destiny`, `Destiny.Save()`, `Game.Load()`

MÉTODO

# Método: Destiny.Save()

## Sintaxe

```
Destiny.Save(Slot, [Áreas de memória])
```

## Descrição

Com o método Save, as áreas de memória (indicadas) da Destiny.dll são salvas em um determinado slot de save (= arquivo no disco). O arquivo do slot fica na pasta do jogo e, por padrão, tem o nome SaveXX.dsd, sendo que XX deve ser substituído pelo slot no formato de dois dígitos. O save atual do RPG_RT não é salvo com isso. Para salvar o save atual do RPG_RT, pode-se usar o método `Save()` do objeto `Game`. Se as áreas de memória não puderem ser salvas, um erro é lançado.

> **Informação:** Se tanto o save atual quanto as áreas de memória da Destiny.dll devem ser salvos, primeiro deveriam ser salvas as áreas de memória da Destiny.dll, pois, caso contrário, ao carregar o save, as áreas de memória da Destiny.dll só seriam salvas depois, o que já seria tarde demais.

## Lista de parâmetros

### Parâmetro: `Slot`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** Indica o número do slot de save

### Parâmetro: `Áreas de memória`

- **Tipo de dado:** `string`
- **Opcional:** sim
- **Valor padrão:** "ADFLE"
- **Descrição:** Indica as áreas de memória a salvar em uma String. Cada letra informada representa uma área de memória. Não é necessário diferenciar maiúsculas de minúsculas. Se forem informadas outras letras, elas são ignoradas. São possíveis as seguintes indicações:

Letra | Descrição

A Salva todas as Strings (`a[]`)

D Salva todos os Dwords (`d[]`)

F Salva todos os Floats (`f[]`)

L Salva o idioma configurado da Destiny.dll (`Destiny.Language`)

E Salva as opções de erro configuradas (`Errors` e `Error[]`)

## Exemplo

```
$
// Guardar o número 5.0 em f[1]
f[1] = 5.0;

// Salvar as áreas de memória da Destiny.dll no primeir
Destiny.Save(1);
// Salvar apenas as Strings e os Dwords da Destiny.dll
Destiny.Save(2, "AD");

// Guardar o número 0.0 em f[1]
f[1] = 0.0;

// Carregar as áreas de memória do segundo slot
Destiny.Load(2);     // f[1] = 0.0
// Carregar as áreas de memória do primeiro slot
Destiny.Load(1);     // f[1] = 5.0
```

## Erros possíveis

`DESTINYERROR_VALUE`, `DESTINYERROR_CANTACCESSFILE`

## Veja também

`Destiny`, `Destiny.Load()`, `Game.Save()`

PROPRIEDADE

# Propriedade: Destiny.VersionMajor

## Sintaxe

```
Destiny.VersionMajor
```

## Descrição

Devolve a parte inteira do número de versão da Destiny.dll em uso.

> **Informação:** A distinção entre o número de versão da Destiny.dll em uso e o número de versão da Destiny.dll disponível foi pensada para modos de compatibilidade, mas ainda não é usada. Para verificar o número de versão da Destiny.dll, o ideal é sempre usar `Destiny.VersionMajor` e `Destiny.VersionMinor`.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

## Exemplo

```
$
// Montar o número de versão da Destiny.dll
a[1] = Destiny.VersionMajor .. "." .. Destiny.Ver
```

## Veja também

`Destiny`, `Destiny.VersionMinor`

PROPRIEDADE

# Propriedade: Destiny.VersionMinor

## Sintaxe

```
Destiny.VersionMinor
```

## Descrição

Devolve a parte decimal do número de versão da Destiny.dll em uso.

> **Informação:** A distinção entre o número de versão da Destiny.dll em uso e o número de versão da Destiny.dll disponível foi pensada para modos de compatibilidade, mas ainda não é usada. Para verificar o número de

> versão da Destiny.dll, o ideal é sempre usar `Destiny.VersionMajor` e `Destiny.VersionMinor`.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

## Exemplo

```
$
// Montar o número de versão da Destiny.dll
a[1] = Destiny.VersionMajor .. "." .. Destiny.Ver
```

## Veja também

`Destiny`, `Destiny.VersionMajor`

OBJETO

# Objeto: Directory

## Sintaxe

```
Directory
```

## Descrição

Com o objeto Directory é possível editar estruturas de pastas e arquivos, além de buscar em pastas. Assim como no objeto `File`, por motivos de segurança só são permitidos acessos dentro da pasta do jogo.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Atributos de arquivos/diretórios` | Categoria | Nesta categoria estão reunidos todos os métodos para ler e gravar os atributos de arquivos e pastas. |
| `Operações com arquivos/diretórios` | Categoria | Nesta categoria estão reunidas todas as operações para editar arquivos e pastas como itens do sistema de arquivos. |
| `Buscas em diretórios` | Categoria | Nesta categoria estão reunidos todos os métodos e propriedades necessários para buscar arquivos e subpastas dentro de uma pasta. |

CATEGORIA

# Categoria: Atributos de arquivos/diretórios

## Descrição

Nesta categoria estão reunidos todos os métodos para ler e gravar os atributos de arquivos e pastas.

Esses atributos fazem parte do sistema de arquivos; exemplos conhecidos são "Somente leitura" e "Oculto".

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `GetAttributes()` | Método | Com o método GetAttributes são consultados os atributos de um arquivo/uma pasta existente na pasta do jogo (ou em uma subpasta, no caso de caminho relativo). |
| `SetAttributes()` | Método | Com o método SetAttributes são definidos os atributos de um arquivo/uma pasta existente na pasta do jogo (ou em uma subpasta, no caso de caminho relativo). |

MÉTODO

# Método: Directory.GetAttributes()

## Sintaxe

```
Directory.GetAttributes(Nome)
```

## Descrição

Com o método GetAttributes são consultados os atributos de um arquivo/uma pasta existente na pasta do jogo (ou em uma subpasta, no caso de caminho relativo). Só é permitido consultar atributos de arquivos/pastas dentro da pasta do jogo (e de suas subpastas). Acessos fora da pasta do jogo não são permitidos!

Esses atributos fazem parte do sistema de arquivos; exemplos conhecidos são "Somente leitura" e "Oculto".

## Lista de parâmetros

### Parâmetro: `Nome`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** Indica o nome do arquivo/da pasta cujos atributos devem ser consultados. Para indicar uma subpasta, podem ser usadas a barra e/ou a barra invertida.

### Valor de retorno

- **Tipo de dado:** `dword`
- **Descrição:** Os atributos do arquivo/da pasta. Para identificação, podem ser usadas as `constantes de atributo de arquivo`. Elas valem para arquivos e pastas e podem ser combinadas pelo operador binário OU. O atributo `FILE_ATTRIBUTE_DIRECTORY` identifica uma pasta (e só é válido em pastas).

## Exemplo

```
$
// Consultar os atributos do arquivo "RPG_RT.exe"
v[1] = Directory.GetAttributes("RPG_RT.exe");

// Verificar se o arquivo está oculto no momento e
if(v[1] & FILE_ATTRIBUTE_HIDDEN)
  a[1] = "Die RPG_RT.exe ist momentan versteckt.
else
  a[1] = "Die RPG_RT.exe ist sichtbar. Wir haben
endif;
```

```
// Remover o atributo "Oculto" do RPG_RT.exe
Directory.SetAttributes("RPG_RT.exe", v[1] & ~FIL
```

## Erros possíveis

`DESTINYERROR_MEMORY`, `DESTINYERROR_CANTREADATTRIBUTES`, `DESTINYERROR_NOPERMISSION`, `DESTINYERROR_CANTRESOLVEPATH`

## Veja também

`Directory`, `Directory.SetAttributes()`

MÉTODO

# Método: Directory.SetAttributes()

## Sintaxe

```
Directory.SetAttributes(Nome, Atributos)
```

## Descrição

Com o método SetAttributes são definidos os atributos de um arquivo/uma pasta existente na pasta do jogo (ou em uma subpasta, no caso de caminho relativo). Só é permitido definir atributos de arquivos/pastas dentro da pasta do jogo (e de suas subpastas). Acessos fora da pasta do jogo não são permitidos!

Esses atributos fazem parte do sistema de arquivos; exemplos conhecidos são "Somente leitura" e "Oculto".

> **Informação:** Para adicionar um atributo aos atributos existentes, usa-se o operador binário OU. (por exemplo, Atributos | FILE_ATTRIBUTE_HIDDEN)

Para inverter um atributo nos atributos existentes, usa-se o operador binário OU EXCLUSIVO. (por exemplo, Atributos ^ FILE_ATTRIBUTE_HIDDEN) Para remover um atributo dos atributos existentes, faz-se uma operação binária E com o complemento de um (NÃO binário) da respectiva constante. (por exemplo, Atributos & ~FILE_ATTRIBUTE_HIDDEN)

## Lista de parâmetros

### Parâmetro: `Nome`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** Indica o nome do arquivo/da pasta cujos atributos devem ser definidos. Para indicar uma subpasta, podem ser usadas a barra e/ou a barra invertida.

### Parâmetro: `Atributos`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** Os novos atributos do arquivo/da pasta. Para defini-los, podem ser usadas as `constantes de atributo de arquivo`. Elas valem para arquivos e pastas e podem ser combinadas pelo operador binário OU.

O atributo `FILE_ATTRIBUTE_DIRECTORY` identifica uma pasta (e só é válido em pastas; portanto, não pode ser definido em arquivos nem removido de pastas).

## Exemplo

```
$
// Consultar os atributos do arquivo "RPG_RT.exe"
v[1] = Directory.GetAttributes("RPG_RT.exe");

// Verificar se o arquivo está oculto no momento e
if(v[1] & FILE_ATTRIBUTE_HIDDEN)
  a[1] = "Die RPG_RT.exe ist momentan versteckt.
else
  a[1] = "Die RPG_RT.exe ist sichtbar. Wir haben
endif;

// Remover o atributo "Oculto" do RPG_RT.exe
Directory.SetAttributes("RPG_RT.exe", v[1] & ~FIL
```

## Erros possíveis

`DESTINYERROR_MEMORY`, `DESTINYERROR_CANTWRITEATTRIBUTES`, `DESTINYERROR_NOPERMISSION`, `DESTINYERROR_CANTRESOLVEPATH`

## Veja também

`Directory`, `Directory.GetAttributes()`

CATEGORIA

# Categoria: Operações com arquivos/diretórios

## Descrição

Nesta categoria estão reunidas todas as operações para editar arquivos e pastas como itens do sistema de arquivos. Isso inclui criar, renomear, mover e remover pastas, bem como copiar, renomear, mover e excluir arquivos. Para criar arquivos, pode ser usado o objeto `File`.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `CopyFile()` | Método | Com o método CopyFile é copiado um arquivo existente na pasta do jogo (ou em uma subpasta, no caso de caminho relativo). |
| `CreateDir()` | Método | Com o método CreateDir é criada uma nova pasta na pasta do jogo (ou em uma subpasta, no caso de caminho relativo). |
| `DeleteFile()` | Método | Com o método DeleteFile é excluído um arquivo existente na pasta do jogo (ou em uma subpasta, no caso de caminho relativo). |
| `RemoveDir()` | Método | Com o método RemoveDir é removida uma pasta existente na pasta do jogo (ou em uma subpasta, no caso de caminho relativo). |
| `Rename()` | Método | Com o método Rename, um arquivo/uma pasta existente na pasta do jogo (ou em uma subpasta, no caso de caminho relativo) é renomeado(a) ou movido(a). |

MÉTODO

# Método: Directory.CopyFile()

## Sintaxe

```
Directory.CopyFile(Arquivo de origem, Arquivo de destino)
```

## Descrição

Com o método CopyFile é copiado um arquivo existente na pasta do jogo (ou em uma subpasta, no caso de caminho relativo). A cópia é criada com o nome de arquivo indicado na pasta do jogo (ou na respectiva subpasta, no caso de caminho relativo). Só é permitido copiar arquivos dentro da pasta do jogo (e de suas subpastas), tanto para o arquivo de origem quanto para o de destino. Acessos fora da pasta do jogo não são permitidos!

## Lista de parâmetros

### Parâmetro: `Arquivo de origem`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** Indica o nome do arquivo a copiar. Para indicar uma subpasta, podem ser usadas a barra e/ou a barra invertida.

### Parâmetro: `Arquivo de destino`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** Indica o nome do arquivo de destino (ou seja, o nome/caminho da cópia). Para as restrições do nome do arquivo, valem as restrições usuais de nomes de arquivo da Microsoft. Assim, os caracteres especiais \\ / : ? * " < > | não podem aparecer no nome do arquivo (barra e barra invertida, porém, são permitidas para indicar subpastas). Também não podem aparecer espaços à direita do nome do arquivo.

## Exemplo

```
$
// Copiar o arquivo "RPG_RT.exe" da pasta do jogo
Directory.CopyFile("RPG_RT.exe", "RPG_RT2.exe");
// A cópia criada "RPG_RT2.exe" na pasta do jogo
Directory.DeleteFile("RPG_RT2.exe");
```

## Erros possíveis

`DESTINYERROR_MEMORY`, `DESTINYERROR_CANTCOPYFILE`, `DESTINYERROR_NOPERMISSION`, `DESTINYERROR_CANTRESOLVEPATH`

## Veja também

`Directory`, `Directory.Rename()`, `File[].Open()`, `Directory.DeleteFile()`

MÉTODO

# Método: Directory.CreateDir()

## Sintaxe

```
Directory.CreateDir(Nome do diretório)
```

## Descrição

Com o método CreateDir é criada uma nova pasta na pasta do jogo (ou em uma subpasta, no caso de caminho relativo). Só é permitido criar pastas dentro da pasta do jogo (e de suas subpastas). Acessos fora da pasta do jogo não são permitidos!

## Lista de parâmetros

### Parâmetro: `Nome do diretório`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** Indica o nome da pasta (incluindo caminho relativo, se for o caso). Para as restrições do nome da pasta, valem as restrições usuais de nomes de arquivo da Microsoft. Assim, os caracteres especiais \\ / : ? * " < > | não podem aparecer no nome da pasta (barra e barra invertida, porém, são permitidas para indicar subpastas). Também não podem aparecer espaços à direita do nome da pasta.

## Exemplo

```
$
// Uma pasta chamada "Test" na pasta do jogo
Directory.CreateDir("Test");
// Uma pasta chamada "Hallo" como subpasta
Directory.CreateDir("Test\Hallo");
```

## Erros possíveis

`DESTINYERROR_MEMORY`, `DESTINYERROR_CANTCREATEDIR`, `DESTINYERROR_NOPERMISSION`, `DESTINYERROR_CANTRESOLVEPATH`

## Veja também

`Directory`, `Directory.RemoveDir()`, `Directory.Rename()`

MÉTODO

# Método: Directory.DeleteFile()

## Sintaxe

```
Directory.DeleteFile(Nome do arquivo)
```

## Descrição

Com o método DeleteFile é excluído um arquivo existente na pasta do jogo (ou em uma subpasta, no caso de caminho relativo). Só é permitido excluir arquivos dentro da pasta do jogo (e de suas subpastas). Acessos fora da pasta do jogo não são permitidos!

## Lista de parâmetros

### Parâmetro: `Nome do arquivo`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** Indica o nome do arquivo a excluir. Para indicar uma subpasta, podem ser usadas a barra e/ou a barra invertida.

## Exemplo

```
$
// Copiar o arquivo "RPG_RT.exe" da pasta do jogo
Directory.CopyFile("RPG_RT.exe", "RPG_RT2.exe");
// A cópia criada "RPG_RT2.exe" na pasta do jogo
Directory.DeleteFile("RPG_RT2.exe");
```

## Erros possíveis

`DESTINYERROR_MEMORY`, `DESTINYERROR_CANTDELETEFILE`, `DESTINYERROR_NOPERMISSION`, `DESTINYERROR_CANTRESOLVEPATH`

## Veja também

`Directory`, `Directory.Rename()`, `Directory.CopyFile()`

MÉTODO

# Método: Directory.RemoveDir()

## Sintaxe

```
Directory.RemoveDir(Nome do diretório)
```

## Descrição

Com o método RemoveDir é removida uma pasta existente na pasta do jogo (ou em uma subpasta, no caso de caminho relativo). Essa pasta precisa estar vazia (ou seja, não pode conter arquivos nem subpastas). Só é permitido remover pastas dentro da pasta do jogo (e de suas subpastas). Acessos fora da pasta do jogo não são permitidos!

## Lista de parâmetros

### Parâmetro: `Nome do diretório`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** Indica o nome da pasta (incluindo caminho relativo, se for o caso). Para indicar uma subpasta, podem ser usadas a barra e/ou a barra invertida.

## Exemplo

```
$
// Uma pasta chamada "Hallo" (como subpasta
Directory.RemoveDir("Test\Hallo");
// Uma pasta chamada "Test" (na pasta do jogo
Directory.RemoveDir("Test");
```

## Erros possíveis

`DESTINYERROR_MEMORY`, `DESTINYERROR_CANTREMOVEDIR`, `DESTINYERROR_NOPERMISSION`, `DESTINYERROR_CANTRESOLVEPATH`

## Veja também

`Directory`, `Directory.CreateDir()`, `Directory.Rename()`, `Directory.DeleteFile()`

MÉTODO

# Método: Directory.Rename()

## Sintaxe

```
Directory.Rename(Nome antigo, Novo nome)
```

## Descrição

Com o método Rename, um arquivo/uma pasta existente na pasta do jogo (ou em uma subpasta, no caso de caminho relativo) é renomeado(a) ou movido(a). Para que um arquivo/uma pasta seja movido(a) em vez de renomeado(a), o caminho indicado no novo nome precisa ser diferente do caminho indicado no nome antigo (veja o exemplo). Só é permitido renomear ou mover arquivos/pastas dentro da pasta do jogo (e de suas subpastas). Acessos fora da pasta do jogo não são permitidos!

## Lista de parâmetros

### Parâmetro: `Nome antigo`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** Indica o nome atual do arquivo/da pasta. Para indicar uma subpasta, podem ser usadas a barra e/ou a barra invertida.

### Parâmetro: `Novo nome`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** Indica o novo nome do arquivo/da pasta. Para as restrições do nome do arquivo/da pasta, valem as restrições usuais de nomes de arquivo da Microsoft. Assim, os caracteres especiais \\ / : ? * " < > | não podem aparecer no nome do arquivo/da pasta (barra e barra invertida, porém, são permitidas para indicar subpastas). Também não podem aparecer espaços à direita do nome do arquivo/da pasta. Se aqui for indicado um caminho diferente do nome antigo, o arquivo/a pasta é movido(a).

## Exemplo

```
$
// Mover o arquivo "RPG_RT.ini" da pasta do jogo para
Directory.Rename("RPG_RT.ini", "Blubb.ini");
// Mover o arquivo "Blubb.ini" da pasta do jogo para
Directory.Rename("Blubb.ini", "Picture\Blubb.ini"
// Mover o arquivo "Blubb.ini" da pasta Picture para
Directory.Rename("Picture\Blubb.ini", "RPG_RT.ini
```

## Erros possíveis

`DESTINYERROR_MEMORY`, `DESTINYERROR_CANTRENAMEFILE`, `DESTINYERROR_NOPERMISSION`, `DESTINYERROR_CANTRESOLVEPATH`

## Veja também

`Directory`, `Directory.CreateDir()`, `Directory.RemoveDir()`, `Directory.CopyFile()`, `File[].Open()`

CATEGORIA

# Categoria: Buscas em diretórios

## Descrição

Nesta categoria estão reunidos todos os métodos e propriedades necessários para buscar arquivos e subpastas dentro de uma pasta.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `FindClose()` | Método | Com o método FindClose é possível encerrar a última busca em pasta iniciada. |
| `FindCloseAll()` | Método | Com o método FindCloseAll é possível encerrar todas as buscas em pasta atualmente iniciadas. |
| `FindFirst()` | Método | Com o método FindFirst é possível iniciar a busca em uma pasta. |
| `FindLevel` | Propriedade | Com a propriedade FindLevel é possível descobrir a quantidade de buscas em pasta atualmente iniciadas. |
| `FindNext()` | Método | Com o método FindNext é possível continuar a última busca em pasta iniciada. |

MÉTODO

# Método: Directory.FindClose()

## Sintaxe

```
Directory.FindClose()
```

## Descrição

Com o método FindClose é possível encerrar a última busca em pasta iniciada. Se várias buscas tiverem sido iniciadas antes pelo método `FindFirst`, apenas a última busca é encerrada aqui. Se nenhuma busca tiver sido iniciada ainda, nada acontece. Para iniciar uma busca, usa-se o método `FindFirst`, e para continuar uma busca, usa-se o método `FindNext`.

Alteração em relação ao DestinyPatch 1.0: agora é possível iniciar mais de uma busca ao mesmo tempo (o erro `DESTINYERROR_SEARCHSTILLOPEN` ficou, portanto, obsoleto). Essas buscas são colocadas em uma "pilha", no princípio LIFO (ou seja, a última busca iniciada é a atual).

## Exemplo

Busca simples em pasta:

```
$
// Listar todos os arquivos PNG da pasta Picture
a[1] = Directory.FindFirst("Picture\*.png");
a[2] = "";

do
  if(a[1] != "")
    // Anexar o arquivo encontrado à lista
    a[2] ..= a[1] .. CRLF;
    // Buscar o próximo arquivo
    a[1] = Directory.FindNext();
  endif;
loop until(a[1] == "");

Directory.FindClose();
```

Busca complexa em pasta (com subpastas):

```
$
// Inicializar as variáveis usadas
a[1] = "";     // Nome do arquivo/diretório do resultado atual
a[2] = "";     // Lista de todos os arquivos/diretórios encontrados
a[3] = "";     // Diretório inicial da busca (com...
// a[4] em diante, as strings seguintes servem para montar o caminho de busca com subpastas
s[1] = False;  // Indica se uma recursão deve ser
s[2] = True;   // iniciada
v[1] = 0;      // Indica se uma busca (recursiva) já

do
  if(!s[2])
    // foi iniciada
    a[1] = Directory.FindNext();
  else
    if(a[3 + Directory.FindLevel] == "")
      // Indica os atributos do resultado
      a[1] = Directory.FindFirst("*");
```

```
    else
      // encontrado
      a[1] = Directory.FindFirst(a[3 + Directory.
    endif;
  endif;

  if(a[1] == "")
    // Continuar a busca no diretório atual
    Directory.FindClose();
  endif;

  // Iniciar a busca na pasta do jogo
  s[1] = !(a[1] == "" || a[1] == "." || a[1] == "

  if(s[1])
    // Iniciar a busca em uma subpasta
    // Nenhum resultado -> fechar a busca (aqui
    v[1] = Directory.GetAttributes(a[2 + Director
  else
    // ...
    v[1] = 0;
  endif;

  // Verificar se o resultado é inválido para a
  s[2] = (v[1] & FILE_ATTRIBUTE_DIRECTORY) != 0;

  // recursão
  if(s[2])
    if(3 + Directory.FindLevel > a.Count)
      // Resultado válido
      a.ReAlloc(3 + Directory.FindLevel);
    endif;
    a[3 + Directory.FindLevel] = a[3 + Directory.
  endif;

  // -> consultar os atributos (para verificar
  if(s[1])
    if(s[2])
      a[2] ..= "Verzeichnis: ";
    else
      a[2] ..= "Datei:       ";
    endif;
    a[2] ..= a[3 + Directory.FindLevel - 1] .. a[
  endif;
loop until(Directory.FindLevel == 0);

// se é um diretório)
Directory.FindCloseAll();
```

## Observações adicionais

Esse método é equivalente à função FindClose do Windows.

## Veja também

`Directory`, `Directory.FindFirst()`, `Directory.FindNext()`, `Directory.FindCloseAll()`, `Directory.FindLevel`, `Directory.GetAttributes()`, `a.Count`, `a.ReAlloc()`

MÉTODO

# Método: Directory.FindCloseAll()

## Sintaxe

```
Directory.FindCloseAll()
```

## Descrição

Com o método FindCloseAll é possível encerrar todas as buscas em pasta atualmente iniciadas. Se nenhuma busca tiver sido iniciada ainda, nada acontece. Para iniciar uma busca, usa-se o método `FindFirst`, para continuar uma busca, usa-se o método `FindNext`, e para encerrar uma única busca, usa-se o método `FindClose`.

Alteração em relação ao DestinyPatch 1.0: agora é possível iniciar mais de uma busca ao mesmo tempo (o erro `DESTINYERROR_SEARCHSTILLOPEN` ficou, portanto, obsoleto). Essas buscas são colocadas em uma "pilha", no princípio LIFO (ou seja, a última busca iniciada é a atual).

## Exemplo

Busca complexa em pasta (com subpastas):

```
$
// Inicializar as variáveis usadas
a[1] = "";     // Nome do arquivo/diretório do resultado atual
a[2] = "";     // Lista de todos os arquivos/diretórios encontrados
a[3] = "";     // Diretório inicial da busca (com...
// a[4] em diante, as strings seguintes servem para montar o caminho de busca com subpastas
s[1] = False;  // Indica se uma recursão deve ser
s[2] = True;   // iniciada
v[1] = 0;      // Indica se uma busca (recursiva) já

do
  if(!s[2])
    // foi iniciada
    a[1] = Directory.FindNext()
  else
    if(a[3 + Directory.FindLevel] == "")
      // Indica os atributos do resultado
      a[1] = Directory.FindFirst("*");
    else
      // encontrado
      a[1] = Directory.FindFirst(a[3 + Directory.
    endif;
  endif;

  if(a[1] == "")
    // Continuar a busca no diretório atual
    Directory.FindClose();
  endif;

  // Iniciar a busca na pasta do jogo
  s[1] = !(a[1] == "" || a[1] == "." || a[1] == "

  if(s[1])
    // Iniciar a busca em uma subpasta
    // Nenhum resultado -> fechar a busca (aqui
    v[1] = Directory.GetAttributes(a[2 + Director
  else
    // ...
```

```
    v[1] = 0;
  endif;

  // Verificar se o resultado é inválido para a
  s[2] = (v[1] & FILE_ATTRIBUTE_DIRECTORY) != 0;

  // recursão
  if(s[2])
    if(3 + Directory.FindLevel > a.Count)
      // Resultado válido
      a.ReAlloc(3 + Directory.FindLevel);
    endif;
    a[3 + Directory.FindLevel] = a[3 + Directory.
  endif;

  // -> consultar os atributos (para verificar
  if(s[1])
    if(s[2])
      a[2] ..= "Verzeichnis: ";
    else
      a[2] ..= "Datei:       ";
    endif;
    a[2] ..= a[3 + Directory.FindLevel - 1] .. a[
  endif;
loop until(Directory.FindLevel == 0);

// se é um diretório)
Directory.FindCloseAll();
```

## Observações adicionais

Esse método é equivalente à função FindClose do Windows (dentro de um laço).

## Veja também

`Directory`, `Directory.FindFirst()`, `Directory.FindNext()`, `Directory.FindClose()`, `Directory.FindLevel`, `Directory.GetAttributes()`, `a.Count`, `a.ReAlloc()`

MÉTODO

# Método: Directory.FindFirst()

## Sintaxe

```
Directory.FindFirst(Máscara de busca)
```

## Descrição

Com o método FindFirst é possível iniciar a busca em uma pasta. Se já houver uma busca iniciada, a busca antiga é pausada até que a busca recém-iniciada termine. A máscara de busca indica quais critérios os arquivos/pastas encontrados precisam atender. Só é permitido buscar na pasta do jogo (e em suas subpastas). Acessos fora da pasta do jogo não são permitidos! Para listar mais arquivos/subpastas da pasta, usa-se o método `FindNext`, e para encerrar uma busca, usa-se o método `FindClose`. Na maioria das buscas, as primeiras pastas exibidas são "." e "..". "." representa a pasta atual e ".." a pasta superior. Se essas informações não forem desejadas, esses resultados precisam ser ignorados.

Alteração em relação ao DestinyPatch 1.0: agora é possível iniciar mais de uma busca ao mesmo tempo (o erro `DESTINYERROR_SEARCHSTILLOPEN` ficou, portanto, obsoleto). Essas buscas são colocadas em uma "pilha", no princípio LIFO (ou seja, a última busca iniciada é a atual).

## Lista de parâmetros

### Parâmetro: `Máscara de busca`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** Indica o caminho (relativo) da pasta a buscar, além dos curingas. Como curingas estão disponíveis o asterisco (*) e a interrogação (?). O asterisco representa qualquer quantidade de caracteres desconhecidos (inclusive zero), e a interrogação representa exatamente um caractere desconhecido. Pelo menos um desses curingas precisa ser informado, para que seja possível fazer uma busca útil. Geralmente usa-se o asterisco para listar todos os arquivos/subpastas da pasta do jogo (veja o exemplo).

### Valor de retorno

- **Tipo de dado:** `string`
- **Descrição:** O nome do arquivo/da pasta (sem caminho), ou uma string vazia, caso nada tenha sido encontrado. `Directory.FindClose()` deveria ser chamado quando esse valor for uma string vazia.

## Exemplo

Busca simples em pasta:

```
$
// Listar todos os arquivos PNG da pasta Picture
a[1] = Directory.FindFirst("Picture\*.png");
a[2] = "";

do
  if(a[1] != "")
    // Anexar o arquivo encontrado à lista
    a[2] ..= a[1] .. CRLF;
    // Buscar o próximo arquivo
    a[1] = Directory.FindNext();
  endif;
loop until(a[1] == "");

Directory.FindClose();
```

Busca complexa em pasta (com subpastas):

```
$
// Inicializar as variáveis usadas
a[1] = "";     // Nome do arquivo/diretório do resultado atual
a[2] = "";     // Lista de todos os arquivos/diretórios encontrados
a[3] = "";     // Diretório inicial da busca (com...
// a[4] em diante, as strings seguintes servem para montar o caminho de busca com subpastas
s[1] = False;  // Indica se uma recursão deve ser
s[2] = True;   // iniciada
v[1] = 0;      // Indica se uma busca (recursiva) já

do
  if(!s[2])
    // foi iniciada
    a[1] = Directory.FindNext();
  else
```

```
    if(a[3 + Directory.FindLevel] == "")
      // Indica os atributos do resultado
      a[1] = Directory.FindFirst("*");
    else
      // encontrado
      a[1] = Directory.FindFirst(a[3 + Directory.
    endif;
  endif;

  if(a[1] == "")
    // Continuar a busca no diretório atual
    Directory.FindClose();
  endif;

  // Iniciar a busca na pasta do jogo
  s[1] = !(a[1] == "" || a[1] == "." || a[1] == "

  if(s[1])
    // Iniciar a busca em uma subpasta
    // Nenhum resultado -> fechar a busca (aqui
    v[1] = Directory.GetAttributes(a[2 + Director
  else
    // ...
    v[1] = 0;
  endif;

  // Verificar se o resultado é inválido para a
  s[2] = (v[1] & FILE_ATTRIBUTE_DIRECTORY) != 0;

  // recursão
  if(s[2])
    if(3 + Directory.FindLevel > a.Count)
      // Resultado válido
      a.ReAlloc(3 + Directory.FindLevel);
    endif;
    a[3 + Directory.FindLevel] = a[3 + Directory.
  endif;

  // -> consultar os atributos (para verificar
  if(s[1])
    if(s[2])
      a[2] ..= "Verzeichnis: ";
    else
      a[2] ..= "Datei:       ";
    endif;
    a[2] ..= a[3 + Directory.FindLevel - 1] .. a[
  endif;
loop until(Directory.FindLevel == 0);

// se é um diretório)
Directory.FindCloseAll();
```

## Observações adicionais

Esse método é equivalente à função FindFirstFile do Windows.

## Erros possíveis

`DESTINYERROR_MEMORY`, `DESTINYERROR_CANTSTARTSEARCH`, `DESTINYERROR_NOPERMISSION`, `DESTINYERROR_CANTRESOLVEPATH`

## Veja também

`Directory`, `Directory.FindNext()`, `Directory.FindClose()`, `Directory.FindCloseAll()`, `Directory.FindLevel`, `Directory.GetAttributes()`, `a.Count`, `a.ReAlloc()`

PROPRIEDADE

# Propriedade: Directory.FindLevel

## Sintaxe

```
Directory.FindLevel
```

## Descrição

Com a propriedade FindLevel é possível descobrir a quantidade de buscas em pasta atualmente iniciadas. Se nenhuma busca tiver sido iniciada ainda, esse valor é igual a 0. Para iniciar uma busca, usa-se o método `FindFirst`, para continuar uma busca, usa-se o método `FindNext`, e para encerrar uma única busca, usa-se o método `FindClose`.

Alteração em relação ao DestinyPatch 1.0: agora é possível iniciar mais de uma busca ao mesmo tempo (o erro `DESTINYERROR_SEARCHSTILLOPEN` ficou, portanto, obsoleto). Essas buscas são colocadas em uma "pilha", no princípio LIFO (ou seja, a última busca iniciada é a atual).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

## Exemplo

Busca complexa em pasta (com subpastas):

```
$
// Inicializar as variáveis usadas
a[1] = "";     // Nome do arquivo/diretório do resultado atual
a[2] = "";     // Lista de todos os arquivos/diretórios encontrados
a[3] = "";     // Diretório inicial da busca (com...
// a[4] em diante, as strings seguintes servem para montar o caminho de busca com subpastas
s[1] = False;  // Indica se uma recursão deve ser
s[2] = True;   // iniciada
v[1] = 0;      // Indica se uma busca (recursiva) já

do
  if(!s[2])
    // foi iniciada
    a[1] = Directory.FindNext()
  else
    if(a[3 + Directory.FindLevel] == "")
      // Indica os atributos do resultado
      a[1] = Directory.FindFirst("*");
    else
      // encontrado
      a[1] = Directory.FindFirst(a[3 + Directory.
    endif;
  endif;
```

```
  if(a[1] == "")
    // Continuar a busca no diretório atual
    Directory.FindClose();
  endif;

  // Iniciar a busca na pasta do jogo
  s[1] = !(a[1] == "" || a[1] == "." || a[1] == "

  if(s[1])
    // Iniciar a busca em uma subpasta
    // Nenhum resultado -> fechar a busca (aqui
    v[1] = Directory.GetAttributes(a[2 + Director
  else
    // ...
    v[1] = 0;
  endif;

  // Verificar se o resultado é inválido para a
  s[2] = (v[1] & FILE_ATTRIBUTE_DIRECTORY) != 0;

  // recursão
  if(s[2])
    if(3 + Directory.FindLevel > a.Count)
      // Resultado válido
      a.ReAlloc(3 + Directory.FindLevel);
    endif;
    a[3 + Directory.FindLevel] = a[3 + Directory.
  endif;

  // -> consultar os atributos (para verificar
  if(s[1])
    if(s[2])
      a[2] ..= "Verzeichnis: ";
    else
      a[2] ..= "Datei:       ";
    endif;
    a[2] ..= a[3 + Directory.FindLevel - 1] .. a[
  endif;
loop until(Directory.FindLevel == 0);

// se é um diretório)
Directory.FindCloseAll();
```

## Veja também

`Directory`, `Directory.FindFirst()`, `Directory.FindNext()`, `Directory.FindClose()`, `Directory.FindCloseAll()`, `Directory.GetAttributes()`, `a.Count`, `a.ReAlloc()`

MÉTODO

# Método: Directory.FindNext()

## Sintaxe

```
Directory.FindNext()
```

## Descrição

Com o método FindNext é possível continuar a última busca em pasta iniciada. Se uma busca for encerrada pelo método `FindClose()`, este método passa a continuar a busca iniciada anteriormente a ela. Se nenhuma busca tiver sido iniciada ainda, um erro é lançado. Para iniciar uma busca, usa-se o método `FindFirst`, e para encerrar uma busca, usa-se o método `FindClose`. Na maioria das buscas, as primeiras pastas exibidas são "." e "..". "." representa a pasta atual e ".." a pasta superior. Se essas informações não forem desejadas, esses resultados precisam ser ignorados.

Alteração em relação ao DestinyPatch 1.0: agora é possível iniciar mais de uma busca ao mesmo tempo (o erro `DESTINYERROR_SEARCHSTILLOPEN` ficou, portanto, obsoleto). Essas buscas são colocadas em uma "pilha", no princípio LIFO (ou seja, a última busca iniciada é a atual).

## Lista de parâmetros

### Valor de retorno

- **Tipo de dado:** `string`
- **Descrição:** O nome do arquivo/da pasta (sem caminho), ou uma string vazia, caso nada tenha sido encontrado. `Directory.FindClose()` deveria ser chamado quando esse valor for uma string vazia.

## Exemplo

Busca simples em pasta:

```
$
// Listar todos os arquivos PNG da pasta Picture
a[1] = Directory.FindFirst("Picture\*.png");
a[2] = "";

do
  if(a[1] != "")
    // Anexar o arquivo encontrado à lista
    a[2] ..= a[1] .. CRLF;
    // Buscar o próximo arquivo
    a[1] = Directory.FindNext();
  endif;
loop until(a[1] == "");

Directory.FindClose();
```

Busca complexa em pasta (com subpastas):

```
$
// Inicializar as variáveis usadas
a[1] = "";     // Nome do arquivo/diretório do resultado atual
a[2] = "";     // Lista de todos os arquivos/diretórios encontrados
a[3] = "";     // Diretório inicial da busca (com...
// a[4] em diante, as strings seguintes servem para montar o caminho de busca com subpastas
s[1] = False;  // Indica se uma recursão deve ser
s[2] = True;   // iniciada
v[1] = 0;      // Indica se uma busca (recursiva) já

do
  if(!s[2])
    // foi iniciada
    a[1] = Directory.FindNext();
  else
    if(a[3 + Directory.FindLevel] == "")
```

```
      // Indica os atributos do resultado
      a[1] = Directory.FindFirst("*");
    else
      // encontrado
      a[1] = Directory.FindFirst(a[3 + Directory.
    endif;
  endif;

  if(a[1] == "")
    // Continuar a busca no diretório atual
    Directory.FindClose();
  endif;

  // Iniciar a busca na pasta do jogo
  s[1] = !(a[1] == "" || a[1] == "." || a[1] == "

  if(s[1])
    // Iniciar a busca em uma subpasta
    // Nenhum resultado -> fechar a busca (aqui
    v[1] = Directory.GetAttributes(a[2 + Director
  else
    // ...
    v[1] = 0;
  endif;

  // Verificar se o resultado é inválido para a
  s[2] = (v[1] & FILE_ATTRIBUTE_DIRECTORY) != 0;

  // recursão
  if(s[2])
    if(3 + Directory.FindLevel > a.Count)
      // Resultado válido
      a.ReAlloc(3 + Directory.FindLevel);
    endif;
    a[3 + Directory.FindLevel] = a[3 + Directory.
  endif;

  // -> consultar os atributos (para verificar
  if(s[1])
    if(s[2])
      a[2] ..= "Verzeichnis: ";
    else
      a[2] ..= "Datei:       ";
    endif;
    a[2] ..= a[3 + Directory.FindLevel - 1] .. a[
  endif;
loop until(Directory.FindLevel == 0);

// se é um diretório)
Directory.FindCloseAll();
```

## Observações adicionais

Esse método é equivalente à função FindNextFile do Windows.

## Erros possíveis

`DESTINYERROR_MEMORY`, `DESTINYERROR_NOSEARCHSTARTED`

## Veja também

`Directory`, `Directory.FindFirst()`, `Directory.FindClose()`, `Directory.FindCloseAll()`, `Directory.FindLevel`, `Directory.GetAttributes()`, `a.Count`, `a.ReAlloc()`

OBJETO

# Objeto: Error

## Sintaxe

```
Error
```

## Descrição

Com o objeto Error é possível (des)ativar mensagens de erro individuais. Além disso, é possível descobrir informações (texto e título) de um erro, ou disparar um erro específico.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Count` | Propriedade | Devolve a quantidade de elementos nesse array. |
| `DisableAll()` | Método | Com o método DisableAll, todas as mensagens de erro são desativadas individualmente. |
| `EnableAll()` | Método | Com o método EnableAll, todas as mensagens de erro são reativadas individualmente. |
| `Error[]` | Objeto | Permite o acesso a uma mensagem de erro específica. |

PROPRIEDADE

# Propriedade: Error.Count

## Sintaxe

```
Error.Count
```

## Descrição

Devolve a quantidade de elementos nesse array. Isso pode ser usado para fazer um laço sobre todas as entradas de `Error[]` (veja o exemplo).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

## Exemplo

```
$
// Montar uma lista de todos os títulos de erro e
```

```
a[1] = "";
for(v[1] = 0; v[1] < Error.Count; v[1]++)
  a[1] ..= Error[v[1]].Title .. CRLF;
next;
```

## Veja também

`Error[]`, `Error[].Title`

MÉTODO

# Método: Error.DisableAll()

## Sintaxe

```
Error.DisableAll()
```

## Descrição

Com o método DisableAll, todas as mensagens de erro são desativadas individualmente. Isso é útil quando se deseja que apenas algumas poucas mensagens de erro fiquem ativadas. Para isso, primeiro desativam-se todas as mensagens de erro com este método e, em seguida, reativam-se individualmente as mensagens de erro desejadas.

> **Informação:** Para (des)ativar a saída de erros de forma global, recomenda-se usar os métodos `Enable` e `Disable` do objeto `Errors`.

## Exemplo

```
$
// Desativar todas as mensagens de erro
Error.DisableAll();
// Verificar se a mensagem de erro "Divisão por zero"
s[1] = Error[DESTINYERROR_DIVISIONBYZERO].Enabled
```

## Veja também

`Error`, `Error[]`, `Error[].Enabled`, `Error.EnableAll()`

MÉTODO

# Método: Error.EnableAll()

## Sintaxe

```
Error.EnableAll()
```

## Descrição

Com o método EnableAll, todas as mensagens de erro são reativadas individualmente. Isso é útil quando algumas mensagens de erro específicas foram desativadas antes e devem ser rapidamente reativadas todas de uma vez.

> **Informação:** Para (des)ativar a saída de erros de forma global, recomenda-se usar os métodos `Enable` e `Disable` do objeto `Errors`.

## Exemplo

```
$
// Desativar a mensagem de erro "Divisão por zero"
Error[DESTINYERROR_DIVISIONBYZERO].Enabled = Fals
// Reativar todas as mensagens de erro
Error.EnableAll();
// Verificar se a mensagem de erro "Divisão por zero"
s[1] = Error[DESTINYERROR_DIVISIONBYZERO].Enabled
```

## Veja também

`Error`, `Error[]`, `Error[].Enabled`, `Error.DisableAll()`

OBJETO

# Objeto: Error[]

## Sintaxe

```
Error[Fehler-Id]
```

## Descrição

Permite o acesso a uma mensagem de erro específica.

## Lista de parâmetros

### Parâmetro: `Fehler-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** -1 ... Error.Count
- **Descrição:** Indica o número do respectivo erro. Precisa estar na faixa de -1 a `Error.Count`. Para isso podem ser usadas as `constantes de erro do Destiny`. O número de erro 0 é idêntico aqui ao número de erro -1, para garantir compatibilidade com a versão antiga do DestinyPatch.

## Exemplo

```
$
// Montar uma lista de todos os títulos de erro e
a[1] = "";
for(v[1] = 0; v[1] < Error.Count; v[1]++)
  a[1] ..= Error[v[1]].Title .. CRLF;
next;
```

## Erros possíveis

`DESTINYERROR_ARRAYBOUND`

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Enabled` | Propriedade | Com a propriedade Enabled, uma mensagem de erro específica pode ser (des)ativada explicitamente. |
| `Message` | Propriedade | Contém o texto da mensagem de um determinado erro. |
| `Raise()` | Método | Com o método Raise é possível disparar explicitamente um erro específico. |
| `Title` | Propriedade | Contém o título da mensagem de um determinado erro. |

## Veja também

`Error[].Title`, `Error.Count`

PROPRIEDADE

# Propriedade: Error[].Enabled

## Sintaxe

```
Error[Fehler-Id].Enabled
```

## Descrição

Com a propriedade Enabled, uma mensagem de erro específica pode ser (des)ativada explicitamente.

> **Informação:** Se essa mensagem de erro estiver ativada, mas todas as mensagens de erro estiverem desativadas globalmente por `Errors.Disable()`, então a mensagem de erro não é exibida, caso ocorra um erro.

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** não

## Exemplo

```
$
// A partir daqui, em caso de erro, continuar na próxima linha de comando
Errors.Resume();

// Desativar o erro de divisão por zero
Error[DESTINYERROR_DIVISIONBYZERO].Enabled = Fals

// Dividir um número qualquer por outro número qualquer
v[1] = Math.Rand(0, 100);
v[2] = Math.Rand(0, 100);
v[3] = v[1] / v[2];
```

```
// Ativar o erro de divisão por zero
Error[DESTINYERROR_DIVISIONBYZERO].Enabled = True

// A partir daqui, em caso de erro, voltar a interromper os scripts
Errors.Halt();

// Descobrir o número do último erro e guardá-lo em
v[1] = Errors.Catch();
// Se for o caso, disparar o último erro (isso encerra
if(v[1] != 0)
  Error[v[1]].Raise();
endif;
```

## Veja também

`Error[].Raise()`, `Errors`, `Errors.Enable()`, `Errors.Disable()`, `Errors.Halt()`, `Errors.Resume()`, `Errors.Catch()`, `Math.Rand()`

PROPRIEDADE

# Propriedade: Error[].Message

## Sintaxe

```
Error[Fehler-Id].Message
```

## Descrição

Contém o texto da mensagem de um determinado erro. Esse texto depende do idioma configurado no DestinyPatch, já que normalmente é usado para mostrar uma mensagem de erro ao usuário.

## Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** sim
- **Depende do idioma:** sim

## Exemplo

```
$
// A partir daqui, desativar globalmente as mensagens de erro
Errors.Disable();
// A partir daqui, em caso de erro, continuar na próxima linha de comando
Errors.Resume();

// Disparar um erro (aqui: divisão por zero)
v[1] /= 0;

// Esta linha ainda é executada; por isso, o número
v[2] = Errors.Catch();        // 15 (= número do erro
// Se ocorreu um erro (aqui, sim
if(v[2] != 0)
  a[1] = Error[v[2]].Message; // "Erro 15: Divi
  a[2] = Error[v[2]].Title;   // "Tentou-se
else
```

```
  a[1] = "";
  a[2] = "";
endif;

// A partir daqui, em caso de erro, voltar a interromper os scripts
Errors.Halt();
// A partir daqui, voltar a exibir globalmente as mensagens de erro
Errors.Enable();
```

## Veja também

`Error[].Title`, `Errors`, `Errors.Enable()`, `Errors.Disable()`, `Errors.Halt()`, `Errors.Resume()`, `Errors.Catch()`

MÉTODO

# Método: Error[].Raise()

## Sintaxe

```
Error[Fehler-Id].Raise()
```

## Descrição

Com o método Raise é possível disparar explicitamente um erro específico. Isso se comporta exatamente como se o erro tivesse realmente ocorrido. Ou seja, `Errors.Catch()` devolve esse erro como o último erro, a mensagem de erro é exibida (se não estiver desativada) e o DestinyScript é interrompido (se não estiver configurado de outra forma).

## Exemplo

```
$
// A partir daqui, em caso de erro, continuar na próxima linha de comando
Errors.Resume();

// Desativar o erro de divisão por zero
Error[DESTINYERROR_DIVISIONBYZERO].Enabled = Fals

// Dividir um número qualquer por outro número qualquer
v[1] = Math.Rand(0, 100);
v[2] = Math.Rand(0, 100);
v[3] = v[1] / v[2];

// Ativar o erro de divisão por zero
Error[DESTINYERROR_DIVISIONBYZERO].Enabled = True

// A partir daqui, em caso de erro, voltar a interromper os scripts
Errors.Halt();

// Descobrir o número do último erro e guardá-lo em
v[1] = Errors.Catch();
// Se for o caso, disparar o último erro (isso encerra
if(v[1] != 0)
  Error[v[1]].Raise();
endif;
```

## Veja também

`Error[].Enabled`, `Errors`, `Errors.Enable()`, `Errors.Disable()`, `Errors.Halt()`, `Errors.Resume()`, `Errors.Catch()`, `Math.Rand()`

PROPRIEDADE

# Propriedade: Error[].Title

## Sintaxe

```
Error[Fehler-Id].Title
```

## Descrição

Contém o título da mensagem de um determinado erro. Esse título depende do idioma configurado no DestinyPatch, já que normalmente é usado para mostrar uma mensagem de erro ao usuário.

## Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** sim
- **Depende do idioma:** sim

## Exemplo

```
$
// A partir daqui, desativar globalmente as mensagens de erro
Errors.Disable();
// A partir daqui, em caso de erro, continuar na próxima linha de comando
Errors.Resume();

// Disparar um erro (aqui: divisão por zero)
v[1] /= 0;

// Esta linha ainda é executada; por isso, o número
v[2] = Errors.Catch();        // 15 (= número do erro
// Se ocorreu um erro (aqui, sim
if(v[2] != 0)
  a[1] = Error[v[2]].Message; // "Erro 15: Divi
  a[2] = Error[v[2]].Title;   // "Tentou-se
else
  a[1] = "";
  a[2] = "";
endif;

// A partir daqui, em caso de erro, voltar a interromper os scripts
Errors.Halt();
// A partir daqui, voltar a exibir globalmente as mensagens de erro
Errors.Enable();
```

## Veja também

`Error[].Message`, `Errors`, `Errors.Enable()`, `Errors.Disable()`, `Errors.Halt()`, `Errors.Resume()`, `Errors.Catch()`

OBJETO

# Objeto: Errors

## Sintaxe

```
Errors
```

## Descrição

Com o objeto Errors é possível controlar o tratamento de erros. Assim, é possível, por exemplo, suprimir a exibição de mensagens de erro ou fazer os DestinyScripts continuarem na próxima linha de comando em caso de erro.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Catch()` | Método | Devolve o número do último erro (se tiver ocorrido). |
| `Disable()` | Método | Desativa a exibição de mensagens de erro, o que faz com que, em caso de erro, nenhuma mensagem apareça, de modo que isso talvez não seja percebido durante o desenvolvimento do jogo. |
| `Enable()` | Método | Ativa a exibição de mensagens de erro, o que faz com que, em caso de erro, o usuário veja, se for o caso, uma mensagem de erro. |
| `Halt()` | Método | Faz os DestinyScripts serem interrompidos em caso de erro. |
| `Resume()` | Método | Faz os DestinyScripts continuarem na próxima linha de comando em caso de erro. |

MÉTODO

# Método: Errors.Catch()

## Sintaxe

```
Errors.Catch()
```

## Descrição

Devolve o número do último erro (se tiver ocorrido). Uma vez lido o número de erro, o valor é reiniciado para 0.

> **Informação:** No início de cada DestinyScript, o número do último erro é reiniciado para 0.

## Lista de parâmetros

### Valor de retorno

- **Tipo de dado:** `dword`
- **Descrição:** O número do último erro ocorrido, ou 0, caso nenhum erro tenha ocorrido.

## Exemplo

```
$
// A partir daqui, desativar globalmente as mensagens de erro
Errors.Disable();
// A partir daqui, em caso de erro, continuar na próxima linha de comando
Errors.Resume();

// Disparar um erro (aqui: divisão por zero)
v[1] /= 0;

// Esta linha ainda é executada; por isso, o número
v[2] = Errors.Catch();        // 15 (= número do erro
// Se ocorreu um erro (aqui, sim
if(v[2] != 0)
  a[1] = Error[v[2]].Message; // "Erro 15: Divi
  a[2] = Error[v[2]].Title;   // "Tentou-se
else
  a[1] = "";
  a[2] = "";
endif;

// A partir daqui, em caso de erro, voltar a interromper os scripts
Errors.Halt();
// A partir daqui, voltar a exibir globalmente as mensagens de erro
Errors.Enable();
```

## Observações adicionais

Para avaliar o valor de retorno dessa função, recomenda-se usar as `constantes de erro do Destiny`.

## Veja também

`Errors`, `Errors.Enable()`, `Errors.Disable()`, `Errors.Resume()`, `Errors.Halt()`, `Error[].Message`, `Error[].Title`

MÉTODO

# Método: Errors.Disable()

## Sintaxe

```
Errors.Disable()
```

## Descrição

Desativa a exibição de mensagens de erro, o que faz com que, em caso de erro, nenhuma mensagem apareça, de modo que isso talvez não seja percebido durante o desenvolvimento do jogo.

> **Informação:** Se uma determinada mensagem de erro tiver sido ativada explicitamente pelo objeto `Error[]` e este método for chamado aqui, então o respectivo erro não é exibido, caso ocorra.

> **Informação:** A configuração padrão é que as mensagens de erro são exibidas.

## Exemplo

```
$
// A partir daqui, desativar globalmente as mensagens de erro
Errors.Disable();
// A partir daqui, em caso de erro, continuar na próxima linha de comando
Errors.Resume();

// Disparar um erro (aqui: divisão por zero)
v[1] /= 0;

// Esta linha ainda é executada; por isso, o número
v[2] = Errors.Catch();        // 15 (= número do erro
// Se ocorreu um erro (aqui, sim
if(v[2] != 0)
  a[1] = Error[v[2]].Message; // "Erro 15: Divi
  a[2] = Error[v[2]].Title;   // "Tentou-se
else
  a[1] = "";
  a[2] = "";
endif;

// A partir daqui, em caso de erro, voltar a interromper os scripts
Errors.Halt();
// A partir daqui, voltar a exibir globalmente as mensagens de erro
Errors.Enable();
```

## Veja também

`Errors`, `Errors.Enable()`, `Errors.Resume()`, `Errors.Halt()`, `Errors.Catch()`, `Error[].Message`, `Error[].Title`

MÉTODO

# Método: Errors.Enable()

## Sintaxe

```
Errors.Enable()
```

## Descrição

Ativa a exibição de mensagens de erro, o que faz com que, em caso de erro, o usuário veja, se for o caso, uma mensagem de erro.

> **Informação:** Se uma determinada mensagem de erro tiver sido desativada explicitamente pelo objeto `Error[]` e este método for chamado aqui, então o respectivo erro não é exibido, caso ocorra.

> **Informação:** A configuração padrão é que as mensagens de erro são exibidas.

## Exemplo

```
$
// A partir daqui, desativar globalmente as mensagens de erro
```

```
Errors.Disable();
// A partir daqui, em caso de erro, continuar na próxima linha de comando
Errors.Resume();

// Disparar um erro (aqui: divisão por zero)
v[1] /= 0;

// Esta linha ainda é executada; por isso, o número
v[2] = Errors.Catch();        // 15 (= número do erro
// Se ocorreu um erro (aqui, sim
if(v[2] != 0)
  a[1] = Error[v[2]].Message; // "Erro 15: Divi
  a[2] = Error[v[2]].Title;   // "Tentou-se
else
  a[1] = "";
  a[2] = "";
endif;

// A partir daqui, em caso de erro, voltar a interromper os scripts
Errors.Halt();
// A partir daqui, voltar a exibir globalmente as mensagens de erro
Errors.Enable();
```

## Veja também

`Errors`, `Errors.Disable()`, `Errors.Resume()`, `Errors.Halt()`, `Errors.Catch()`, `Error[].Message`, `Error[].Title`

MÉTODO

# Método: Errors.Halt()

## Sintaxe

```
Errors.Halt()
```

## Descrição

Faz os DestinyScripts serem interrompidos em caso de erro.

> **Informação:** Se os DestinyScripts forem interrompidos em caso de erro e as mensagens de erro estiverem suprimidas, talvez o desenvolvedor do jogo não perceba que ocorreu um erro. Para também exibir mensagens de erro (caso tenham sido desativadas antes), é preciso chamar o método `Enable` do objeto `Errors`.

> **Informação:** A configuração padrão é que os DestinyScripts são interrompidos em caso de erro.

## Exemplo

```
$
// A partir daqui, desativar globalmente as mensagens de erro
Errors.Disable();
// A partir daqui, em caso de erro, continuar na próxima linha de comando
Errors.Resume();

// Disparar um erro (aqui: divisão por zero)
```

```
v[1] /= 0;

// Esta linha ainda é executada; por isso, o número
v[2] = Errors.Catch();        // 15 (= número do erro
// Se ocorreu um erro (aqui, sim
if(v[2] != 0)
  a[1] = Error[v[2]].Message; // "Erro 15: Divi
  a[2] = Error[v[2]].Title;   // "Tentou-se
else
  a[1] = "";
  a[2] = "";
endif;

// A partir daqui, em caso de erro, voltar a interromper os scripts
Errors.Halt();
// A partir daqui, voltar a exibir globalmente as mensagens de erro
Errors.Enable();
```

## Veja também

`Errors`, `Errors.Enable()`, `Errors.Disable()`, `Errors.Halt()`, `Errors.Catch()`, `Error[].Message`, `Error[].Title`

MÉTODO

# Método: Errors.Resume()

## Sintaxe

```
Errors.Resume()
```

## Descrição

Faz os DestinyScripts continuarem na próxima linha de comando em caso de erro.

> **Informação:** Se os DestinyScripts continuarem rodando em caso de erro e as mensagens de erro não estiverem suprimidas, ainda assim pode acontecer de o usuário ver uma mensagem de erro. Para também suprimir mensagens de erro, é preciso chamar o método `Disable` do objeto `Errors`.

> **Informação:** A configuração padrão é que os DestinyScripts são interrompidos em caso de erro.

## Exemplo

```
$
// A partir daqui, desativar globalmente as mensagens de erro
Errors.Disable();
// A partir daqui, em caso de erro, continuar na próxima linha de comando
Errors.Resume();

// Disparar um erro (aqui: divisão por zero)
v[1] /= 0;

// Esta linha ainda é executada; por isso, o número
v[2] = Errors.Catch();        // 15 (= número do erro
// Se ocorreu um erro (aqui, sim
```

```
if(v[2] != 0)
  a[1] = Error[v[2]].Message; // "Erro 15: Divi
  a[2] = Error[v[2]].Title;   // "Tentou-se
else
  a[1] = "";
  a[2] = "";
endif;

// A partir daqui, em caso de erro, voltar a interromper os scripts
Errors.Halt();
// A partir daqui, voltar a exibir globalmente as mensagens de erro
Errors.Enable();
```

## Veja também

`Errors`, `Errors.Enable()`, `Errors.Disable()`, `Errors.Halt()`, `Errors.Catch()`, `Error[].Message`, `Error[].Title`

OBJETO

# Objeto: Event

## Sintaxe

```
Event
```

## Descrição

O objeto Event serve para tornar acessíveis os eventos do mapa. Diferente do objeto `MapEvent`, aqui a Event-Id é usada como índice para acessar um evento específico do mapa (portanto, o índice é baseado em 1, e não contínuo).

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Count` | Propriedade | A propriedade Count permite descobrir a quantidade de eventos no mapa. |
| `Event[]` | Objeto | O objeto Event[] permite o acesso a um determinado evento do mapa. |

## Veja também

`MapEvent`

PROPRIEDADE

# Propriedade: Event.Count

## Sintaxe

```
Event.Count
```

## Descrição

A propriedade Count permite descobrir a quantidade de eventos no mapa.

> **Informação:** Embora o valor dessa propriedade indique a quantidade de eventos, ela não é adequada para fazer um laço sobre os eventos do mapa, já que o índice do objeto `Event[]` é a Event-Id, que não é atribuída de forma contínua pelo RPG Maker. Para fazer um laço sobre todos os eventos, deve-se usar o objeto `MapEvent`, que tem um índice contínuo baseado em zero.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

## Exemplo

```
$
// Descobrir a quantidade de eventos e guardar em v
v[1] = Event.Count;
```

## Observações adicionais

Essa propriedade é totalmente idêntica a `MapEvent`.Count e `Map.EventCount`.

## Veja também

`Event`, `Event[]`

OBJETO

# Objeto: Event[]

## Sintaxe

```
Event[Evento-Id]
```

## Descrição

O objeto Event[] permite o acesso a um determinado evento do mapa. Aqui é usada a Event-Id como índice.

> **Informação:** Como a Event-Id no RPG Maker é variável, pode acontecer de não ser possível uma numeração sequencial aqui. Nesse caso, deve-se usar o objeto `MapEvent`.

## Lista de parâmetros

### Parâmetro: `Evento-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não

- **Descrição:** A Event-Id do respectivo evento. Aqui também podem ser usadas as constantes de `eventos especiais`:

`HERO`: o evento do herói `BOAT`: o evento do barco `SHIP`: o evento do navio `AIRSHIP`: o evento da aeronave `THIS`: o evento atual

## Erros possíveis

`DESTINYERROR_ARRAYBOUND`

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `AuroraSheet` | Objeto | Com o objeto AuroraSheet é possível acessar o gráfico do evento. |
| `CanMoveTo()` | Método | Com o método CanMoveTo é possível verificar se o evento pode se mover na direção indicada. |
| `Charframe` | Propriedade | Com a propriedade Charframe é possível ler/gravar o índice do Frame usado. |
| `Charset` | Propriedade | Com a propriedade Charset é possível descobrir o nome do arquivo (sem caminho/extensão e relativo à pasta de Charsets) do Charset atualmente usado pelo evento. |
| `DirFlags` | Propriedade | Com a propriedade DirFlags é possível ler/gravar tanto a direção em que o evento olha quanto sua direção adicional de movimento. |
| `DirLook` | Propriedade | Com a propriedade DirLook é possível ler/gravar a direção em que o evento olha. |
| `DirMove` | Propriedade | Com a propriedade DirMove é possível ler/gravar a direção de movimento do evento. |
| `Exists` | Propriedade | Com a propriedade Exists é possível ler/gravar o status de existência do evento. |
| `FixDir` | Propriedade | Com a propriedade FixDir é possível ler/gravar a propriedade "direção fixa" do evento. |
| `Frame` | Propriedade | Com a propriedade Frame é possível ler/gravar o frame de movimento atual do evento. |
| `FrameSimple` | Propriedade | Com a propriedade FrameSimple é possível descobrir o frame de movimento atual do evento em forma simplificada. |
| `Frequency` | Propriedade | Com a propriedade Frequency é possível ler/gravar a frequência de movimento do evento. |
| `Id` | Propriedade | Com a propriedade Id é possível descobrir a Event-Id do evento. |
| `JumpTime` | Propriedade | Com a propriedade JumpTime é possível ler/gravar o tempo interno de ação do evento. |
| `MapId` | Propriedade | Com a propriedade MapId é possível descobrir a Map-Id do evento. |
| `MoveRoute` | Objeto | Com o objeto MoveRoute é possível acessar o trajeto individual de movimento de um evento. |
| `MoveRoute[]` | Objeto | Com o objeto MoveRoute[] é possível acessar um único movimento da MoveRoute. |
| `Name` | Propriedade | Com a propriedade Name é possível descobrir o nome do evento. |
| `Offset` | Propriedade | Com a propriedade Offset é possível ler/gravar a distância atual do evento até o campo seguinte no mapa. |
| `Phasing` | Propriedade | Com a propriedade Phasing é possível ler/gravar a propriedade "atravessar paredes" do evento. |
| `ScreenX` | Propriedade | Com a propriedade ScreenX é possível descobrir a coordenada X do evento na tela. |
| `ScreenY` | Propriedade | Com a propriedade ScreenY é possível descobrir a coordenada Y do evento na tela. |
| `Speed` | Propriedade | Com a propriedade Speed é possível ler/gravar a velocidade do evento. |
| `StopAnimation` | Propriedade | Com a propriedade StopAnimation é possível ler/gravar a propriedade "sem animação de movimento" do evento. |
| `Transparency` | Propriedade | Com a propriedade Transparency é possível ler/gravar o valor de transparência do evento. |

| Nome | Tipo | Resumo |
|---|---|---|
| `X` | Propriedade | Com a propriedade X é possível ler/gravar a coordenada X do evento no mapa. |
| `Y` | Propriedade | Com a propriedade Y é possível ler/gravar a coordenada Y do evento no mapa. |

OBJETO

# Objeto: Event[].AuroraSheet

## Sintaxe

```
Event[Evento-Id].AuroraSheet
```

## Descrição

Com o objeto AuroraSheet é possível acessar o gráfico do evento. Esse gráfico pode ser compartilhado por vários eventos (por exemplo, quando vários eventos usam o mesmo Charset). Se um evento usa o chipset como gráfico, este objeto aponta para o gráfico do chipset. Como o objeto AuroraSheet, sendo padronizado, está disponível como subobjeto em vários objetos, sua descrição foi escrita apenas uma vez e colocada como subordinada ao objeto `Picture[]`. A descrição se encontra aqui: `Picture[].AuroraSheet`.

## Veja também

`Event`, `Event[]`, `Picture[].AuroraSheet`

MÉTODO

# Método: Event[].CanMoveTo()

## Sintaxe

```
Event[Evento-Id].CanMoveTo(Direção)
```

## Descrição

Com o método CanMoveTo é possível verificar se o evento pode se mover na direção indicada. Entre outras coisas, verifica-se aqui:

Se o evento pode, de modo geral, se mover para o campo. Se há outros eventos no caminho.

> **Informação:** Verificações especiais (como, por exemplo, se o barco pode se mover sobre água profunda) também são consideradas aqui.

## Lista de parâmetros

### Parâmetro: `Direção`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 7

- **Descrição:** A direção para a qual se deve verificar se o evento pode se mover. Para isso podem ser usadas as `constantes de direção`:

Constante | Descrição

`DIR_UP` Verificar se o evento pode se mover para cima

`DIR_RIGHT` Verificar se o evento pode se mover para a direita

`DIR_DOWN` Verificar se o evento pode se mover para baixo

`DIR_LEFT` Verificar se o evento pode se mover para a esquerda

`DIR_RIGHT_UP` Verificar se o evento pode se mover para a direita-cima

`DIR_RIGHT_DOWN` Verificar se o evento pode se mover para a direita-baixo

`DIR_LEFT_DOWN` Verificar se o evento pode se mover para a esquerda-baixo

`DIR_LEFT_UP` Verificar se o evento pode se mover para a esquerda-cima

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** Indica se o evento pode se mover na direção indicada (true) ou não (false).

## Exemplo

```
$
// Verificar se o herói pode se mover em linha reta
if(Event[HERO].CanMoveTo(Event[HERO].DirLook))
  // ...
endif;
```

## Observações adicionais

Para a verificação é usada a mesma rotina do RPG_RT. Por isso, o resultado é sempre idêntico ao do RPG_RT.

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Event`, `Event[]`, `Event[].DirMove`, `Event[].DirLook`

PROPRIEDADE

# Propriedade: Event[].Charframe

## Sintaxe

```
Event[Evento-Id].Charframe
```

## Descrição

Com a propriedade Charframe é possível ler/gravar o índice do `Frame` usado. Junto com a propriedade `Charset`, isso indica o char atualmente usado. Esse valor deveria ficar na faixa de 0 ... 7 (veja a figura).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não
- **Faixa de valores:** 0 ... 7
- **Volátil:** sim

## Veja também

`Event`, `Event[]`, `Event[].Charframe`

PROPRIEDADE

# Propriedade: Event[].Charset

## Sintaxe

```
Event[Evento-Id].Charset
```

## Descrição

Com a propriedade Charset é possível descobrir o nome do arquivo (sem caminho/extensão e relativo à pasta de Charsets) do Charset atualmente usado pelo evento. Junto com a propriedade `Charframe`, isso indica o char atualmente usado.

## Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** sim

## Veja também

`Event`, `Event[]`, `Event[].Charframe`

PROPRIEDADE

# Propriedade: Event[].DirFlags

## Sintaxe

```
Event[Evento-Id].DirFlags
```

## Descrição

Com a propriedade DirFlags é possível ler/gravar tanto a direção em que o evento olha quanto sua direção adicional de movimento. Isso é prático para alterar as duas propriedades de uma só vez. Para a indicação, podem ser usadas as `constantes de direção`.

`DIR_UP`: o evento olha para cima `DIR_RIGHT`: o evento olha para a direita `DIR_DOWN`: o evento olha para baixo `DIR_LEFT`: o evento olha para a esquerda

> **Informação:** Como alterar a direção adicional de movimento exige um deslocamento de bits de 8 posições para a esquerda, recomenda-se alterar os valores individuais separadamente, através de `Event[].DirMove` e `Event[].DirLook`.

## Propriedades

- **Tipo de dado:** `word`
- **Somente leitura:** não

## Exemplo

```
$
// Ler a direção em que olha e a direção adicional
v[1] = Event[1].DirFlags;
// Definir a direção em que olha e a direção adicion
Event[1].DirFlags = DIR_LEFT | (DIR_LEFT << 8);
```

## Veja também

`Event`, `Event[]`, `Event[].DirMove`, `Event[].DirLook`

PROPRIEDADE

# Propriedade: Event[].DirLook

## Sintaxe

```
Event[Evento-Id].DirLook
```

## Descrição

Com a propriedade DirLook é possível ler/gravar a direção em que o evento olha. Para isso podem ser usadas as `constantes de direção`:

Constante | Descrição

`DIR_UP` O evento se move para cima

`DIR_RIGHT` O evento se move para a direita

`DIR_DOWN` O evento se move para baixo

`DIR_LEFT` O evento se move para a esquerda

> **Informação:** A diferença entre `Event[].DirMove` e `Event[].DirLook` é que a propriedade DirLook representa a direção para a qual o evento efetivamente olha, enquanto a propriedade DirMove indica a direção do movimento (por exemplo, quando o herói anda na diagonal). Tanto ao andar na diagonal quanto ao fixar a direção do olhar, a direção em que se olha pode ficar diferente da direção do movimento.

## Sinônimos

Dir2

## Propriedades

- **Tipo de dado:** `byte`
- **Somente leitura:** não

## Exemplo

```
$
// Descobrir a direção em que olha o primeiro even
v[1] = Event[1].DirLook;
// Girar em sentido horário a direção em que olha o
switch(Event[1].DirLook)
  case DIR_UP:
    Event[1].DirLook = DIR_RIGHT;
    break;
  case DIR_RIGHT:
    Event[1].DirLook = DIR_DOWN;
    break;
  case DIR_DOWN:
    Event[1].DirLook = DIR_LEFT;
    break;
  case DIR_LEFT:
    Event[1].DirLook = DIR_UP;
    break;
endswitch;
```

## Veja também

`Event`, `Event[]`, `Event[].DirMove`, `Event[].DirFlags`, `Event[].CanMoveTo()`

PROPRIEDADE

# Propriedade: Event[].DirMove

## Sintaxe

```
Event[Evento-Id].DirMove
```

## Descrição

Com a propriedade DirMove é possível ler/gravar a direção de movimento do evento. Para isso podem ser usadas as `constantes de direção`:

Constante | Descrição

`DIR_UP` O evento se move para cima

`DIR_RIGHT` O evento se move para a direita

`DIR_DOWN` O evento se move para baixo

`DIR_LEFT` O evento se move para a esquerda

`DIR_RIGHT_UP` O evento se move para a direita-cima

`DIR_RIGHT_DOWN` O evento se move para a direita-baixo

`DIR_LEFT_DOWN` O evento se move para a esquerda-baixo

`DIR_LEFT_UP` O evento se move para a esquerda-cima

> **Informação:** A diferença entre `Event[].DirMove` e `Event[].DirLook` é que a propriedade DirLook representa a direção para a qual o evento efetivamente olha, enquanto a propriedade DirMove indica a direção do movimento (por exemplo, quando o herói anda na diagonal). Tanto ao andar na diagonal quanto ao fixar a direção do olhar, a direção em que se olha pode ficar diferente da direção do movimento.

## Sinônimos

Dir1

## Propriedades

- **Tipo de dado:** `byte`
- **Somente leitura:** não

## Exemplo

```
$
// Descobrir a direção de movimento do primeiro eve
v[1] = Event[1].DirMove;
// Girar a direção de movimento do primeiro evento n
switch(Event[1].DirMove)
  case DIR_UP:
    Event[1].DirMove = DIR_DOWN;
    break;
  case DIR_RIGHT:
    Event[1].DirMove = DIR_LEFT;
    break;
  case DIR_DOWN:
    Event[1].DirMove = DIR_UP;
    break;
  case DIR_LEFT:
    Event[1].DirMove = DIR_RIGHT;
    break;
  case DIR_RIGHT_UP:
    Event[1].DirMove = DIR_LEFT_DOWN;
    break;
  case DIR_RIGHT_DOWN:
    Event[1].DirMove = DIR_LEFT_UP;
    break;
  case DIR_LEFT_DOWN:
    Event[1].DirMove = DIR_RIGHT_UP;
    break;
  case DIR_LEFT_UP:
    Event[1].DirMove = DIR_RIGHT_DOWN;
```

```
    break;
endswitch;
```

## Veja também

`Event`, `Event[]`, `Event[].DirLook`, `Event[].DirFlags`, `Event[].CanMoveTo()`

PROPRIEDADE

# Propriedade: Event[].Exists

## Sintaxe

```
Event[Evento-Id].Exists
```

## Descrição

Com a propriedade Exists é possível ler/gravar o status de existência do evento. Se esse valor for true, o evento é exibido normalmente. Se esse valor for false, o evento é considerado apagado.

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** não
- **Volátil:** sim

## Exemplo

```
$
// Verificar se o primeiro evento existe
if(Event[1].Exists)
  // Marcar o primeiro evento como apagado
  Event[1].Exists = false;
endif;
```

## Veja também

`Event`, `Event[]`

PROPRIEDADE

# Propriedade: Event[].FixDir

## Sintaxe

```
Event[Evento-Id].FixDir
```

## Descrição

Com a propriedade FixDir é possível ler/gravar a propriedade "direção fixa" do evento. False: o evento altera sua direção normalmente (por exemplo, ao andar para a esquerda, o evento olha para a esquerda). True: o evento sempre mantém a direção atual em que olha; ela não é alterada.

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** não
- **Volátil:** sim

## Veja também

`Event`, `Event[]`

PROPRIEDADE

# Propriedade: Event[].Frame

## Sintaxe

```
Event[Evento-Id].Frame
```

## Descrição

Com a propriedade Frame é possível ler/gravar o frame de movimento atual do evento. Para isso podem ser usadas as constantes `EventFrame`.

> **Informação:** A diferença em relação à propriedade `FrameSimple` é que, na propriedade FrameSimple, o valor `EVENTFRAME_NORMAL2` é definido como `EVENTFRAME_NORMAL`, já que esse valor não é relevante para o cálculo gráfico. Em compensação, a propriedade FrameSimple não pode ser gravada.

`EVENTFRAME_LEFT`: o evento está com a perna do lado esquerdo para baixo (frame da esquerda). `EVENTFRAME_NORMAL`: o evento está com as duas pernas no chão (frame do meio). `EVENTFRAME_RIGHT`: o evento está com a perna do lado direito para baixo (frame da direita). `EVENTFRAME_NORMAL2`: o evento está com as duas pernas no chão (frame do meio).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não
- **Faixa de valores:** 0 ... 3
- **Volátil:** sim

## Exemplo

Este exemplo não usa FrameSimple; por isso, a correção precisa ser feita pelo script:

```
$
// As coordenadas do frame de evento atual do
```

```
// (largura de um Charframe = 24, altura de um Char
// (Charframes por char e linha = 3, Charframes
d[1] = Event[1].Frame;
if(d[1] == EVENTFRAME_NORMAL2)
  d[1] = EVENTFRAME_NORMAL;
endif;
v[1] = (Event[1].Charframe % 4) * (3 * 24) + d[1]
v[2] = (Event[1].Charframe / 4) * (4 * 32) + (Eve
```

## Veja também

`Event`, `Event[]`, `Event[].FrameSimple`, `Event[].Charframe`, `Event[].DirLook`

PROPRIEDADE

# Propriedade: Event[].FrameSimple

## Sintaxe

```
Event[Evento-Id].FrameSimple
```

## Descrição

Com a propriedade FrameSimple é possível descobrir o frame de movimento atual do evento em forma simplificada. Para isso podem ser usadas as constantes `EventFrame`.

> **Informação:** A diferença em relação à propriedade `Frame` é que o valor `EVENTFRAME_NORMAL2` é definido como `EVENTFRAME_NORMAL`, já que esse valor não é relevante para o cálculo gráfico. Em compensação, a propriedade FrameSimple não pode ser gravada.

`EVENTFRAME_LEFT`: o evento está com a perna do lado esquerdo para baixo (frame da esquerda). `EVENTFRAME_NORMAL`: o evento está com as duas pernas no chão (frame do meio). `EVENTFRAME_RIGHT`: o evento está com a perna do lado direito para baixo (frame da direita).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** 0 ... 2
- **Volátil:** sim

## Exemplo

Este exemplo usa FrameSimple; por isso, não é preciso fazer correção pelo script:

```
$
// As coordenadas do frame de evento atual do
// (largura de um Charframe = 24, altura de um Char
// (Charframes por char e linha = 3, Charframes
v[1] = (Event[1].Charframe % 4) * (3 * 24) + Even
v[2] = (Event[1].Charframe / 4) * (4 * 32) + (Eve
```

## Veja também

`Event`, `Event[]`, `Event[].Frame`, `Event[].Charframe`, `Event[].DirLook`

PROPRIEDADE

# Propriedade: Event[].Frequency

## Sintaxe

```
Event[Evento-Id].Frequency
```

## Descrição

Com a propriedade Frequency é possível ler/gravar a frequência de movimento do evento.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não
- **Faixa de valores:** 1 ... 8
- **Volátil:** sim

## Veja também

`Event`, `Event[]`

PROPRIEDADE

# Propriedade: Event[].Id

## Sintaxe

```
Event[Evento-Id].Id
```

## Descrição

Com a propriedade Id é possível descobrir a Event-Id do evento. Se o respectivo evento não tiver Event-Id (por exemplo, um dos eventos especiais), é devolvido 0.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

## Exemplo

```
$
// Descobrir a Event-Id do primeiro evento e guardar em
v[1] = Event[1].Id;   // = 1, pois a Event-Id já
```

## Veja também

`Event[].Name`

PROPRIEDADE

# Propriedade: Event[].JumpTime

## Sintaxe

```
Event[Evento-Id].JumpTime
```

## Descrição

Com a propriedade JumpTime é possível ler/gravar o tempo interno de ação do evento. Os valores dessa propriedade não são conhecidos com mais precisão, mas a contagem recomeça de 0 para cima a cada ação de movimento.

> **Informação:** Essa propriedade só foi mantida por compatibilidade com o DestinyPatch 1.0 e não deve ser usada, pois a função desejada (verificar se o evento está pulando ou não) não está disponível por ela. Atualmente não é possível verificar se um evento está pulando ou não, já que isso não é possível por parte do RPG_RT. Todas as movimentações dentro de um salto são executadas pelo RPG_RT como um único movimento.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não
- **Volátil:** sim
- **Obsoleto:** sim

## Veja também

`Event`, `Event[]`

PROPRIEDADE

# Propriedade: Event[].MapId

## Sintaxe

```
Event[Evento-Id].MapId
```

## Descrição

Com a propriedade MapId é possível descobrir a Map-Id do evento.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

## Exemplo

```
$
// Descobrir a Map-Id do primeiro evento e guardar em
v[1] = Event[1].MapId;
```

## Veja também

`Event`, `Event[]`, `Event[].X`, `Event[].Y`, `Map.Id`

OBJETO

# Objeto: Event[].MoveRoute

## Sintaxe

```
Event[Evento-Id].MoveRoute
```

## Descrição

Com o objeto MoveRoute é possível acessar o trajeto individual de movimento de um evento. Um trajeto individual de movimento é criado sempre que um evento é movido pelo comando `MoveEvent`. Em todos os outros casos, é usado o trajeto de movimento padrão, conforme a respectiva página do evento.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Active` | Propriedade | Com a propriedade Active é definido se a MoveRoute individual é usada (true) ou não (false). |
| `Count` | Propriedade | Com a propriedade Count é possível consultar a quantidade de movimentos dentro do objeto MoveRoute. |
| `CurrentIndex` | Propriedade | Com a propriedade CurrentIndex é determinado o movimento atual dentro da MoveRoute. |
| `IgnoreIfCantMove` | Propriedade | Com a propriedade IgnoreIfCantMove é determinado se movimentos que não podem ser executados são omitidos (true) ou não (false). |
| `Repeat` | Propriedade | Com a propriedade Repeat é determinado se a MoveRoute se repete (true) ou não (false). |

## Veja também

`Event`, `Event[]`

PROPRIEDADE

# Propriedade: Event[].MoveRoute.Active

## Sintaxe

```
Event[Evento-Id].MoveRoute.Active
```

## Descrição

Com a propriedade Active é definido se a MoveRoute individual é usada (true) ou não (false). Se esse valor for false, é usado o movimento/MoveRoute da respectiva página do evento.

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** não

## Exemplo

```
$

// Interromper o movimento da MoveRoute
Event[HERO].MoveRoute.Active = false;
```

## Veja também

`Event`, `Event[]`, `Event[].MoveRoute`

PROPRIEDADE

# Propriedade: Event[].MoveRoute.Count

## Sintaxe

```
Event[Evento-Id].MoveRoute.Count
```

## Descrição

Com a propriedade Count é possível consultar a quantidade de movimentos dentro do objeto MoveRoute.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** >= 0

## Exemplo

```
$

// Alterar todos os passos de movimento do herói para
for(v[1] = 0; v[1] < Event[HERO].MoveRoute.Count;
  Event[HERO].MoveRoute[v[1]].Command = MC_STEPUP
next;
```

## Veja também

`Event`, `Event[]`, `Event[].MoveRoute`, `Event[].MoveRoute[].Command`

PROPRIEDADE

# Propriedade: Event[].MoveRoute.CurrentIndex

## Sintaxe

```
Event[Evento-Id].MoveRoute.CurrentIndex
```

## Descrição

Com a propriedade CurrentIndex é determinado o movimento atual dentro da MoveRoute. Se o valor for -1, nenhum movimento está sendo executado no momento. Se o valor ficar entre 0 e (`Event[].MoveRoute.Count` - 1), o respectivo movimento está sendo executado.

> **Informação:** Se o valor da propriedade `Active` for false, esse valor é sempre -1, pois, nesse caso, essa MoveRoute não pode ser usada.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não
- **Faixa de valores:** -1; 0 ... (Event[].MoveRoute.Count - 1)

## Exemplo

```
$

// Recomeçar a MoveRoute do início
Event[HERO].MoveRoute.CurrentIndex = 0;
```

## Veja também

`Event`, `Event[]`, `Event[].MoveRoute`, `Event[].MoveRoute.Count`, `Event[].MoveRoute.Active`

PROPRIEDADE

# Propriedade: Event[].MoveRoute.IgnoreIfCantMove

## Sintaxe

```
Event[Evento-Id].MoveRoute.IgnoreIfCantMove
```

## Descrição

Com a propriedade IgnoreIfCantMove é determinado se movimentos que não podem ser executados são omitidos (true) ou não (false). Se um movimento não puder ser executado, continua-se com o próximo movimento que puder ser executado.

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** não

## Exemplo

```
$

// Omitir passos não executáveis
Event[HERO].MoveRoute.IgnoreIfCantMove = true;
```

## Veja também

`Event`, `Event[]`, `Event[].MoveRoute`

PROPRIEDADE

# Propriedade: Event[].MoveRoute.Repeat

## Sintaxe

```
Event[Evento-Id].MoveRoute.Repeat
```

## Descrição

Com a propriedade Repeat é determinado se a MoveRoute se repete (true) ou não (false). Se a MoveRoute se repete, depois do último passo continua-se novamente no primeiro.

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** não

## Exemplo

```
$

// Impedir a repetição da MoveRoute
Event[HERO].MoveRoute.Repeat = false;
```

## Veja também

`Event`, `Event[]`, `Event[].MoveRoute`

OBJETO

# Objeto: Event[].MoveRoute[]

## Sintaxe

```
Event[Evento-Id].MoveRoute[Bewegungsindex]
```

## Descrição

Com o objeto MoveRoute[] é possível acessar um único movimento da MoveRoute.

## Lista de parâmetros

### Parâmetro: `Bewegungsindex`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... (Event[].MoveRoute.Count - 1)
- **Descrição:** O índice do respectivo comando de movimento. Esse valor precisa ficar na faixa de 0 a ( `Event[].MoveRoute.Count` - 1), sendo que 0 descreve o primeiro movimento da MoveRoute.

## Erros possíveis

`DESTINYERROR_ARRAYBOUND`

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Charframe` | Propriedade | Com a propriedade Charframe é possível ler/gravar o índice do Frame a ser usado no comando de movimento MC_SETCHAR. |
| `Charset` | Propriedade | Com a propriedade Charset é possível ler/gravar o Charset a ser usado no comando de movimento MC_SETCHAR. |
| `Command` | Propriedade | Com a propriedade Command é possível ler/gravar o comando do movimento. |
| `Panpot` | Propriedade | Com a propriedade Panpot é possível ler/gravar o balanço do efeito sonoro entre os alto-falantes no comando de movimento MC_PLAYSOUND. |
| `Sound` | Propriedade | Com a propriedade Sound é possível ler/gravar o nome do efeito sonoro no comando de movimento MC_PLAYSOUND. |

| Nome | Tipo | Resumo |
|---|---|---|
| `Switch` | Propriedade | Com a propriedade Switch é possível ler/gravar o switch a ser usado nos comandos de movimento MC_SETSWITCHON e MC_SETSWITCHOFF. |
| `Tempo` | Propriedade | Com a propriedade Tempo é possível ler/gravar a velocidade do efeito sonoro no comando de movimento MC_PLAYSOUND. |
| `Volume` | Propriedade | Com a propriedade Volume é possível ler/gravar o volume do efeito sonoro no comando de movimento MC_PLAYSOUND. |

## Veja também

`Event`, `Event[]`, `Event[].MoveRoute`, `Event[].MoveRoute.Count`

PROPRIEDADE

# Propriedade: Event[].MoveRoute[].Charframe

## Sintaxe

```
Event[Evento-Id].MoveRoute[Bewegungsindex].Charframe
```

## Descrição

Com a propriedade Charframe é possível ler/gravar o índice do `Frame` a ser usado no comando de movimento `MC_SETCHAR`. Junto com a propriedade `Charset`, isso indica o char a ser usado. Esse valor deveria ficar na faixa de 0 ... 7 (veja a figura).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não
- **Faixa de valores:** 0 ... 7

## Exemplo

```
$

// Definir o Charset como Alex no comando de movimento SetChar
for(v[1] = 0; v[1] < Event[HERO].MoveRoute.Count;
  if(Event[HERO].MoveRoute[v[1]].Command == MC_SE
    Event[HERO].MoveRoute[v[1]].Charset   = Actor
    Event[HERO].MoveRoute[v[1]].Charframe = Actor
  endif;
next;
```

## Veja também

`Event`, `Event[]`, `Event[].MoveRoute`, `Event[].MoveRoute[]`, `Event[].MoveRoute[].Command`, `Event[].MoveRoute.Count`, `Event[].MoveRoute[].Charset`, `Actor[].Charset`, `Actor[].Charframe`

PROPRIEDADE

# Propriedade: Event[].MoveRoute[].Charset

## Sintaxe

```
Event[Evento-Id].MoveRoute[Bewegungsindex].Charset
```

## Descrição

Com a propriedade Charset é possível ler/gravar o Charset a ser usado no comando de movimento `MC_SETCHAR`. Em todos os outros comandos de movimento, o valor dessa propriedade é ignorado. Essa propriedade é usada junto com a propriedade `Charframe` para identificar um char.

## Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** não

## Exemplo

```
$

// Definir o Charset como Alex no comando de movimento SetChar
for(v[1] = 0; v[1] < Event[HERO].MoveRoute.Count;
  if(Event[HERO].MoveRoute[v[1]].Command == MC_SE
    Event[HERO].MoveRoute[v[1]].Charset   = Actor
    Event[HERO].MoveRoute[v[1]].Charframe = Actor
  endif;
next;
```

## Veja também

`Event`, `Event[]`, `Event[].MoveRoute`, `Event[].MoveRoute[]`, `Event[].MoveRoute[].Command`, `Event[].MoveRoute.Count`, `Event[].MoveRoute[].Charframe`, `Actor[].Charset`, `Actor[].Charframe`

PROPRIEDADE

# Propriedade: Event[].MoveRoute[].Command

## Sintaxe

```
Event[Evento-Id].MoveRoute[Bewegungsindex].Command
```

## Descrição

Com a propriedade Command é possível ler/gravar o comando do movimento. Para isso são usadas as constantes de `comandos de movimento`:

Constante | Descrição

`MC_STEPUP` Um passo para cima

`MC_STEPRIGHT` Um passo para a direita

`MC_STEPDOWN` Um passo para baixo

`MC_STEPLEFT` Um passo para a esquerda

`MC_STEPRIGHTUP` Um passo na diagonal para a direita e para cima

`MC_STEPRIGHTDOWN` Um passo na diagonal para a direita e para baixo

`MC_STEPLEFTDOWN` Um passo na diagonal para a esquerda e para baixo

`MC_STEPLEFTUP` Um passo na diagonal para a esquerda e para cima

Um passo em uma direção aleatória

`MC_STEPRANDOM`

(cima, esquerda, direita ou baixo)

Um passo em direção ao jogador (cima,

`MC_STEPTOWARDHERO`

esquerda, direita ou baixo)

Um passo para longe do jogador (cima,

`MC_STEPAWAYFROMHERO`

esquerda, direita ou baixo)

Um passo para frente (na direção atual

`MC_STEPFORWARD`

em que olha)

`MC_LOOKUP` Olhar para cima

`MC_LOOKRIGHT` Olhar para a direita

`MC_LOOKDOWN` Olhar para baixo

`MC_LOOKLEFT` Olhar para a esquerda

`MC_TURNRIGHT` Girar 90° para a direita

`MC_TURNLEFT` Girar 90° para a esquerda

`MC_TURN180` Girar 180°

Girar 90° aleatoriamente para a esquerda ou para a direita

`MC_RIGHTLEFTTURN`

`MC_LOOKRANDOM` Olhar em uma direção aleatória

(cima, esquerda, direita ou baixo)

`MC_LOOKTOWARDHERO` Olhar em direção ao jogador

`MC_LOOKAWAYFROMHERO` Olhar para longe do jogador

`MC_WAIT` Aguardar um instante

`MC_STARTJUMPING` Começar a pular

`MC_STOPJUMPING` Parar de pular

`MC_FIXDIRECTION` Fixar a direção em que olha

`MC_RELEASEDIRECTION` Liberar a direção em que olha

`MC_INCREASESPEED` Aumentar a velocidade de movimento

`MC_DECREASESPEED` Diminuir a velocidade de movimento

`MC_INCREASEFREQUENCY` Aumentar a frequência de movimento

`MC_DECREASEFREQUENCY` Diminuir a frequência de movimento

`MC_SETSWITCHON` Ligar um switch

`MC_SETSWITCHOFF` Desligar um switch

`MC_SETCHAR` Definir o char do evento movido

`MC_PLAYSOUND` Reproduzir um som

`MC_STARTPHASING` Ativar o movimento sem colisão

`MC_STOPPHASING` Desativar o movimento sem colisão

`MC_FIXANIMATION` Desativar a animação de movimento

`MC_RELEASEANIMATION` Ativar a animação de movimento

`MC_INCREASETRANSPARENCY` Aumentar a transparência

`MC_DECREASETRANSPARENCY` Diminuir a transparência

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

## Exemplo

```
$

// Alterar todos os passos de movimento do herói para
for(v[1] = 0; v[1] < Event[HERO].MoveRoute.Count;
  Event[HERO].MoveRoute[v[1]].Command = MC_STEPUP
next;
```

## Veja também

`Event`, `Event[]`, `Event[].MoveRoute`, `Event[].MoveRoute[]`, `Event[].MoveRoute.Count`

PROPRIEDADE

# Propriedade: Event[].MoveRoute[].Panpot

## Sintaxe

```
Event[Evento-Id].MoveRoute[Bewegungsindex].Panpot
```

## Descrição

Com a propriedade Panpot é possível ler/gravar o balanço do efeito sonoro entre os alto-falantes no comando de movimento `MC_PLAYSOUND`. A proporção do balanço do som entre os alto-falantes é informada em porcentagem:

0%: o som é tocado apenas pelo alto-falante esquerdo. 50%: o som é distribuído igualmente pelos dois alto-falantes. 100%: o som é tocado apenas pelo alto-falante direito.

Junto com as propriedades `Sound`, `Volume` e `Tempo`, isso indica o som.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não
- **Faixa de valores:** 0 ... 50 ... 100

## Exemplo

```
$

// Definir o som como "Bell" no comando de movimento PlaySound
for(v[1] = 0; v[1] < Event[HERO].MoveRoute.Count;
  if(Event[HERO].MoveRoute[v[1]].Command == MC_PL
    Event[HERO].MoveRoute[v[1]].Sound     = "BELL
    Event[HERO].MoveRoute[v[1]].Volume    = 100;
    Event[HERO].MoveRoute[v[1]].Tempo     = 100;
    Event[HERO].MoveRoute[v[1]].Panpot    = 50;
  endif;
next;
```

## Veja também

`Event`, `Event[]`, `Event[].MoveRoute`, `Event[].MoveRoute[]`, `Event[].MoveRoute[].Command`, `Event[].MoveRoute.Count`, `Event[].MoveRoute[].Sound`, `Event[].MoveRoute[].Volume`, `Event[].MoveRoute[].Tempo`

PROPRIEDADE

# Propriedade: Event[].MoveRoute[].Sound

## Sintaxe

```
Event[Evento-Id].MoveRoute[Bewegungsindex].Sound
```

## Descrição

Com a propriedade Sound é possível ler/gravar o nome do efeito sonoro no comando de movimento `MC_PLAYSOUND`. Junto com as propriedades `Volume`, `Tempo` e `Panpot`, isso indica o som.

## Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** não

## Exemplo

```
$

// Definir o som como "Bell" no comando de movimento PlaySound
for(v[1] = 0; v[1] < Event[HERO].MoveRoute.Count;
  if(Event[HERO].MoveRoute[v[1]].Command == MC_PL
    Event[HERO].MoveRoute[v[1]].Sound     = "BELL
    Event[HERO].MoveRoute[v[1]].Volume    = 100;
    Event[HERO].MoveRoute[v[1]].Tempo     = 100;
    Event[HERO].MoveRoute[v[1]].Panpot    = 50;
  endif;
next;
```

## Veja também

`Event`, `Event[]`, `Event[].MoveRoute`, `Event[].MoveRoute[]`, `Event[].MoveRoute[].Command`, `Event[].MoveRoute.Count`, `Event[].MoveRoute[].Volume`, `Event[].MoveRoute[].Tempo`, `Event[].MoveRoute[].Panpot`

PROPRIEDADE

# Propriedade: Event[].MoveRoute[].Switch

## Sintaxe

```
Event[Evento-Id].MoveRoute[Bewegungsindex].Switch
```

## Descrição

Com a propriedade Switch é possível ler/gravar o switch a ser usado nos comandos de movimento `MC_SETSWITCHON` e `MC_SETSWITCHOFF`. Em todos os outros comandos de movimento, o valor dessa propriedade é ignorado.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

## Exemplo

```
$

// Definir todos os switches como 1 no comando de movimento Switch
for(v[1] = 0; v[1] < Event[HERO].MoveRoute.Count;
  if(Event[HERO].MoveRoute[v[1]].Command == MC_SE
     Event[HERO].MoveRoute[v[1]].Command == MC_SE
    Event[HERO].MoveRoute[v[1]].Switch = 1;
  endif;
next;
```

## Veja também

`Event`, `Event[]`, `Event[].MoveRoute`, `Event[].MoveRoute[]`, `Event[].MoveRoute[].Command`, `Event[].MoveRoute.Count`

PROPRIEDADE

# Propriedade: Event[].MoveRoute[].Tempo

## Sintaxe

```
Event[Evento-Id].MoveRoute[Bewegungsindex].Tempo
```

## Descrição

Com a propriedade Tempo é possível ler/gravar a velocidade do efeito sonoro no comando de movimento `MC_PLAYSOUND`. A indicação é feita em porcentagem (50 = metade da velocidade, 100 = velocidade total). Junto com as propriedades `Sound`, `Volume` e `Panpot`, isso indica o som.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não
- **Faixa de valores:** 50 ... 100 ... 150

## Exemplo

```
$

// Definir o som como "Bell" no comando de movimento PlaySound
for(v[1] = 0; v[1] < Event[HERO].MoveRoute.Count;
  if(Event[HERO].MoveRoute[v[1]].Command == MC_PL
    Event[HERO].MoveRoute[v[1]].Sound     = "BELL
    Event[HERO].MoveRoute[v[1]].Volume    = 100;
    Event[HERO].MoveRoute[v[1]].Tempo     = 100;
    Event[HERO].MoveRoute[v[1]].Panpot    = 50;
  endif;
next;
```

## Veja também

`Event`, `Event[]`, `Event[].MoveRoute`, `Event[].MoveRoute[]`, `Event[].MoveRoute[].Command`, `Event[].MoveRoute.Count`, `Event[].MoveRoute[].Sound`, `Event[].MoveRoute[].Volume`, `Event[].MoveRoute[].Panpot`

PROPRIEDADE

# Propriedade: Event[].MoveRoute[].Volume

## Sintaxe

```
Event[Evento-Id].MoveRoute[Bewegungsindex].Volume
```

## Descrição

Com a propriedade Volume é possível ler/gravar o volume do efeito sonoro no comando de movimento `MC_PLAYSOUND`. A indicação é feita em porcentagem (0 = mudo, 100 = volume total). Junto com as propriedades `Sound`, `Tempo` e `Panpot`, isso indica o som.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não
- **Faixa de valores:** 0 ... 100

## Exemplo

```
$

// Definir o som como "Bell" no comando de movimento PlaySound
for(v[1] = 0; v[1] < Event[HERO].MoveRoute.Count;
  if(Event[HERO].MoveRoute[v[1]].Command == MC_PL
    Event[HERO].MoveRoute[v[1]].Sound     = "BELL
    Event[HERO].MoveRoute[v[1]].Volume    = 100;
    Event[HERO].MoveRoute[v[1]].Tempo     = 100;
    Event[HERO].MoveRoute[v[1]].Panpot    = 50;
  endif;
next;
```

## Veja também

`Event`, `Event[]`, `Event[].MoveRoute`, `Event[].MoveRoute[]`, `Event[].MoveRoute[].Command`, `Event[].MoveRoute.Count`, `Event[].MoveRoute[].Sound`, `Event[].MoveRoute[].Tempo`, `Event[].MoveRoute[].Panpot`

PROPRIEDADE

# Propriedade: Event[].Name

## Sintaxe

```
Event[Evento-Id].Name
```

## Descrição

Com a propriedade Name é possível descobrir o nome do evento. Se o respectivo evento não tiver nome (por exemplo, um dos eventos especiais), é devolvida uma string vazia.

## Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** sim

## Exemplo

```
$
// Descobrir o nome do primeiro evento e guardar em a
a[1] = Event[1].Name;   // por exemplo, "EV0001"
```

## Veja também

`Event[].Id`

PROPRIEDADE

# Propriedade: Event[].Offset

## Sintaxe

```
Event[Evento-Id].Offset
```

## Descrição

Com a propriedade Offset é possível ler/gravar a distância atual do evento até o campo seguinte no mapa. O campo seguinte no mapa é indicado pelas coordenadas `X` e `Y` do evento. A direção da distância é determinada pela propriedade `DirMove`. O valor dessa propriedade é a distância em pixels multiplicada por 16:

O valor 0 significa que atualmente não há distância em relação às coordenadas. O valor 256 significa que atualmente há uma distância de exatamente um campo em relação às coordenadas.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não
- **Volátil:** sim

## Exemplo

```
$
// Posicionar o herói no campo do canto inferior direito
Event[HERO].X = Map.Width - 1;
Event[HERO].Y = Map.Height - 1;
Event[HERO].DirMove = DIR_RIGHT_DOWN;
Event[HERO].DirLook = DIR_RIGHT;
Event[HERO].Offset = 256;

// Trazer o herói para a área visível
Map.ScreenX = Map.Width * 16 - 320;
Map.ScreenY = Map.Height * 16 - 240;
```

## Veja também

`Event`, `Event[]`, `Event[].X`, `Event[].Y`, `Event[].DirMove`, `Event[].DirLook`, `Map.Width`, `Map.Height`, `Map.ScreenX`, `Map.ScreenY`

PROPRIEDADE

# Propriedade: Event[].Phasing

## Sintaxe

```
Event[Evento-Id].Phasing
```

## Descrição

Com a propriedade Phasing é possível ler/gravar a propriedade "atravessar paredes" do evento. False: o evento não pode atravessar paredes etc. True: o evento pode atravessar paredes etc.

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** não
- **Volátil:** sim

## Veja também

`Event`, `Event[]`

PROPRIEDADE

# Propriedade: Event[].ScreenX

## Sintaxe

```
Event[Evento-Id].ScreenX
```

## Descrição

Com a propriedade ScreenX é possível descobrir a coordenada X do evento na tela. Para isso é usado o algoritmo do RPG_RT. A indicação se refere ao centro do evento.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

## Exemplo

```
$
// A coordenada X do evento na tela
v[1] = Event[1].ScreenX;
```

## Veja também

`Event`, `Event[]`, `Event[].ScreenY`

PROPRIEDADE

# Propriedade: Event[].ScreenY

## Sintaxe

```
Event[Evento-Id].ScreenY
```

## Descrição

Com a propriedade ScreenY é possível descobrir a coordenada Y do evento na tela. Para isso é usado o algoritmo do RPG_RT. A indicação se refere à borda inferior do evento.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

## Exemplo

```
$
// A coordenada Y do evento na tela
v[1] = Event[1].ScreenY;
```

## Veja também

`Event`, `Event[]`, `Event[].ScreenX`

PROPRIEDADE

# Propriedade: Event[].Speed

## Sintaxe

```
Event[Evento-Id].Speed
```

## Descrição

Com a propriedade Speed é possível ler/gravar a velocidade do evento.

O valor dessa propriedade indica em quantas unidades o valor da propriedade `Offset` é reduzido por unidade de tempo do jogo:

Valor da propriedade Speed

123456

Redução do valor da propriedade Offset

48163264128

Movimento em pixels por ¼½1248 unidade de tempoPixelPixelPixelPixelPixelPixel

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não
- **Faixa de valores:** 1 ... 6
- **Volátil:** sim

## Veja também

`Event`, `Event[]`

PROPRIEDADE

# Propriedade: Event[].StopAnimation

## Sintaxe

```
Event[Evento-Id].StopAnimation
```

## Descrição

Com a propriedade StopAnimation é possível ler/gravar a propriedade "sem animação de movimento" do evento. False: o evento exibe a animação de movimento normalmente. True: o evento não exibe animação de movimento.

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** não
- **Volátil:** sim

## Veja também

`Event`, `Event[]`

PROPRIEDADE

# Propriedade: Event[].Transparency

## Sintaxe

```
Event[Evento-Id].Transparency
```

## Descrição

Com a propriedade Transparency é possível ler/gravar o valor de transparência do evento. Esse valor deveria ficar na faixa de 0 ... 8, sendo que 0 significa não transparente e 4 significa meio-transparente. Com valores maiores que

8 ocorrem erros de cálculo divertidos (ou seja, o evento fica com uma aparência bem estranha; veja a figura).

> **Informação:** Se, nas propriedades de um herói, a caixa Transparente estiver marcada, isso corresponde a uma transparência de 3.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não
- **Faixa de valores:** 0 ... 8
- **Volátil:** sim

## Veja também

`Event`, `Event[]`

PROPRIEDADE

# Propriedade: Event[].X

## Sintaxe

```
Event[Evento-Id].X
```

## Descrição

Com a propriedade X é possível ler/gravar a coordenada X do evento no mapa. Esse valor deveria ficar na faixa de 0 a (`Map.Width` - 1).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

## Exemplo

```
$
// Mover o primeiro evento 1 campo para a direita
if(Event[1].X + 1 < Map.Width)
  Event[1].X++;
endif;
```

## Veja também

`Event`, `Event[]`, `Event[].Y`, `Event[].MapId`, `Map.Width`

PROPRIEDADE

## Propriedade: Event[].Y

### Sintaxe

```
Event[Evento-Id].Y
```

### Descrição

Com a propriedade Y é possível ler/gravar a coordenada Y do evento no mapa. Esse valor deveria ficar na faixa de 0 a (`Map.Height` - 1).

### Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

### Exemplo

```
$
// Mover o primeiro evento 1 campo para cima (
if(Event[1].X - 1 >= 0)
  Event[1].Y--;
endif;
```

### Veja também

`Event`, `Event[]`, `Event[].X`, `Event[].MapId`, `Map.Height`

OBJETO

# Objeto: f

### Sintaxe

```
f
```

### Descrição

O objeto f serve para tornar acessíveis os Floats (números com vírgula) da Destiny.dll. Os Floats do objeto f são gerenciados pela Destiny.dll.

### Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Count` | Propriedade | Devolve a quantidade de Floats no objeto f. |
| `ReAlloc()` | Método | Com o método ReAlloc é possível aumentar/diminuir a quantidade de Floats no objeto f. |
| `f[]` | Propriedade | Permite o acesso a um determinado Float do objeto f. |

## Veja também

`a`, `d`, `v`, `s`

PROPRIEDADE

# Propriedade: f.Count

## Sintaxe

```
f.Count
```

## Descrição

Devolve a quantidade de Floats no objeto `f`. Isso pode ser usado para fazer um laço sobre todos os Floats.

> **Informação:** A quantidade de Floats disponível ao iniciar o jogo é definida pelo DestinyPatcher.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** >= 0

## Exemplo

```
$
// Definir todos os Floats como 0.5
for(v[1] = 1; v[1] <= f.Count; v[1]++)
  f[v[1]] = 0.5;
next;
```

## Veja também

`f`, `f[]`, `f.ReAlloc()`

MÉTODO

# Método: f.ReAlloc()

## Sintaxe

```
f.ReAlloc(Quantidade)
```

## Descrição

Com o método ReAlloc é possível aumentar/diminuir a quantidade de Floats no objeto `f`.

> **Informação:** Se a quantidade de Floats informada for grande demais, é lançado o erro `DESTINYERROR_MEMORY`. O conteúdo original do objeto `f` é mantido.

> **Informação:** Ao aumentar a quantidade de Floats, os novos Floats são definidos como 0.0.

## Lista de parâmetros

### Parâmetro: `Quantidade`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 0
- **Descrição:** A nova quantidade de Floats no objeto `f`. Esse valor precisa ser maior/igual a 1. Com um número grande demais, a chamada deste método falha.

## Exemplo

```
$
// Aumentar em 1 a quantidade máxima de Floats
f.ReAlloc(f.Count + 1);
// Atribuir ao novo Float o valor PI (depois do
f[f.Count] = Math.Pi;
```

## Erros possíveis

`DESTINYERROR_VALUE`, `DESTINYERROR_MEMORY`

## Veja também

`f`, `f[]`, `f.Count`, `Math.Pi`

PROPRIEDADE

# Propriedade: f[]

## Sintaxe

```
f[Índice]
```

## Descrição

Permite o acesso a um determinado Float do objeto `f`.

## Propriedades

- **Tipo de dado:** `float`
- **Somente leitura:** não

## Lista de parâmetros

### Parâmetro: `Índice`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... f.Count
- **Descrição:** O índice do Float. Precisa estar na faixa de 1 ... `f.Count`.

## Exemplo

```
$
// Definir o valor do primeiro Float como 1.5
f[1] = 1.5;
```

## Erros possíveis

`DESTINYERROR_ARRAYBOUND`

## Veja também

`f`, `f.Count`, `f.ReAlloc()`

OBJETO

# Objeto: Faceset

## Sintaxe

```
Faceset
```

## Descrição

Com o objeto Faceset é possível obter informações sobre o Faceset carregado para a exibição de mensagens, bem como alterar o AuroraSheet do face a ser exibido. Todas as alterações feitas no AuroraSheet ficam visíveis assim que uma mensagem é iniciada/está sendo exibida. Se as alterações forem feitas enquanto a mensagem estiver sendo exibida, elas ficam visíveis imediatamente (no próximo ciclo de desenho de caracteres).

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `AuroraSheet` | Objeto | Com o objeto AuroraSheet é possível acessar o gráfico do face atualmente carregado. |
| `Filename` | Propriedade | Com a propriedade Filename é possível descobrir o nome do arquivo (sem caminho/extensão e relativo à pasta de Facesets). |
| `Flipped` | Propriedade | Com a propriedade Flipped é possível descobrir se o face é exibido espelhado (false = não espelhado; true = espelhado). |
| `Frame` | Propriedade | Com a propriedade Frame é possível descobrir o índice do Frame usado do face. |
| `Position` | Propriedade | Com a propriedade Position é possível ler/gravar a posição de exibição do face (esquerda ou direita). |

OBJETO

# Objeto: Faceset.AuroraSheet

## Sintaxe

```
Faceset.AuroraSheet
```

## Descrição

Com o objeto AuroraSheet é possível acessar o gráfico do face atualmente carregado. Como o objeto AuroraSheet, sendo padronizado, está disponível como subobjeto em vários objetos, sua descrição foi escrita apenas uma vez e colocada como subordinada ao objeto `Picture[]`. A descrição se encontra aqui: `Picture[].AuroraSheet`.

## Exemplo

```
$
// "Inverter" as cores do face usado (P
for(v[1] = 0; v[1] < Faceset.AuroraSheet.Width; v
  for(v[2] = 0; v[2] < Faceset.AuroraSheet.Height
    Faceset.AuroraSheet.Pixel[v[1], v[2]] = 255 -
  next;
next;
```

## Veja também

`Faceset`, `Picture[].AuroraSheet`

PROPRIEDADE

# Propriedade: Faceset.Filename

## Sintaxe

```
Faceset.Filename
```

## Descrição

Com a propriedade Filename é possível descobrir o nome do arquivo (sem caminho/extensão e relativo à pasta de Facesets). Junto com a propriedade `Frame`, isso indica o face atualmente usado.

> **Informação:** Como o RPG Maker guarda internamente uma cópia do face usado (veja `Faceset.AuroraSheet`), não é possível alterar esse valor diretamente. Para usar outro face, é preciso usar o método `ShowFace` do objeto `Command`.

## Sinônimos

Faceset

## Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** sim

## Exemplo

```
$
// Descobrir o Faceset atualmente usado e guardar em
a[1] = Faceset.Filename;
// Descobrir o Frame do face atualmente usado
v[1] = Faceset.Frame;
// Descobrir a posição do face atualmente usada
v[2] = Faceset.Position;
// Descobrir o status de espelhamento atual do face
s[1] = Faceset.Flipped;
// Carregar o face seguinte em cada caso (ou o primeiro,
Command.ShowFace(a[1], (v[1] + 1) % 16, v[2], s[1
```

## Veja também

`Faceset`, `Faceset.Frame`, `Faceset.Position`, `Faceset.Flipped`, `Command.ShowFace()`

PROPRIEDADE

# Propriedade: Faceset.Flipped

## Sintaxe

```
Faceset.Flipped
```

## Descrição

Com a propriedade Flipped é possível descobrir se o face é exibido espelhado (false = não espelhado; true = espelhado).

> **Informação:** Como o RPG Maker guarda internamente uma cópia do face usado (veja `Faceset.AuroraSheet`), não é possível alterar esse valor diretamente. Para usar outro face, é preciso usar o método `ShowFace` do objeto `Command`.

## Sinônimos

Mirrored

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

## Exemplo

```
$
// Descobrir o Faceset atualmente usado e guardar em
a[1] = Faceset.Filename;
// Descobrir o Frame do face atualmente usado
v[1] = Faceset.Frame;
// Descobrir a posição do face atualmente usada
v[2] = Faceset.Position;
// Descobrir o status de espelhamento atual do face
s[1] = Faceset.Flipped;
// Carregar o face seguinte em cada caso (ou o primeiro,
Command.ShowFace(a[1], (v[1] + 1) % 16, v[2], s[1
```

## Veja também

`Faceset`, `Faceset.Filename`, `Faceset.Position`, `Faceset.Flipped`, `Command.ShowFace()`

PROPRIEDADE

# Propriedade: Faceset.Frame

## Sintaxe

```
Faceset.Frame
```

## Descrição

Com a propriedade Frame é possível descobrir o índice do `Frame` usado do face. Junto com a propriedade `Filename`, isso indica o face atualmente usado. Esse valor fica na faixa de 0 ... 15 (veja a figura).

> **Informação:** Como o RPG Maker guarda internamente uma cópia do face usado (veja `Faceset.AuroraSheet`), não é possível alterar esse valor diretamente. Para usar outro face, é preciso usar o método `ShowFace` do objeto `Command`.

## Sinônimos

Faceframe

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** 0 ... 15

## Exemplo

```
$
// Descobrir o Faceset atualmente usado e guardar em
a[1] = Faceset.Filename;
// Descobrir o Frame do face atualmente usado
v[1] = Faceset.Frame;
```

```
// Descobrir a posição do face atualmente usada
v[2] = Faceset.Position;
// Descobrir o status de espelhamento atual do face
s[1] = Faceset.Flipped;
// Carregar o face seguinte em cada caso (ou o primeiro,
Command.ShowFace(a[1], (v[1] + 1) % 16, v[2], s[1
```

## Veja também

`Faceset`, `Faceset.Filename`, `Faceset.Position`, `Faceset.Flipped`, `Command.ShowFace()`

PROPRIEDADE

# Propriedade: Faceset.Position

## Sintaxe

```
Faceset.Position
```

## Descrição

Com a propriedade Position é possível ler/gravar a posição de exibição do face (esquerda ou direita). Para isso podem ser usadas as constantes `FacePosition`.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não
- **Faixa de valores:** 0 ... 1

## Exemplo

```
$
// Descobrir a posição do face atualmente usada
v[1] = Faceset.Position;
// Inverter a posição usada do face (esquerda
if(v[1] == FACEPOSITION_LEFT)
  Faceset.Position = FACEPOSITION_RIGHT;
else
  Faceset.Position = FACEPOSITION_LEFT;
endif;
```

## Observações adicionais

Ao contrário das outras propriedades do objeto Faceset, a propriedade Position não é somente leitura, pois alterar o valor faz diferença.

## Veja também

`Faceset`, `Faceset.Filename`, `Faceset.Position`, `Faceset.Flipped`, `Command.ShowFace()`

OBJETO

# Objeto: File

## Sintaxe

```
File
```

## Descrição

Com o objeto File é possível ler/gravar arquivos. Isso é possível tanto com arquivos de texto quanto com arquivos binários. Assim como no objeto `Directory`, por motivos de segurança só são permitidos acessos dentro da pasta do jogo.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `CloseAll()` | Método | Com o método CloseAll é possível fechar todos os handles de arquivo abertos neste array. |
| `Count` | Propriedade | Com a propriedade Count é possível descobrir a quantidade de handles de arquivo disponíveis neste array. |
| `NextFree()` | Método | Com o método NextFree é possível descobrir o próximo handle de arquivo livre neste array. |
| `ReAlloc()` | Método | Com o método ReAlloc é possível alterar a quantidade de handles de arquivo disponíveis neste array. |
| `File[]` | Objeto | Reúne todas as propriedades e métodos de um handle de arquivo. |

MÉTODO

# Método: File.CloseAll()

## Sintaxe

```
File.CloseAll()
```

## Descrição

Com o método CloseAll é possível fechar todos os handles de arquivo abertos neste array.

## Exemplo

```
$
// Definir a quantidade de handles de arquivo disponíveis como 10
File.ReAlloc(100);
// Abrir 100 arquivos diferentes chamados "Datei_
for(v[1] = 0; v[1] < File.Count; v[1]++)
  File[v[1]].Open("Datei_" .. String.Format("D3",
next;
// Fechar novamente todos os handles de arquivo abertos
File.CloseAll();
```

## Veja também

`File`, `File[]`, `File.Count`, `File.ReAlloc()`, `File[].Open()`, `String.Format()`

PROPRIEDADE

# Propriedade: File.Count

## Sintaxe

```
File.Count
```

## Descrição

Com a propriedade Count é possível descobrir a quantidade de handles de arquivo disponíveis neste array.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

## Exemplo

```
$
// Fechar todos os handles de arquivo individualmente
for(v[1] = 0; v[1] < File.Count; v[1]++)
  File[v[1]].Close();
next;
```

`File`, `File[]`, `File[].Close()`

MÉTODO

# Método: File.NextFree()

## Sintaxe

```
File.NextFree([Índice inicial])
```

## Descrição

Com o método NextFree é possível descobrir o próximo handle de arquivo livre neste array. Se não existir nenhum handle de arquivo livre, o tamanho do array é aumentado em 1, de modo que um novo handle de arquivo (livre) fica disponível.

## Lista de parâmetros

### Parâmetro: `Índice inicial`

- **Tipo de dado:** `dword`

- **Opcional:** sim
- **Valor padrão:** 0
- **Descrição:** Indica o índice a partir do qual a busca começa. Se aqui for informado um valor maior/igual à quantidade de handles de arquivo, ou um valor negativo, um erro é lançado.

### Valor de retorno

- **Tipo de dado:** `dword`
- **Descrição:** O índice do handle de arquivo livre

## Exemplo

```
$
// Definir a quantidade de handles de arquivo disponíveis como 1
File.ReAlloc(1);
// Descobrir o índice do próximo handle de arquivo livre
v[1] = File.NextFree();   // = 0
// Descobrir o índice do próximo handle de arquivo livre
v[2] = File.NextFree();   // = 0 (ainda, já que o
// Descobrir a quantidade de handles de arquivo disponíveis
v[3] = File.Count;        // = 1 (já que existe
// Abrir o handle de arquivo em v[1] (aqui "Test.tx
File[v[1]].Open("Test.txt", FILE_READ | FILE_WRIT
// Descobrir o índice do próximo handle de arquivo livre
v[2] = File.NextFree();   // = 1 (já que o primeiro han
// Descobrir a quantidade de handles de arquivo disponíveis
v[3] = File.Count;        // = 2 (já que o último ha
// Fechar novamente todos os handles de arquivo abertos
File.CloseAll();
```

## Erros possíveis

`DESTINYERROR_VALUE`, `DESTINYERROR_MEMORY`

## Veja também

`File`, `File[]`, `File.Count`, `File.ReAlloc()`, `File.CloseAll()`, `File[].Open()`

MÉTODO

# Método: File.ReAlloc()

## Sintaxe

```
File.ReAlloc(Quantidade)
```

## Descrição

Com o método ReAlloc é possível alterar a quantidade de handles de arquivo disponíveis neste array.

> **Informação:** Depois de iniciar o jogo, a quantidade de handles de arquivo disponíveis é inicializada, por padrão, como 32.

## Lista de parâmetros

### Parâmetro: `Quantidade`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** Indica a quantidade de handles de arquivo disponíveis depois da alteração de tamanho. Se esse valor for menor que a quantidade atual de handles de arquivo disponíveis, então depois disso ficam disponíveis menos handles de arquivo. Se os handles de arquivo no final do array estivessem abertos, eles são fechados automaticamente.

## Exemplo

```
$
// Definir a quantidade de handles de arquivo disponíveis como 10
File.ReAlloc(100);
// Abrir 100 arquivos diferentes chamados "Datei_
for(v[1] = 0; v[1] < File.Count; v[1]++)
  File[v[1]].Open("Datei_" .. String.Format("D3",
next;
// Fechar novamente todos os handles de arquivo abertos
File.CloseAll();
```

## Erros possíveis

`DESTINYERROR_VALUE`, `DESTINYERROR_MEMORY`

## Veja também

`File`, `File[]`, `File.Count`, `File.CloseAll()`, `File[].Open()`, `String.Format()`

OBJETO

# Objeto: File[]

## Sintaxe

```
File[Índice]
```

## Descrição

Reúne todas as propriedades e métodos de um handle de arquivo. Ele é identificado pelo índice. A quantidade de handles de arquivo disponíveis pode ser alterada com `File.ReAlloc()`.

> **Informação:** Depois de iniciar o jogo, a quantidade de handles de arquivo disponíveis é inicializada, por padrão, como 32.

## Lista de parâmetros

### Parâmetro: `Índice`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... (File.Count - 1)
- **Descrição:** Indica o índice do handle de arquivo

## Erros possíveis

`DESTINYERROR_ARRAYBOUND`

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Close()` | Método | Com este método, um arquivo já aberto é fechado. |
| `GetFilePointer()` | Método | Com este método é possível descobrir a posição atual de leitura/escrita do ponteiro de arquivo interno. |
| `Length()` | Método | Com este método é possível descobrir o comprimento atual do arquivo. |
| `Open()` | Método | Com este método, um arquivo é aberto para leitura/escrita. |
| `ReadRawData()` | Método | Com este método são lidos dados de um arquivo. |
| `SetFilePointer()` | Método | Com este método é possível definir a posição atual de leitura/escrita do ponteiro de arquivo interno. |
| `Truncate()` | Método | Com este método, a parte final do arquivo (isto é, a partir da posição do ponteiro de arquivo interno) é cortada. |
| `WriteRawData()` | Método | Com este método, dados são gravados em um arquivo. |

MÉTODO

# Método: File[].Close()

## Sintaxe

```
File[Índice].Close()
```

## Descrição

Com este método, um arquivo já aberto é fechado. Com isso, o respectivo handle de arquivo é liberado novamente e todas as alterações são definitivamente confirmadas. Dependendo do sistema operacional e do hardware, é possível que operações de escrita não tenham efeito imediato e só sejam de fato gravadas depois de uma chamada do método Close(); por isso, a chamada dessa função no final (a partir do ponto em que o arquivo não é mais necessário) é absolutamente necessária.

> **Informação:** Fechar um handle de arquivo que já está fechado não traz nenhum problema.

## Exemplo

```
$
// Abrir o arquivo "Test.txt" pelo primeiro handle
File[0].Open("Test.txt", FILE_WRITE);
// Esvaziar o conteúdo do arquivo (se houver algum
File[0].Truncate();
// Escrever a palavra "Hallo" no arquivo
File[0].WriteRawData("Hallo");
// Fechar novamente o arquivo "Test.txt"
File[0].Close();
```

## Veja também

`File`, `File[].Open()`, `File[].Truncate()`, `File[].WriteRawData()`

MÉTODO

# Método: File[].GetFilePointer()

## Sintaxe

```
File[Índice].GetFilePointer()
```

## Descrição

Com este método é possível descobrir a posição atual de leitura/escrita do ponteiro de arquivo interno.

## Lista de parâmetros

### Valor de retorno

- **Tipo de dado:** `dword`
- **Faixa de valores:** 0 ... File[].Length()
- **Descrição:** A posição atual de leitura/escrita do ponteiro de arquivo interno. Ela fica na faixa de 0 (para o início do arquivo) até `File[].Length()` (para o fim do arquivo, onde não há mais dados).

## Exemplo

```
$
// Abrir o arquivo "Test.txt" pelo primeiro handle
File[0].Open("Test.txt", FILE_WRITE);
// Esvaziar o conteúdo do arquivo (se houver algum
File[0].Truncate();
// Descobrir o ponteiro de arquivo interno atual do primeiro
v[1] = File[0].GetFilePointer();  // = 0, pois o
// Escrever a palavra "Hallo" no arquivo
File[0].WriteRawData("Hallo");
// Descobrir o ponteiro de arquivo interno atual do primeiro
v[2] = File[0].GetFilePointer();  // = 5, pois depois
// Descobrir o ponteiro de arquivo interno atual do primeiro
File[0].SetFilePointer(2);
// Ler o conteúdo do primeiro handle de arquivo a partir
File[0].Truncate();
```

```
// Guardar o comprimento atual do arquivo em v[3]
v[3] = File[0].Length();          // = 2, pois a
// Fechar novamente o arquivo "Test.txt"
File[0].Close();
```

## Erros possíveis

`DESTINYERROR_CANTACCESSFILE`, `DESTINYERROR_FILENOTOPEN`

## Veja também

`File`, `File[].Open()`, `File[].WriteRawData()`, `File[].SetFilePointer()`, `File[].Length()`, `File[].Truncate()`, `File[].Close()`

MÉTODO

# Método: File[].Length()

## Sintaxe

```
File[Índice].Length()
```

## Descrição

Com este método é possível descobrir o comprimento atual do arquivo.

## Lista de parâmetros

### Valor de retorno

- **Tipo de dado:** `dword`
- **Descrição:** O comprimento atual do arquivo. Esse valor pode aumentar durante a execução, quando se escreve no fim do arquivo.

## Exemplo

```
$
// Abrir o arquivo "Test.txt" pelo primeiro handle
File[0].Open("Test.txt", FILE_WRITE);
// Esvaziar o conteúdo do arquivo (se houver algum
File[0].Truncate();
// Escrever a palavra "Hallo" no arquivo
File[0].WriteRawData("Hallo");
// Descobrir o comprimento atual do arquivo e guardar em
v[1] = File[0].Length();           // = 5, pois "Ha
// Fechar novamente o arquivo "Test.txt"
File[0].Close();
```

## Erros possíveis

`DESTINYERROR_CANTACCESSFILE`, `DESTINYERROR_FILENOTOPEN`

## Veja também

`File`, `File[].Open()`, `File[].WriteRawData()`, `File[].Truncate()`, `File[].Close()`

MÉTODO

# Método: File[].Open()

## Sintaxe

```
File[Índice].Open(Nome do arquivo, Modo)
```

## Descrição

Com este método, um arquivo é aberto para leitura/escrita. Se um arquivo for aberto para escrita e ainda não existir, ele é criado. Se um arquivo for aberto apenas para leitura e ainda não existir, um erro é lançado ao abrir. No parâmetro "Dateiname", informa-se o nome do arquivo com o caminho relativo à pasta do jogo. O parâmetro "Modus" indica se o arquivo deve ser aberto para leitura ou escrita. Para isso podem ser usadas as `constantes de modo de arquivo`. Depois que um arquivo é aberto e todos os dados necessários são lidos/gravados, o arquivo precisa, ao final, ser fechado novamente com o método `Close`. Se isso não for feito, talvez não haja handle de arquivo livre disponível para acessos posteriores a arquivos.

## Lista de parâmetros

### Parâmetro: `Nome do arquivo`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** Indica o nome do arquivo (incluindo o caminho relativo) do arquivo a ser aberto. Para as restrições do nome do arquivo, valem as restrições usuais de nomes de arquivo da Microsoft. Assim, os caracteres especiais \\ / : ? * " < > | não podem aparecer no nome do arquivo (barra e barra invertida, porém, são permitidas para indicar subpastas). Também não podem aparecer espaços à direita do nome do arquivo. Só é permitido abrir arquivos dentro da pasta do jogo (e de suas subpastas). Acessos fora da pasta do jogo não são permitidos!

### Parâmetro: `Modo`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... 7
- **Descrição:** Indica se o arquivo deve ser aberto para leitura ou escrita. Para isso podem ser usadas as constantes de modo de arquivo. Se o arquivo deve ser aberto para leitura e escrita, as constantes de modo de arquivo podem ser combinadas com o operador binário OU (por exemplo, FILE_READ | FILE_WRITE). Se um arquivo for aberto no modo Append, o modo de arquivo é automaticamente combinado com o modo de escrita.

## Exemplo

```
$
// Abrir o arquivo "Test.txt" pelo primeiro handle
File[0].Open("Test.txt", FILE_WRITE);
// Esvaziar o conteúdo do arquivo (se houver algum
File[0].Truncate();
```

```
// Escrever a palavra "Hallo" no arquivo
File[0].WriteRawData("Hallo");
// Fechar novamente o arquivo "Test.txt"
File[0].Close();
```

## Erros possíveis

`DESTINYERROR_NOFREEFILEHANDLE`, `DESTINYERROR_CANTRESOLVEPATH`, `DESTINYERROR_NOPERMISSION`, `DESTINYERROR_CANTOPENFILE`, `DESTINYERROR_MEMORY`, `DESTINYERROR_VALUE`

## Veja também

`File`, `File[].Truncate()`, `File[].WriteRawData()`, `File[].Close()`, `Directory.CreateDir()`, `Directory.Rename()`, `Directory.CopyFile()`, `Directory.DeleteFile()`

MÉTODO

# Método: File[].ReadRawData()

## Sintaxe

```
File[Índice].ReadRawData(Datentyp, [Comprimento])
```

## Descrição

Com este método são lidos dados de um arquivo. Por isso, o respectivo handle de arquivo precisa estar aberto com acesso de leitura. Depois que os dados forem lidos, o ponteiro de arquivo interno é automaticamente avançado pela quantidade de bytes lidos.

> **Informação:** Se o arquivo for um arquivo de texto, só deveriam ser lidas Strings.

## Lista de parâmetros

### Parâmetro: `Tipo de dado`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... 7
- **Descrição:** Indica o tipo de dado devolvido por este método. Para isso podem ser usadas as constantes de `tipo de dado`.

### Parâmetro: `Comprimento`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 0
- **Descrição:** Indica a quantidade de bytes a ler. O limite máximo de bytes a ler é determinado pelo tipo de dado. Se aqui for informado 0, é lido o comprimento máximo possível para o respectivo tipo de dado. Apenas no tipo de dado String é obrigatório informar um comprimento aqui. Se aqui forem lidos menos bytes do que o tipo de dado realmente precisa, os bytes faltantes são preenchidos com bytes NULL.

### Valor de retorno

- **Tipo de dado:** `scalar`
- **Descrição:** Os dados lidos, no tipo de dado indicado

## Exemplo

```
$
// Abrir o arquivo "Test.txt" pelo primeiro handle
File[0].Open("Test.txt", FILE_READ);
// Ler todo o conteúdo do arquivo em a[1]
if(File[0].Length() > 0)
  a[1] = File[0].ReadRawData(TYPE_STRING, File[0]
else
  a[1] = "";
endif;
// Fechar novamente o arquivo "Test.txt"
File[0].Close();
```

## Erros possíveis

`DESTINYERROR_MEMORY`, `DESTINYERROR_VALUE`, `DESTINYERROR_CANTACCESSFILE`, `DESTINYERROR_FILENOTOPEN`

## Veja também

`File`, `File[].Open()`, `File[].Length()`, `File[].Close()`

MÉTODO

# Método: File[].SetFilePointer()

## Sintaxe

```
File[Índice].SetFilePointer(Position, [Ponto de referência])
```

## Descrição

Com este método é possível definir a posição atual de leitura/escrita do ponteiro de arquivo interno.

## Lista de parâmetros

### Parâmetro: `Posição`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... File[].Length()
- **Descrição:** A nova posição de leitura/escrita do ponteiro de arquivo interno. Ela fica na faixa de 0 (para o início do arquivo) até `File[].Length()` (para o fim do arquivo, onde não há mais dados). A indicação é feita em relação à posição do segundo parâmetro.

### Parâmetro: `Ponto de referência`

- **Tipo de dado:** `dword`

- **Opcional:** sim
- **Valor padrão:** 0 (= FILEPOS_BEGIN)
- **Faixa de valores:** 0 ... 2
- **Descrição:** Indica o ponto de referência para o novo ponteiro de arquivo interno. Para isso são usadas as `constantes de posição em arquivo`:

Constante | Descrição

A indicação de posição se refere

`FILEPOS_BEGIN` ao início do arquivo, sendo que 0 como nova posição corresponde ao início do arquivo.

A indicação de posição se refere

`FILEPOS_CURRENT` à posição atual do ponteiro de arquivo interno, sendo que 0 como nova posição não causa nenhuma alteração.

A indicação de posição se refere

`FILEPOS_END` ao fim do arquivo, sendo que 0 como nova posição corresponde ao fim do arquivo.

## Exemplo

```
$
// Abrir o arquivo "Test.txt" pelo primeiro handle
File[0].Open("Test.txt", FILE_READ | FILE_WRITE);
// Esvaziar o conteúdo do arquivo (se houver algum
File[0].Truncate();
// Escrever 12 espaços no arquivo
File[0].WriteRawData(String.Fill(" ", 12));
// Posicionar o ponteiro de arquivo interno no início do
File[0].SetFilePointer(0);
// Escrever a palavra "Hallo" no arquivo
File[0].WriteRawData("Hallo");
// Avançar o ponteiro de arquivo interno em 1
File[0].SetFilePointer(1, FILEPOS_CURRENT);
// Escrever a palavra "du" no arquivo
File[0].WriteRawData("du");
// Posicionar o ponteiro de arquivo interno três caracteres
File[0].SetFilePointer(-3, FILEPOS_END);
// Escrever a palavra "da!" no arquivo
File[0].WriteRawData("da!");
// Posicionar o ponteiro de arquivo interno no início do
File[0].SetFilePointer(0);
// Ler o conteúdo do arquivo e guardar em a[1]
a[1] = File[0].ReadRawData(TYPE_STRING, File[0].L
// Fechar novamente o arquivo "Test.txt"
File[0].Close();
```

## Erros possíveis

`DESTINYERROR_VALUE`, `DESTINYERROR_CANTACCESSFILE`, `DESTINYERROR_FILENOTOPEN`

## Veja também

`File`, `File[].Open()`, `File[].ReadRawData()`, `File[].WriteRawData()`, `File[].GetFilePointer()`, `File[].Length()`, `File[].Truncate()`, `File[].Close()`

MÉTODO

# Método: File[].Truncate()

## Sintaxe

```
File[Índice].Truncate()
```

## Descrição

Com este método, a parte final do arquivo (isto é, a partir da posição do ponteiro de arquivo interno) é cortada. Assim, por exemplo, é possível apagar todo o conteúdo de um arquivo, se o ponteiro de arquivo interno estiver apontando para o início do arquivo. Para cortar o conteúdo de um arquivo, ele precisa estar aberto com acesso de escrita.

## Exemplo

```
$
// Abrir o arquivo "Test.txt" pelo primeiro handle
File[0].Open("Test.txt", FILE_WRITE);
// Esvaziar o conteúdo do arquivo (se houver algum
File[0].Truncate();
// Fechar novamente o arquivo "Test.txt"
File[0].Close();
```

## Erros possíveis

`DESTINYERROR_CANTACCESSFILE`, `DESTINYERROR_FILENOTOPEN`

## Veja também

`File`, `File[].Open()`, `File[].Truncate()`, `File[].Close()`

MÉTODO

# Método: File[].WriteRawData()

## Sintaxe

```
File[Índice].WriteRawData(Datenquelle, [Comprimento])
```

## Descrição

Com este método, dados são gravados em um arquivo. Por isso, o respectivo handle de arquivo precisa estar aberto com acesso de escrita. Depois que os dados forem gravados, o ponteiro de arquivo interno é automaticamente avançado pela quantidade de bytes gravados. Se já houver dados na posição do ponteiro de arquivo interno, eles são sobrescritos. Se dados forem gravados no fim do arquivo, o comprimento do arquivo é automaticamente aumentado.

> **Informação:** Se o arquivo for um arquivo de texto, só deveriam ser gravadas Strings.

> **Informação:** Para anexar dados diretamente ao fim do arquivo, recomenda-se abri-lo com o modo `FILE_APPEND`, pois assim o ponteiro de arquivo interno é automaticamente movido para o fim do arquivo ao abrir.

## Lista de parâmetros

### Parâmetro: `Fonte de dados`

- **Tipo de dado:** `any`
- **Opcional:** não
- **Descrição:** Indica a fonte de dados cujo conteúdo deve ser gravado no arquivo.

### Parâmetro: `Comprimento`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 0
- **Descrição:** Indica a quantidade de bytes a gravar. O limite máximo de bytes a gravar é determinado pelo tipo de dado da fonte de dados. Se aqui for informado 0, é gravado no arquivo o comprimento máximo possível para o respectivo tipo de dado (isso também vale aqui para Strings).

## Exemplo

```
$
// Abrir o arquivo "Test.txt" pelo primeiro handle
File[0].Open("Test.txt", FILE_WRITE);
// Esvaziar o conteúdo do arquivo (se houver algum
File[0].Truncate();
// Escrever a palavra "Hallo" no arquivo
File[0].WriteRawData("Hallo");
// Fechar novamente o arquivo "Test.txt"
File[0].Close();
```

## Erros possíveis

`DESTINYERROR_VALUE`, `DESTINYERROR_CANTACCESSFILE`, `DESTINYERROR_FILENOTOPEN`

## Veja também

`File`, `File[].Open()`, `File[].Truncate()`, `File[].Close()`

OBJETO

# Objeto: Game

## Sintaxe

```
Game
```

## Descrição

Com o objeto Game é possível acessar o próprio jogo atual (praticamente o RPG_RT). Além da informação se é um jogo do RPG Maker 2000 ou 2003, aqui também é possível salvar/carregar o save atual, e encerrar o jogo.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Load()` | Método | Com o método Load é carregado um save de um determinado slot (= arquivo no disco). |
| `Quit()` | Método | Encerra o jogo atual e retorna ao sistema operacional (geralmente o Windows). |
| `RM2K` | Propriedade | Com a propriedade RM2K é possível verificar se o jogo é um jogo do RPG Maker 2000. |
| `RM2K3` | Propriedade | Com a propriedade RM2K3 é possível verificar se o jogo é um jogo do RPG Maker 2003. |
| `Save()` | Método | Com o método Save é salvo o save atual em um determinado slot (= arquivo no disco). |
| `SlotExists()` | Método | Com o método SlotExists é verificado se um determinado slot de save (= arquivo no disco) existe e pode ser aberto. |

MÉTODO

# Método: Game.Load()

## Sintaxe

```
Game.Load(Slot)
```

## Descrição

Com o método Load é carregado um save de um determinado slot (= arquivo no disco). O arquivo do slot fica na pasta do jogo e, por padrão, tem o nome SaveXX.lsd, sendo que XX deve ser substituído pelo slot no formato de dois dígitos. As áreas de memória (`a`, `d`, `f`, ...) da Destiny.dll não são carregadas com isso. Para carregar as áreas de memória da Destiny.dll, pode-se usar o método `Load()` do objeto `Destiny`. Se não houver save salvo no respectivo slot (ou se ele não puder ser aberto), um erro é lançado. Porém, a validade dos dados salvos não é verificada.

> **Informação:** Se o slot de save estiver corrompido, o jogo trava ao carregar (por parte do RPG_RT). Isso não pode ser evitado, já que, em tempo de execução, só é possível ter um save carregado por vez. Mesmo que o jogo não travasse ao tentar carregar um save corrompido, o save atual (antes de carregar) já teria sido perdido de qualquer forma. Mas, com a verificação da existência do arquivo do slot de save, já se evita a maioria dos erros possíveis.

> **Informação:** O save, caso exista e possa ser aberto, só é carregado depois do término do DestinyScript atual. Ou seja, comandos de DestinyScript que porventura estejam depois do comando Game.Load() ainda são executados, antes de o save ser carregado.

## Lista de parâmetros

### Parâmetro: `Slot`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** Indica o número do slot de save

## Exemplo

```
$
// Verificar se o primeiro slot de save existe e
if(Game.SlotExists(1))
  Game.Load(1);
endif;
```

## Erros possíveis

`DESTINYERROR_VALUE`, `DESTINYERROR_CANTACCESSSAVESLOT`

## Veja também

`Game`, `Game.Save()`, `Game.SlotExists()`, `Destiny.Load()`

MÉTODO

# Método: Game.Quit()

## Sintaxe

```
Game.Quit()
```

## Descrição

Encerra o jogo atual e retorna ao sistema operacional (geralmente o Windows).

## Exemplo

```
$
// Encerrar o jogo atual
Game.Quit();
```

## Veja também

`Game`, `Command.TitleScreen()`, `Command.GameOver()`

PROPRIEDADE

# Propriedade: Game.RM2K

## Sintaxe

```
Game.RM2K
```

## Descrição

Com a propriedade RM2K é possível verificar se o jogo é um jogo do RPG Maker 2000. Em possíveis versões futuras do DestinyPatch, poderão existir métodos/propriedades disponíveis apenas para o RPG Maker 2003. Por isso, essa propriedade foi criada, para facilitar, em scripts, distinguir qual tipo de RPG_RT o jogo usa.

Para o valor dessa propriedade, vale o seguinte:

Valor | Descrição

False O jogo é um jogo do RPG Maker 2003

True O jogo é um jogo do RPG Maker 2000

## Sinônimos

RM2000

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

## Exemplo

```
$
// Verificar se o jogo é um jogo do RPG Maker 2000
s[1] = Game.RM2K;
```

`Game`, `Game.RM2K3`

PROPRIEDADE

# Propriedade: Game.RM2K3

## Sintaxe

```
Game.RM2K3
```

## Descrição

Com a propriedade RM2K3 é possível verificar se o jogo é um jogo do RPG Maker 2003. Em possíveis versões futuras do DestinyPatch, poderão existir métodos/propriedades disponíveis apenas para o RPG Maker 2003. Por isso, essa propriedade foi criada, para facilitar, em scripts, distinguir qual tipo de RPG_RT o jogo usa.

Para o valor dessa propriedade, vale o seguinte:

Valor | Descrição

False O jogo é um jogo do RPG Maker 2000

True O jogo é um jogo do RPG Maker 2003

## Sinônimos

RM2003

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

## Exemplo

```
$
// Verificar se o jogo é um jogo do RPG Maker 2003
s[1] = Game.RM2K3;
```

`Game`, `Game.RM2K`

MÉTODO

# Método: Game.Save()

## Sintaxe

```
Game.Save(Slot)
```

## Descrição

Com o método Save é salvo o save atual em um determinado slot (= arquivo no disco). O arquivo do slot fica na pasta do jogo e, por padrão, tem o nome SaveXX.lsd, sendo que XX deve ser substituído pelo slot no formato de dois dígitos. As áreas de memória (`a`, `d`, `f`, ...) da Destiny.dll não são salvas com isso. Para salvar as áreas de memória da Destiny.dll, pode-se usar o método `Save()` do objeto `Destiny`. Se o jogo não puder ser salvo, um erro é lançado.

## Lista de parâmetros

### Parâmetro: `Slot`

- **Tipo de dado:** `dword`
- **Opcional:** não

- **Faixa de valores:** >= 1
- **Descrição:** Indica o número do slot de save

## Exemplo

```
$
// Salvar o save atual no primeiro slot
Game.Save(1);
```

## Erros possíveis

`DESTINYERROR_VALUE`, `DESTINYERROR_CANTACCESSSAVESLOT`

## Veja também

`Game`, `Game.Load()`, `Game.SlotExists()`, `Destiny.Save()`

MÉTODO

# Método: Game.SlotExists()

## Sintaxe

```
Game.SlotExists(Slot)
```

## Descrição

Com o método SlotExists é verificado se um determinado slot de save (= arquivo no disco) existe e pode ser aberto. O arquivo do slot fica na pasta do jogo e, por padrão, tem o nome SaveXX.lsd, sendo que XX deve ser substituído pelo slot no formato de dois dígitos.

> **Informação:** Neste método, verifica-se apenas se o slot de save existe e pode ser aberto. Não se verifica, porém, se os dados nesse slot são válidos.

## Lista de parâmetros

### Parâmetro: `Slot`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 1
- **Descrição:** Indica o número do slot de save

## Exemplo

```
$
// Verificar se o primeiro slot de save existe e
if(Game.SlotExists(1))
  Game.Load(1);
endif;
```

## Erros possíveis

`DESTINYERROR_VALUE`, `DESTINYERROR_CANTACCESSSAVESLOT`

## Veja também

`Game`, `Game.Load()`, `Game.SlotExists()`

OBJETO

# Objeto: Item

## Sintaxe

```
Item
```

## Descrição

O objeto Item serve para reunir todas as propriedades de um item (objeto, equipamento etc.). As propriedades gerais ficam reunidas na categoria `Geral`. Propriedades especiais (que valem, por exemplo, só para armas) ficam reunidas no subobjeto correspondente. Para saber que tipo de item se trata, deve-se usar a propriedade Type.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Count` | Propriedade | Devolve a quantidade de itens. |
| `Item[]` | Objeto | O objeto Item serve para reunir todas as propriedades de um item (objeto, equipamento etc.). |

PROPRIEDADE

# Propriedade: Item.Count

## Sintaxe

```
Item.Count
```

## Descrição

Devolve a quantidade de itens. Essa propriedade costuma ser usada para fazer um laço sobre todos os itens.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

## Exemplo

```
$
// Montar uma lista com o nome de todos os itens em a[1]
a[1] = "";
for(v[1] = 1; v[1] < Item.Count; v[1]++)
  a[1] ..= Item[v[1]].Name .. CRLF;
next;
```

OBJETO

# Objeto: Item[]

## Sintaxe

```
Item[Item-Id]
```

## Descrição

O objeto Item serve para reunir todas as propriedades de um item (objeto, equipamento etc.). As propriedades gerais ficam reunidas na categoria `Geral`. Propriedades especiais (que valem, por exemplo, só para armas) ficam reunidas no subobjeto correspondente. Para saber que tipo de item se trata, deve-se usar a propriedade Type.

## Lista de parâmetros

### Parâmetro: `Item-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... Item.Count
- **Descrição:** A Id do item, que pode ser obtida no banco de dados do jogo. Precisa estar na faixa de 1 ... `Item.Count`.

## Erros possíveis

`DESTINYERROR_ARRAYBOUND`

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Geral` | Categoria | A categoria Geral reúne todas as propriedades de um item que valem para qualquer tipo de item. |
| `Material` | Objeto | O objeto Material reúne todas as propriedades específicas de material do objeto Item. |
| `Medicine` | Objeto | O objeto Medicine reúne todas as propriedades específicas de remédio do objeto Item. |
| `Protection` | Objeto | O objeto Protection reúne todas as propriedades específicas de equipamento de proteção do objeto Item. |
| `Scroll` | Objeto | O objeto Scroll reúne todas as propriedades específicas de técnica do objeto Item. |
| `Switch` | Objeto | O objeto Switch reúne todas as propriedades específicas de switch do objeto Item. |
| `Weapon` | Objeto | O objeto Weapon reúne todas as propriedades específicas de arma do objeto Item[]. |

CATEGORIA

# Categoria: Geral

## Descrição

A categoria Geral reúne todas as propriedades de um item que valem para qualquer tipo de item.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Agility` | Propriedade | Com a propriedade Agility é possível saber o valor pelo qual os pontos de agilidade do herói são alterados, caso esse item seja usado. |
| `Attack` | Propriedade | Com a propriedade Attack é possível saber o valor pelo qual os pontos de ataque do herói são alterados, caso esse item seja usado. |
| `ConditionChance` | Propriedade | Indica a chance de que o uso do item cause ou cure as condições associadas a esse item. |
| `Cost` | Propriedade | Com a propriedade Cost é possível saber o preço de compra de um item. |
| `Defense` | Propriedade | Com a propriedade Defense é possível saber o valor pelo qual os pontos de defesa do herói são alterados, caso esse item seja usado. |
| `Description` | Propriedade | Com a propriedade Description é possível saber o texto de descrição do item. |
| `HasAttribute()` | Método | Com o método HasAttribute é possível verificar se um item tem um determinado valor de atributo (True) ou não (False). |
| `HasCondition()` | Método | Com o método HasCondition é possível verificar se um item tem um determinado valor de condição (True) ou não (False). |
| `Mind` | Propriedade | Com a propriedade Mind é possível saber o valor pelo qual os pontos de inteligência do herói são alterados, caso esse item seja usado. |
| `Name` | Propriedade | Com a propriedade Name é possível saber o nome de um item. |
| `NumberOfUses` | Propriedade | Com a propriedade NumberOfUses é possível saber quantas vezes, no máximo, um item pode ser usado. |
| `Type` | Propriedade | Com a propriedade Type é possível saber o tipo de um item. |
| `UsableByActor()` | Método | Com o método UsableByActor é possível verificar se o respectivo item pode ser usado pelo herói indicado (True) ou não (False). |

PROPRIEDADE

# Propriedade: Item[].Agility

## Sintaxe

```
Item[Item-Id].Agility
```

## Descrição

Com a propriedade Agility é possível saber o valor pelo qual os pontos de agilidade do herói são alterados, caso esse item seja usado. O valor dessa propriedade depende, em parte, do tipo do item. Assim, há tipos de item (por exemplo, Switch) para os quais o valor dessa propriedade é inválido.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Item[].Attack

## Sintaxe

```
Item[Item-Id].Attack
```

## Descrição

Com a propriedade Attack é possível saber o valor pelo qual os pontos de ataque do herói são alterados, caso esse item seja usado. O valor dessa propriedade depende, em parte, do tipo do item. Assim, há tipos de item (por exemplo, Switch) para os quais o valor dessa propriedade é inválido.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Item[].ConditionChance

## Sintaxe

```
Item[Item-Id].ConditionChance
```

## Descrição

Indica a chance de que o uso do item cause ou cure as condições associadas a esse item. O significado exato depende do tipo do item.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Item[].Cost

## Sintaxe

```
Item[Item-Id].Cost
```

## Descrição

Com a propriedade Cost é possível saber o preço de compra de um item.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Item[].Defense

## Sintaxe

```
Item[Item-Id].Defense
```

## Descrição

Com a propriedade Defense é possível saber o valor pelo qual os pontos de defesa do herói são alterados, caso esse item seja usado. O valor dessa propriedade depende, em parte, do tipo do item. Assim, há tipos de item (por exemplo, Switch) para os quais o valor dessa propriedade é inválido.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Item[].Description

## Sintaxe

```
Item[Item-Id].Description
```

## Descrição

Com a propriedade Description é possível saber o texto de descrição do item.

## Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** sim

MÉTODO

# Método: Item[].HasAttribute()

## Sintaxe

```
Item[Item-Id].HasAttribute(AtributoId)
```

## Descrição

Com o método HasAttribute é possível verificar se um item tem um determinado valor de atributo (True) ou não (False).

## Lista de parâmetros

### Parâmetro: `AtributoId`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A Id do atributo a verificar.

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** Indica se o item tem ou não o atributo.

## Erros possíveis

`DESTINYERROR_VALUE`

MÉTODO

# Método: Item[].HasCondition()

## Sintaxe

```
Item[Item-Id].HasCondition(CondiçãoId)
```

## Descrição

Com o método HasCondition é possível verificar se um item tem um determinado valor de condição (True) ou não (False).

## Lista de parâmetros

### Parâmetro: `CondiçãoId`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A Id da condição a verificar.

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** Indica se o item tem ou não a condição.

## Erros possíveis

`DESTINYERROR_VALUE`

PROPRIEDADE

# Propriedade: Item[].Mind

## Sintaxe

```
Item[Item-Id].Mind
```

## Descrição

Com a propriedade Mind é possível saber o valor pelo qual os pontos de inteligência do herói são alterados, caso esse item seja usado. O valor dessa propriedade depende, em parte, do tipo do item. Assim, há tipos de item (por exemplo, Switch) para os quais o valor dessa propriedade é inválido.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Item[].Name

## Sintaxe

```
Item[Item-Id].Name
```

## Descrição

Com a propriedade Name é possível saber o nome de um item.

## Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Item[].NumberOfUses

## Sintaxe

```
Item[Item-Id].NumberOfUses
```

## Descrição

Com a propriedade NumberOfUses é possível saber quantas vezes, no máximo, um item pode ser usado. Se essa propriedade tiver o valor da constante `INFINITE_USES`, isso significa que o item pode ser usado infinitas vezes.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Item[].Type

## Sintaxe

```
Item[Item-Id].Type
```

## Descrição

Com a propriedade Type é possível saber o tipo de um item. Para identificar esse valor, podem ser usadas as constantes `ItemType`.

Se o valor dessa propriedade for `ITEMTYPE_COMMON`, só valem as propriedades da categoria `Geral`. Se o valor dessa propriedade for `ITEMTYPE_WEAPON`, além das propriedades da categoria `Geral` também valem as propriedades do subobjeto `Weapon`. Se o valor dessa propriedade for `ITEMTYPE_SHIELD`, `ITEMTYPE_ARMOR`, `ITEMTYPE_HELMET` ou `ITEMTYPE_ACCESSORY`, além das propriedades da categoria `Geral` também valem as propriedades do subobjeto `Protection`. Se o valor dessa propriedade for `ITEMTYPE_MEDICINE`, além das propriedades da categoria `Geral` também valem as propriedades do subobjeto `Medicine`. Se o valor dessa propriedade for `ITEMTYPE_BOOK` ou `ITEMTYPE_UNIQUE`, além das propriedades da categoria `Geral` também valem as propriedades do subobjeto `Scroll`. Se o valor dessa propriedade for `ITEMTYPE_MATERIAL`, além das propriedades da categoria `Geral` também valem as propriedades do subobjeto `Material`. Se o valor dessa propriedade for `ITEMTYPE_SWITCH`, além das propriedades da categoria `Geral` também valem as propriedades do subobjeto `Switch`.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

MÉTODO

# Método: Item[].UsableByActor()

## Sintaxe

```
Item[Item-Id].UsableByActor(HeróiId)
```

## Descrição

Com o método UsableByActor é possível verificar se o respectivo item pode ser usado pelo herói indicado (True) ou não (False).

## Lista de parâmetros

### Parâmetro: `HeróiId`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A Id do herói a verificar.

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** Indica se o herói pode usar o item (True) ou não (False).

## Erros possíveis

`DESTINYERROR_VALUE`

OBJETO

# Objeto: Item[].Material

## Sintaxe

```
Item[Item-Id].Material
```

## Descrição

O objeto Material reúne todas as propriedades específicas de material do objeto Item. Os valores desse objeto só são válidos quando o item também é um material. Por material entende-se objetos que alteram atributos de forma permanente.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `AgilityDiff` | Propriedade | Com a propriedade AgilityDiff é possível saber a quantidade de pontos de agilidade que o jogador ganha ao usar esse objeto (esse valor também pode ser negativo). |

| Nome | Tipo | Resumo |
|---|---|---|
| `AttackDiff` | Propriedade | Com a propriedade AttackDiff é possível saber a quantidade de pontos de ataque que o jogador ganha ao usar esse objeto (esse valor também pode ser negativo). |
| `DefenseDiff` | Propriedade | Com a propriedade DefenseDiff é possível saber a quantidade de pontos de defesa que o jogador ganha ao usar esse objeto (esse valor também pode ser negativo). |
| `MaxHpDiff` | Propriedade | Com a propriedade MaxHpDiff é possível saber a quantidade de pontos de vida máximos que o jogador ganha ao usar esse objeto (esse valor também pode ser negativo). |
| `MaxMpDiff` | Propriedade | Com a propriedade MaxMpDiff é possível saber a quantidade de pontos de magia máximos que o jogador ganha ao usar esse objeto (esse valor também pode ser negativo). |
| `MindDiff` | Propriedade | Com a propriedade MindDiff é possível saber a quantidade de pontos de inteligência que o jogador ganha ao usar esse objeto (esse valor também pode ser negativo). |

PROPRIEDADE

# Propriedade: Item[].Material.AgilityDiff

## Sintaxe

```
Item[Item-Id].Material.AgilityDiff
```

## Descrição

Com a propriedade AgilityDiff é possível saber a quantidade de pontos de agilidade que o jogador ganha ao usar esse objeto (esse valor também pode ser negativo).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Item[].Material.AttackDiff

## Sintaxe

```
Item[Item-Id].Material.AttackDiff
```

## Descrição

Com a propriedade AttackDiff é possível saber a quantidade de pontos de ataque que o jogador ganha ao usar esse objeto (esse valor também pode ser negativo).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Item[].Material.DefenseDiff

### Sintaxe

```
Item[Item-Id].Material.DefenseDiff
```

### Descrição

Com a propriedade DefenseDiff é possível saber a quantidade de pontos de defesa que o jogador ganha ao usar esse objeto (esse valor também pode ser negativo).

### Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Item[].Material.MaxHpDiff

### Sintaxe

```
Item[Item-Id].Material.MaxHpDiff
```

### Descrição

Com a propriedade MaxHpDiff é possível saber a quantidade de pontos de vida máximos que o jogador ganha ao usar esse objeto (esse valor também pode ser negativo).

### Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Item[].Material.MaxMpDiff

### Sintaxe

```
Item[Item-Id].Material.MaxMpDiff
```

### Descrição

Com a propriedade MaxMpDiff é possível saber a quantidade de pontos de magia máximos que o jogador ganha ao usar esse objeto (esse valor também pode ser negativo).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Item[].Material.MindDiff

## Sintaxe

```
Item[Item-Id].Material.MindDiff
```

## Descrição

Com a propriedade MindDiff é possível saber a quantidade de pontos de inteligência que o jogador ganha ao usar esse objeto (esse valor também pode ser negativo).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

OBJETO

# Objeto: Item[].Medicine

## Sintaxe

```
Item[Item-Id].Medicine
```

## Descrição

O objeto Medicine reúne todas as propriedades específicas de remédio do objeto Item. Os valores desse objeto só são válidos quando o item também é um remédio.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `EffectiveOnlyOnDeathHeroes` | Propriedade | Com a propriedade EffectiveOnlyOnDeathHeroes é possível saber se o remédio só pode ser aplicado em heróis mortos. |
| `PercentRecoveryHp` | Propriedade | Com a propriedade PercentRecoveryHp é possível saber a porcentagem de pontos de vida (HP) que o herói recupera ao usar esse remédio. |
| `PercentRecoveryMp` | Propriedade | Com a propriedade PercentRecoveryMp é possível saber a porcentagem de pontos de magia (MP) que o herói recupera ao usar esse remédio. |
| `PointRecoveryHp` | Propriedade | Com a propriedade PointRecoveryHp é possível saber a quantidade de pontos de vida (HP) que o herói recupera ao usar esse remédio. |
| `PointRecoveryMp` | Propriedade | Com a propriedade PointRecoveryMp é possível saber a quantidade de pontos de magia (MP) que o herói recupera ao usar esse remédio. |

| Nome | Tipo | Resumo |
| --- | --- | --- |
| `Scope` | Propriedade | Com a propriedade Scope é possível saber o alcance de efeito do remédio. |
| `UseOnFieldOnly` | Propriedade | Com a propriedade UseOnFieldOnly é possível saber se o remédio só pode ser usado no mapa (ou seja, não no sistema de batalha). |

PROPRIEDADE

## Propriedade: Item[].Medicine.EffectiveOnlyOnDeathHeroes

### Sintaxe

```
Item[Item-Id].Medicine.EffectiveOnlyOnDeathHeroes
```

### Descrição

Com a propriedade EffectiveOnlyOnDeathHeroes é possível saber se o remédio só pode ser aplicado em heróis mortos.

### Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Item[].Medicine.PercentRecoveryHp

### Sintaxe

```
Item[Item-Id].Medicine.PercentRecoveryHp
```

### Descrição

Com a propriedade PercentRecoveryHp é possível saber a porcentagem de pontos de vida (HP) que o herói recupera ao usar esse remédio. Essa porcentagem se refere aos pontos de vida máximos (MaxHP) atuais do herói.

### Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Item[].Medicine.PercentRecoveryMp

### Sintaxe

```
Item[Item-Id].Medicine.PercentRecoveryMp
```

## Descrição

Com a propriedade PercentRecoveryMp é possível saber a porcentagem de pontos de magia (MP) que o herói recupera ao usar esse remédio. Essa porcentagem se refere aos pontos de magia máximos (MaxMP) atuais do herói.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Item[].Medicine.PointRecoveryHp

## Sintaxe

```
Item[Item-Id].Medicine.PointRecoveryHp
```

## Descrição

Com a propriedade PointRecoveryHp é possível saber a quantidade de pontos de vida (HP) que o herói recupera ao usar esse remédio.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Item[].Medicine.PointRecoveryMp

## Sintaxe

```
Item[Item-Id].Medicine.PointRecoveryMp
```

## Descrição

Com a propriedade PointRecoveryMp é possível saber a quantidade de pontos de magia (MP) que o herói recupera ao usar esse remédio.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Item[].Medicine.Scope

### Sintaxe

```
Item[Item-Id].Medicine.Scope
```

### Descrição

Com a propriedade Scope é possível saber o alcance de efeito do remédio. Se o valor dessa propriedade for igual a `ITEMSCOPE_SINGLEALLY`, só um herói pode ser curado por esse remédio. Se o valor for `ITEMSCOPE_ALLALLIES`, ao usá-lo todos os membros do grupo são curados.

### Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Item[].Medicine.UseOnFieldOnly

### Sintaxe

```
Item[Item-Id].Medicine.UseOnFieldOnly
```

### Descrição

Com a propriedade UseOnFieldOnly é possível saber se o remédio só pode ser usado no mapa (ou seja, não no sistema de batalha).

### Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

OBJETO

## Objeto: Item[].Protection

### Sintaxe

```
Item[Item-Id].Protection
```

### Descrição

O objeto Protection reúne todas as propriedades específicas de equipamento de proteção do objeto Item. Os valores desse objeto só são válidos quando o item também é uma peça de equipamento (escudo, armadura, elmo

ou acessório).

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `AvoidPhysicalAttack` | Propriedade | Com a propriedade AvoidPhysicalAttack é possível saber se, ao vestir essa peça de equipamento, ataques físicos são evitados ou reduzidos. |
| `HalfMp` | Propriedade | Com a propriedade HalfMp é possível saber se, ao vestir essa peça de equipamento, os pontos de magia de quem a veste são reduzidos à metade. |
| `PreventCriticalHit` | Propriedade | Com a propriedade PreventCriticalHit é possível saber se, ao vestir essa peça de equipamento, acertos críticos são evitados. |
| `PreventTerrainDamage` | Propriedade | Com a propriedade PreventTerrainDamage é possível saber se, ao vestir essa peça de equipamento, pontos de dano causados pelo terreno do mapa são evitados. |

PROPRIEDADE

# Propriedade: Item[].Protection.AvoidPhysicalAttack

## Sintaxe

```
Item[Item-Id].Protection.AvoidPhysicalAttack
```

## Descrição

Com a propriedade AvoidPhysicalAttack é possível saber se, ao vestir essa peça de equipamento, ataques físicos são evitados ou reduzidos.

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Item[].Protection.HalfMp

## Sintaxe

```
Item[Item-Id].Protection.HalfMp
```

## Descrição

Com a propriedade HalfMp é possível saber se, ao vestir essa peça de equipamento, os pontos de magia de quem a veste são reduzidos à metade.

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Item[].Protection.PreventCriticalHit

### Sintaxe

```
Item[Item-Id].Protection.PreventCriticalHit
```

### Descrição

Com a propriedade PreventCriticalHit é possível saber se, ao vestir essa peça de equipamento, acertos críticos são evitados.

### Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Item[].Protection.PreventTerrainDamage

### Sintaxe

```
Item[Item-Id].Protection.PreventTerrainDamage
```

### Descrição

Com a propriedade PreventTerrainDamage é possível saber se, ao vestir essa peça de equipamento, pontos de dano causados pelo terreno do mapa são evitados.

### Propriedades

- **Tipo de dado:** `bool Schreibgeschützt: ja`

OBJETO

## Objeto: Item[].Scroll

### Sintaxe

```
Item[Item-Id].Scroll
```

### Descrição

O objeto Scroll reúne todas as propriedades específicas de técnica do objeto Item. Os valores desse objeto só são válidos quando o item também está relacionado a técnicas. Por isso, esse objeto é usado tanto em "livros" (itens que ensinam uma técnica ao jogador) quanto em "pergaminhos" (itens que executam a técnica).

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `ShowUseMessage` | Propriedade | Com a propriedade ShowUseMessage é possível saber se, ao usar esse item, aparece a mensagem comum de "usar item" ou a mensagem da técnica associada a ele. |
| `Skill` | Propriedade | Com a propriedade Skill é possível saber qual Skill-Id está associada a esse item. |

PROPRIEDADE

# Propriedade: Item[].Scroll.ShowUseMessage

## Sintaxe

```
Item[Item-Id].Scroll.ShowUseMessage
```

## Descrição

Com a propriedade ShowUseMessage é possível saber se, ao usar esse item, aparece a mensagem comum de "usar item" ou a mensagem da técnica associada a ele. Essa propriedade só é relevante em "pergaminhos" (itens que executam uma técnica).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Item[].Scroll.Skill

## Sintaxe

```
Item[Item-Id].Scroll.Skill
```

## Descrição

Com a propriedade Skill é possível saber qual Skill-Id está associada a esse item. Em "pergaminhos", é a técnica executada ao usá-lo. Em "livros", é a técnica ensinada a um herói ao usá-lo.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

OBJETO

# Objeto: Item[].Switch

## Sintaxe

```
Item[Item-Id].Switch
```

## Descrição

O objeto Switch reúne todas as propriedades específicas de switch do objeto Item. Os valores desse objeto só são válidos quando o item também aciona um switch.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `AvailableOnBattle` | Propriedade | Com a propriedade AvailableOnBattle é possível saber se esse item também pode ser usado durante o combate. |
| `AvailableOnField` | Propriedade | Com a propriedade AvailableOnField é possível saber se esse item também pode ser usado no mapa. |
| `Switch` | Propriedade | Com a propriedade Switch é possível saber a Switch-Id do switch que é acionado ao usar o item. |

PROPRIEDADE

# Propriedade: Item[].Switch.AvailableOnBattle

## Sintaxe

```
Item[Item-Id].Switch.AvailableOnBattle
```

## Descrição

Com a propriedade AvailableOnBattle é possível saber se esse item também pode ser usado durante o combate.

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Item[].Switch.AvailableOnField

## Sintaxe

```
Item[Item-Id].Switch.AvailableOnField
```

## Descrição

Com a propriedade AvailableOnField é possível saber se esse item também pode ser usado no mapa.

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Item[].Switch.Switch

## Sintaxe

```
Item[Item-Id].Switch.Switch
```

## Descrição

Com a propriedade Switch é possível saber a Switch-Id do switch que é acionado ao usar o item.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

OBJETO

# Objeto: Item[].Weapon

## Sintaxe

```
Item[Item-Id].Weapon
```

## Descrição

O objeto Weapon reúne todas as propriedades específicas de arma do objeto Item[]. Os valores desse objeto só são válidos quando o item também é uma arma.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `AttackAllEnemies` | Propriedade | Com a propriedade AttackAllEnemies é possível saber se, num ataque com essa arma, todos os inimigos são atingidos (True) ou apenas um (False). |
| `AttackTwice` | Propriedade | Com a propriedade AttackTwice é possível saber se quem porta a arma ataca duas vezes (True) ou só uma vez (False). |
| `BattleAnimation` | Propriedade | Com a propriedade BattleAnimation é possível saber a BattleAnimation-Id da arma. |
| `CriticalHitChance` | Propriedade | Com a propriedade CriticalHitChance é possível saber qual a chance de obter um acerto crítico. |

| Nome | Tipo | Resumo |
| --- | --- | --- |
| `HitChance` | Propriedade | Com a propriedade HitChance é possível saber qual a taxa de acerto da arma. |
| `IgnoreMonsterEvasion` | Propriedade | Com a propriedade IgnoreMonsterEvasion é possível saber se é impossível para um inimigo esquivar de um ataque com essa arma. |
| `MpCost` | Propriedade | Com a propriedade MpCost é possível saber quanto de MP custa usar a arma. |
| `PreemptiveAttack` | Propriedade | Com a propriedade PreemptiveAttack é possível saber se quem porta a arma pode atacar com prioridade (ou seja, antes de todos os outros) ou não. |
| `TwoHanded` | Propriedade | Com a propriedade TwoHanded é possível saber se uma arma é uma arma de duas mãos. |

PROPRIEDADE

# Propriedade: Item[].Weapon.AttackAllEnemies

## Sintaxe

```
Item[Item-Id].Weapon.AttackAllEnemies
```

## Descrição

Com a propriedade AttackAllEnemies é possível saber se, num ataque com essa arma, todos os inimigos são atingidos (True) ou apenas um (False).

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Item[].Weapon.AttackTwice

## Sintaxe

```
Item[Item-Id].Weapon.AttackTwice
```

## Descrição

Com a propriedade AttackTwice é possível saber se quem porta a arma ataca duas vezes (True) ou só uma vez (False).

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Item[].Weapon.BattleAnimation

### Sintaxe

```
Item[Item-Id].Weapon.BattleAnimation
```

### Descrição

Com a propriedade BattleAnimation é possível saber a BattleAnimation-Id da arma.

### Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Item[].Weapon.CriticalHitChance

### Sintaxe

```
Item[Item-Id].Weapon.CriticalHitChance
```

### Descrição

Com a propriedade CriticalHitChance é possível saber qual a chance de obter um acerto crítico. Esse valor representa uma porcentagem (0 ... 100%).

### Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Item[].Weapon.HitChance

### Sintaxe

```
Item[Item-Id].Weapon.HitChance
```

### Descrição

Com a propriedade HitChance é possível saber qual a taxa de acerto da arma. Esse valor representa uma porcentagem (0 ... 100%).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Item[].Weapon.IgnoreMonsterEvasion

## Sintaxe

```
Item[Item-Id].Weapon.IgnoreMonsterEvasion
```

## Descrição

Com a propriedade IgnoreMonsterEvasion é possível saber se é impossível para um inimigo esquivar de um ataque com essa arma.

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Item[].Weapon.MpCost

## Sintaxe

```
Item[Item-Id].Weapon.MpCost
```

## Descrição

Com a propriedade MpCost é possível saber quanto de MP custa usar a arma. Se um herói não tiver MP suficiente, ele não pode usar essa arma no sistema de batalha padrão.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Item[].Weapon.PreemptiveAttack

## Sintaxe

```
Item[Item-Id].Weapon.PreemptiveAttack
```

## Descrição

Com a propriedade PreemptiveAttack é possível saber se quem porta a arma pode atacar com prioridade (ou seja, antes de todos os outros) ou não.

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Item[].Weapon.TwoHanded

## Sintaxe

```
Item[Item-Id].Weapon.TwoHanded
```

## Descrição

Com a propriedade TwoHanded é possível saber se uma arma é uma arma de duas mãos. Se for o caso, um herói não pode portar essa arma e um escudo (ou outra arma) ao mesmo tempo.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

OBJETO

# Objeto: Keyboard

## Sintaxe

```
Keyboard
```

## Descrição

Com o objeto Keyboard é possível consultar e definir teclas. Os botões do mouse (esquerdo, meio, direito) também são consultados por aqui (para isso usam-se as constantes dos botões do mouse — `VK_LBUTTON`, `VK_MBUTTON` ou `VK_RBUTTON`).

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `GetKey()` | Método | Consulta todas as teclas de 1 a 254 e devolve o primeiro número de tecla correspondente que está pressionado. |
| `GetKeyState()` | Método | Com esta função é consultado o status atual de uma determinada tecla. |

| Nome | Tipo | Resumo |
| --- | --- | --- |
| `GetKeyText()` | Método | Consulta todas as teclas de 1 a 254 e devolve o primeiro número de tecla correspondente que está/esteve pressionado, considerando a repetição de caractere. |
| `SetKeyState()` | Método | Com esta função é possível definir o status de uma determinada tecla. |

MÉTODO

# Método: Keyboard.GetKey()

## Sintaxe

```
Keyboard.GetKey([Tecla])
```

## Descrição

Consulta todas as teclas de 1 a 254 e devolve o primeiro número de tecla correspondente que está pressionado. Se nenhuma tecla estiver pressionada, é devolvido 0. Como parâmetro, pode-se informar opcionalmente onde a busca começa. A esse valor soma-se 1 no início.

## Lista de parâmetros

### Parâmetro: `Tecla`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 0
- **Descrição:** Indica o número de tecla a partir do qual a busca deve começar. Isso é útil para permitir vários pressionamentos de tecla simultâneos. Para isso, informa-se o valor de retorno da última chamada. Se esse parâmetro não for informado, a busca do número de tecla ativo começa novamente em 0.

### Valor de retorno

- **Tipo de dado:** `dword`
- **Descrição:** Devolve 0 se nenhuma tecla estiver pressionada. Caso contrário, o resultado é um `código virtual de tecla`.

## Exemplo

```
$
// Aguardar até que uma tecla seja pressionada (o número
v[1] = 0;
do while(v[1] == 0)
  v[1] = Keyboard.GetKey();
loop;
```

## Veja também

`Keyboard.GetKeyState()`, `Keyboard.GetKeyText()`, `Keyboard.SetKeyState()`

MÉTODO

# Método: Keyboard.GetKeyState()

## Sintaxe

```
Keyboard.GetKeyState(Tecla)
```

## Descrição

Com esta função é consultado o status atual de uma determinada tecla. Para isso podem ser usadas as constantes dos `códigos virtuais de tecla`. Esta função devolve um valor diferente de zero se a tecla estiver/esteve pressionada.

> **Informação:** Esta função é equivalente à função GetAsyncKeyState do Windows.

## Lista de parâmetros

### Parâmetro: `Tecla`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** Indica o número da tecla a consultar. Para isso são usados os `códigos virtuais de tecla`.

### Valor de retorno

- **Tipo de dado:** `word`
- **Descrição:** Valor de retorno | Descrição

0 A tecla não está pressionada no momento e também não foi pressionada desde a última consulta.

1 A tecla não está pressionada no momento, mas foi pressionada desde a última consulta.

-32767 A tecla está pressionada no momento, mas foi pressionada desde a última consulta.

-32768 A tecla está pressionada no momento e a repetição de caractere se aplica.

## Exemplo

```
$
// Verificar se a tecla "Seta para baixo" está pressio
if(Keyboard.GetKeyState(VK_DOWN))
  // ... (a tecla está pressionada)
else
  // ... (a tecla não está pressionada)
endif;
```

## Veja também

`Keyboard.GetKey()`, `Keyboard.GetKeyText()`, `Keyboard.SetKeyState()`

MÉTODO

# Método: Keyboard.GetKeyText()

## Sintaxe

```
Keyboard.GetKeyText([Tecla])
```

## Descrição

Consulta todas as teclas de 1 a 254 e devolve o primeiro número de tecla correspondente que está/esteve pressionado, considerando a repetição de caractere. Se nenhuma tecla estiver/esteve pressionada, é devolvido 0. Como parâmetro, pode-se informar opcionalmente onde a busca começa. A esse valor soma-se 1 no início.

## Lista de parâmetros

### Parâmetro: `Tecla`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 0
- **Descrição:** Indica o número de tecla a partir do qual a busca deve começar. Isso é útil para permitir vários pressionamentos de tecla simultâneos. Para isso, informa-se o valor de retorno da última chamada. Se esse parâmetro não for informado, a busca do número de tecla ativo começa novamente em 0.

### Valor de retorno

- **Tipo de dado:** `dword`
- **Descrição:** Devolve 0 se nenhuma tecla estiver pressionada. Caso contrário, o resultado é um `código virtual de tecla`.

## Exemplo

```
$
// Deixar o jogador digitar um texto de 10 caracteres
v[1] = 0;
do while(String.Length(a[1]) < 10)
  v[1] = Keyboard.GetKey(v[1]);
  a[1] ..= String.Chr(v[1]);
loop;
```

## Veja também

`Keyboard.GetKeyState()`, `Keyboard.GetKeyText()`, `Keyboard.SetKeyState()`, `String.Chr()`, `String.Length()`

MÉTODO

# Método: Keyboard.SetKeyState()

## Sintaxe

```
Keyboard.SetKeyState(Tecla, Status)
```

## Descrição

Com esta função é possível definir o status de uma determinada tecla.

> **Informação:** Esta função é equivalente à função keybd_event do Windows.

> **Informação:** O plural de "status" em português também é "status" — por isso, isto aqui na ajuda não é um erro de digitação...

## Lista de parâmetros

### Parâmetro: `Tecla`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** Indica o número da tecla a definir. Para isso são usados os `códigos virtuais de tecla`.

### Parâmetro: `Status`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** O status da tecla. Para isso são usadas as constantes de `eventos de tecla`.

## Exemplo

No final, o jogador tentaria ir para a direita, porque o computador considera que a tecla de seta direita está pressionada. Só quando a tecla de seta direita for solta (nesse caso seria preciso, na realidade, até pressioná-la primeiro!) o jogador deixaria de tentar ir para a direita. Para "soltar" a tecla de seta direita no código, pode-se usar a constante KEYEVENTF_KEYUP em vez da constante KEYEVENTF_KEYDOWN.

```
$
// Simular o pressionamento da tecla de seta direita
Keyboard.SetKeyState(VK_RIGHT, KEYEVENTF_KEYDOWN)
```

## Veja também

`Keyboard.GetKeyState()`, `Keyboard.GetKey()`, `Keyboard.GetKeyText()`

OBJETO

# Objeto: Logic

## Sintaxe

```
Logic
```

## Descrição

Com o objeto Logic é possível realizar operações lógicas e comparações. A maioria das funcionalidades do objeto Logic hoje já é atendida pela própria linguagem de script, por isso a maioria dos métodos está obsoleta e não deveria mais ser usada.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Operações lógicas` | Categoria | Nesta categoria estão reunidas todas as operações lógicas do objeto Logic. |
| `Operações básicas` | Categoria | Contém as operações básicas, a partir das quais todas as outras operações lógicas são compostas. |
| `Operações compostas` | Categoria | Contém as operações lógicas compostas a partir das operações básicas. |
| `Comparações` | Categoria | Nesta categoria estão reunidas todas as comparações do objeto Logic. |

CATEGORIA

# Categoria: Operações lógicas

## Descrição

Nesta categoria estão reunidas todas as operações lógicas do objeto `Logic`. As operações básicas E, OU e NÃO já são suportadas pela própria linguagem de script e, por isso, na verdade já não são mais necessárias. Ao chamar esses métodos, portanto, é executado o mesmo que nos respectivos operadores.

CATEGORIA

# Categoria: Operações básicas

## Descrição

Contém as operações básicas, a partir das quais todas as outras operações lógicas são compostas.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `And()` | Método | Considera o valor dos parâmetros "Valor1" e "Valor2" como valores booleanos e executa uma operação E (AND) entre esses dois valores. |
| `Not()` | Método | Considera o valor do parâmetro "Valor1" como um booleano e inverte o valor desse booleano. |

| Nome | Tipo | Resumo |
|---|---|---|
| `Or()` | Método | Considera o valor dos parâmetros "Valor1" e "Valor2" como valores booleanos e executa uma operação OU (OR) entre esses dois valores. |

MÉTODO

# Método: Logic.And()

## Sintaxe

```
Logic.And(Valor1, Valor2)
```

## Descrição

Considera o valor dos parâmetros "Valor1" e "Valor2" como valores booleanos e executa uma operação E (AND) entre esses dois valores.

Alteração em relação ao DestinyPatch 1.0: se aqui for informada uma String como parâmetro, uma string vazia é considerada falsa (false) e uma string não vazia é considerada verdadeira (true). Assim, uma string que contenha "0" também é considerada verdadeira. Se isso não for desejado, a string deveria ser convertida antes para um número. Isso pode ser feito com o objeto `Convert`.

## Propriedades

- **Obsoleto:** sim

## Lista de parâmetros

### Parâmetro: `Valor1`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Parâmetro: `Valor2`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** O resultado da operação E (AND) de Valor1 e Valor2

## Tabela-verdade

| Valor1 | Valor2 | Retorno |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |

| Valor1 | Valor2 | Retorno |
|---|---|---|
| 1 | 0 | 0 |
| 1 | 1 | 1 |

## Exemplo

```
$
// Uma operação E entre os dois primeiros switches
s[3] = Logic.And(s[1], s[2]);
```

## Observações adicionais

Este método deve ser considerado obsoleto, já que o mesmo efeito pode ser obtido com o `operador lógico E`.

```
$
// Obter o valor dos dois primeiros switches pelo operador lógico
s[3] = s[1] && s[2];
```

## Veja também

`Logic.And3()`, `Logic.Or()`, `Logic.Not()`

MÉTODO

# Método: Logic.Not()

## Sintaxe

```
Logic.Not(Valor1)
```

## Descrição

Considera o valor do parâmetro "Valor1" como um booleano e inverte o valor desse booleano.

Alteração em relação ao DestinyPatch 1.0: se aqui for informada uma String como parâmetro, uma string vazia é considerada falsa (false) e uma string não vazia é considerada verdadeira (true). Assim, uma string que contenha "0" também é considerada verdadeira. Se isso não for desejado, a string deveria ser convertida antes para um número. Isso pode ser feito com o objeto `Convert`.

## Propriedades

- **Obsoleto:** sim

## Lista de parâmetros

### Parâmetro: `Valor1`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer, cujo estado booleano (ligado/desligado) deve ser invertido

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** O valor invertido do parâmetro Valor1

## Tabela-verdade

| Valor1 | Retorno |
|---|---|
| 0 | 1 |
| 1 | 0 |

## Exemplo

```
$
// Inverter o valor do primeiro switch e guardar
s[2] = Logic.Not(s[1]);
```

## Observações adicionais

Este método deve ser considerado obsoleto, já que o mesmo efeito pode ser obtido com o `operador de negação lógica`.

```
$
// Obter o valor invertido do primeiro switch pelo
s[2] = !s[1];
```

## Veja também

`Logic.And()`, `Logic.Or()`

MÉTODO

# Método: Logic.Or()

## Sintaxe

```
Logic.Or(Valor1, Valor2)
```

## Descrição

Considera o valor dos parâmetros "Valor1" e "Valor2" como valores booleanos e executa uma operação OU (OR) entre esses dois valores.

Alteração em relação ao DestinyPatch 1.0: se aqui for informada uma String como parâmetro, uma string vazia é considerada falsa (false) e uma string não vazia é considerada verdadeira (true). Assim, uma string que contenha "0" também é considerada verdadeira. Se isso não for desejado, a string deveria ser convertida antes para um número. Isso pode ser feito com o objeto `Convert`.

## Propriedades

- **Obsoleto:** sim

## Lista de parâmetros

### Parâmetro: `Valor1`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Parâmetro: `Valor2`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** O resultado da operação OU (OR) de Valor1 e Valor2

## Tabela-verdade

| Valor1 | Valor2 | Retorno |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

## Exemplo

```
$
// Uma operação OU entre os dois primeiros switches
s[3] = Logic.Or(s[1], s[2]);
```

## Observações adicionais

Este método deve ser considerado obsoleto, já que o mesmo efeito pode ser obtido com o `operador lógico OU`.

```
$
// Obter o valor dos dois primeiros switches pelo operador lógico
s[3] = s[1] || s[2];
```

## Veja também

`Logic.Or3()`, `Logic.And()`, `Logic.Not()`

CATEGORIA

# Categoria: Operações compostas

## Descrição

Contém as operações lógicas compostas a partir das operações básicas.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `And3()` | Método | Considera o valor dos parâmetros "Valor1", "Valor2" e "Valor3" como valores booleanos e executa uma operação E (AND) entre esses três valores. |
| `Imp()` | Método | Considera o valor dos parâmetros "Valor1" e "Valor2" como valores booleanos e executa uma operação de Implicação entre esses dois valores. |
| `Inh()` | Método | Considera o valor dos parâmetros "Valor1" e "Valor2" como valores booleanos e executa uma operação de Inibição (Inhibit) entre esses dois valores. |
| `Nand()` | Método | Considera o valor dos parâmetros "Valor1" e "Valor2" como valores booleanos e executa uma operação NAND entre esses dois valores. |
| `Nand3()` | Método | Considera o valor dos parâmetros "Valor1", "Valor2" e "Valor3" como valores booleanos e executa uma operação NAND entre esses três valores. |
| `Nor()` | Método | Considera o valor dos parâmetros "Valor1" e "Valor2" como valores booleanos e executa uma operação NOR entre esses dois valores. |
| `Nor3()` | Método | Considera o valor dos parâmetros "Valor1", "Valor2" e "Valor3" como valores booleanos e executa uma operação NOR entre esses três valores. |
| `Or3()` | Método | Considera o valor dos parâmetros "Valor1", "Valor2" e "Valor3" como valores booleanos e executa uma operação OU (OR) entre esses três valores. |
| `Xnor()` | Método | Considera o valor dos parâmetros "Valor1" e "Valor2" como valores booleanos e executa uma operação de Equivalência (XNOR) entre esses dois valores. |
| `Xnor3()` | Método | Considera o valor dos parâmetros "Valor1", "Valor2" e "Valor3" como valores booleanos e executa uma operação de Equivalência (XNOR) entre esses três valores. |
| `Xor()` | Método | Considera o valor dos parâmetros "Valor1" e "Valor2" como valores booleanos e executa uma operação OU-EXCLUSIVO (também chamada de Antivalência) entre esses dois valores. |
| `Xor3()` | Método | Considera o valor dos parâmetros "Valor1", "Valor2" e "Valor3" como valores booleanos e executa uma operação OU-EXCLUSIVO (também chamada de Antivalência) entre esses três valores. |

MÉTODO

# Método: Logic.And3()

## Sintaxe

```
Logic.And3(Valor1, Valor2, Valor3)
```

## Descrição

Considera o valor dos parâmetros "Valor1", "Valor2" e "Valor3" como valores booleanos e executa uma operação E (AND) entre esses três valores.

Alteração em relação ao DestinyPatch 1.0: se aqui for informada uma String como parâmetro, uma string vazia é considerada falsa (false) e uma string não vazia é considerada verdadeira (true). Assim, uma string que contenha "0" também é considerada verdadeira. Se isso não for desejado, a string deveria ser convertida antes para um número. Isso pode ser feito com o objeto `Convert`.

## Lista de parâmetros

### Parâmetro: `Valor1`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Parâmetro: `Valor2`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Parâmetro: `Valor3`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** O resultado da operação E (AND) de Valor1, Valor2 e Valor3

## Tabela-verdade

| Valor1 | Valor2 | Valor3 | Retorno |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

## Exemplo

```
$
// Uma operação E entre os três primeiros switches
s[4] = Logic.And3(s[1], s[2], s[3]);
```

## Veja também

`Logic.And()`, `Logic.Or3()`, `Logic.And()`, `Logic.Or()`, `Logic.Not()`

MÉTODO

# Método: Logic.Imp()

## Sintaxe

```
Logic.Imp(Valor1, Valor2)
```

## Descrição

Considera o valor dos parâmetros "Valor1" e "Valor2" como valores booleanos e executa uma operação de Implicação entre esses dois valores.

Alteração em relação ao DestinyPatch 1.0: se aqui for informada uma String como parâmetro, uma string vazia é considerada falsa (false) e uma string não vazia é considerada verdadeira (true). Assim, uma string que contenha "0" também é considerada verdadeira. Se isso não for desejado, a string deveria ser convertida antes para um número. Isso pode ser feito com o objeto `Convert`.

## Lista de parâmetros

### Parâmetro: `Valor1`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Parâmetro: `Valor2`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** O resultado da operação de Implicação de Valor1 e Valor2

## Tabela-verdade

| Valor1 | Valor2 | Retorno |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

## Exemplo

```
$
// Uma operação de Implicação entre os dois primeiros
s[3] = Logic.Imp(s[1], s[2]);
```

## Veja também

`Logic.Inh()`, `Logic.And()`, `Logic.Or()`, `Logic.Not()`

MÉTODO

# Método: Logic.Inh()

## Sintaxe

```
Logic.Inh(Valor1, Valor2)
```

## Descrição

Considera o valor dos parâmetros "Valor1" e "Valor2" como valores booleanos e executa uma operação de Inibição (Inhibit) entre esses dois valores.

Alteração em relação ao DestinyPatch 1.0: se aqui for informada uma String como parâmetro, uma string vazia é considerada falsa (false) e uma string não vazia é considerada verdadeira (true). Assim, uma string que contenha "0" também é considerada verdadeira. Se isso não for desejado, a string deveria ser convertida antes para um número. Isso pode ser feito com o objeto `Convert`.

## Lista de parâmetros

### Parâmetro: `Valor1`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Parâmetro: `Valor2`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** O resultado da operação de Inibição (Inhibit) de Valor1 e Valor2

## Tabela-verdade

| Valor1 | Valor2 | Retorno |
|---|---|---|
| 0 | 0 | 0 |

| Valor1 | Valor2 | Retorno |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

## Exemplo

```
$
// Uma operação de Inibição entre os dois primeiros
s[3] = Logic.Inh(s[1], s[2]);
```

## Veja também

`Logic.Imp()`, `Logic.And()`, `Logic.Or()`, `Logic.Not()`

MÉTODO

# Método: Logic.Nand()

## Sintaxe

```
Logic.Nand(Valor1, Valor2)
```

## Descrição

Considera o valor dos parâmetros "Valor1" e "Valor2" como valores booleanos e executa uma operação NAND entre esses dois valores.

Alteração em relação ao DestinyPatch 1.0: se aqui for informada uma String como parâmetro, uma string vazia é considerada falsa (false) e uma string não vazia é considerada verdadeira (true). Assim, uma string que contenha "0" também é considerada verdadeira. Se isso não for desejado, a string deveria ser convertida antes para um número. Isso pode ser feito com o objeto `Convert`.

## Lista de parâmetros

### Parâmetro: `Valor1`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Parâmetro: `Valor2`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Valor de retorno

- **Tipo de dado:** `bool`

- **Descrição:** O resultado da operação NAND de Valor1 e Valor2

## Tabela-verdade

| Valor1 | Valor2 | Retorno |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

## Exemplo

```
$
// Uma operação NAND entre os dois primeiros switches
s[3] = Logic.Nand(s[1], s[2]);
```

## Veja também

`Logic.Nand3()`, `Logic.Nor()`, `Logic.And()`, `Logic.Or()`, `Logic.Not()`

MÉTODO

# Método: Logic.Nand3()

## Sintaxe

```
Logic.Nand3(Valor1, Valor2, Valor3)
```

## Descrição

Considera o valor dos parâmetros "Valor1", "Valor2" e "Valor3" como valores booleanos e executa uma operação NAND entre esses três valores.

Alteração em relação ao DestinyPatch 1.0: se aqui for informada uma String como parâmetro, uma string vazia é considerada falsa (false) e uma string não vazia é considerada verdadeira (true). Assim, uma string que contenha "0" também é considerada verdadeira. Se isso não for desejado, a string deveria ser convertida antes para um número. Isso pode ser feito com o objeto `Convert`.

## Lista de parâmetros

### Parâmetro: `Valor1`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Parâmetro: `Valor2`

- **Tipo de dado:** `scalar`
- **Opcional:** não

- **Descrição:** Um valor qualquer

### Parâmetro: `Valor3`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** O resultado da operação NAND de Valor1, Valor2 e Valor3

## Tabela-verdade

| Valor1 | Valor2 | Valor3 | Retorno |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

## Exemplo

```
$
// Uma operação NAND entre os três primeiros switches
s[4] = Logic.Nand3(s[1], s[2], s[3]);
```

## Veja também

`Logic.Nand()`, `Logic.Nor3()`, `Logic.And()`, `Logic.Or()`, `Logic.Not()`

MÉTODO

# Método: Logic.Nor()

## Sintaxe

```
Logic.Nor(Valor1, Valor2)
```

## Descrição

Considera o valor dos parâmetros "Valor1" e "Valor2" como valores booleanos e executa uma operação NOR entre esses dois valores.

Alteração em relação ao DestinyPatch 1.0: se aqui for informada uma String como parâmetro, uma string vazia é considerada falsa (false) e uma string não vazia é considerada verdadeira (true). Assim, uma string que contenha "0" também é considerada verdadeira. Se isso não for desejado, a string deveria ser convertida antes para um número. Isso pode ser feito com o objeto `Convert`.

## Lista de parâmetros

### Parâmetro: `Valor1`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Parâmetro: `Valor2`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** O resultado da operação NOR de Valor1 e Valor2

## Tabela-verdade

| Valor1 | Valor2 | Retorno |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

## Exemplo

```
$
// Uma operação NOR entre os dois primeiros switches
s[3] = Logic.Nor(s[1], s[2]);
```

## Veja também

`Logic.Nor3()`, `Logic.Nand()`, `Logic.And()`, `Logic.Or()`, `Logic.Not()`

MÉTODO

# Método: Logic.Nor3()

## Sintaxe

```
Logic.Nor3(Valor1, Valor2, Valor3)
```

## Descrição

Considera o valor dos parâmetros "Valor1", "Valor2" e "Valor3" como valores booleanos e executa uma operação NOR entre esses três valores.

Alteração em relação ao DestinyPatch 1.0: se aqui for informada uma String como parâmetro, uma string vazia é considerada falsa (false) e uma string não vazia é considerada verdadeira (true). Assim, uma string que contenha "0" também é considerada verdadeira. Se isso não for desejado, a string deveria ser convertida antes para um número. Isso pode ser feito com o objeto `Convert`.

## Lista de parâmetros

### Parâmetro: `Valor1`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Parâmetro: `Valor2`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Parâmetro: `Valor3`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** O resultado da operação NOR de Valor1, Valor2 e Valor3

## Tabela-verdade

| Valor1 | Valor2 | Valor3 | Retorno |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

## Exemplo

```
$
// Uma operação NOR entre os três primeiros switches
s[4] = Logic.Nor3(s[1], s[2], s[3]);
```

## Veja também

`Logic.Nor()`, `Logic.Nand3()`, `Logic.And()`, `Logic.Or()`, `Logic.Not()`

MÉTODO

# Método: Logic.Or3()

## Sintaxe

```
Logic.Or3(Valor1, Valor2, Valor3)
```

## Descrição

Considera o valor dos parâmetros "Valor1", "Valor2" e "Valor3" como valores booleanos e executa uma operação OU (OR) entre esses três valores.

Alteração em relação ao DestinyPatch 1.0: se aqui for informada uma String como parâmetro, uma string vazia é considerada falsa (false) e uma string não vazia é considerada verdadeira (true). Assim, uma string que contenha "0" também é considerada verdadeira. Se isso não for desejado, a string deveria ser convertida antes para um número. Isso pode ser feito com o objeto `Convert`.

## Lista de parâmetros

### Parâmetro: `Valor1`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Parâmetro: `Valor2`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Parâmetro: `Valor3`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** O resultado da operação OU (OR) de Valor1, Valor2 e Valor3

## Tabela-verdade

| Valor1 | Valor2 | Valor3 | Retorno |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

## Exemplo

```
$
// Uma operação OU entre os três primeiros switches
s[4] = Logic.Or3(s[1], s[2], s[3]);
```

## Veja também

`Logic.Or()`, `Logic.And3()`, `Logic.And()`, `Logic.Or()`, `Logic.Not()`

MÉTODO

# Método: Logic.Xnor()

## Sintaxe

```
Logic.Xnor(Valor1, Valor2)
```

## Descrição

Considera o valor dos parâmetros "Valor1" e "Valor2" como valores booleanos e executa uma operação de Equivalência (XNOR) entre esses dois valores.

Alteração em relação ao DestinyPatch 1.0: se aqui for informada uma String como parâmetro, uma string vazia é considerada falsa (false) e uma string não vazia é considerada verdadeira (true). Assim, uma string que contenha "0" também é considerada verdadeira. Se isso não for desejado, a string deveria ser convertida antes para um número. Isso pode ser feito com o objeto `Convert`.

## Lista de parâmetros

### Parâmetro: `Valor1`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Parâmetro: `Valor2`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** O resultado da operação de Equivalência (XNOR) de Valor1 e Valor2

## Tabela-verdade

| Valor1 | Valor2 | Retorno |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

## Exemplo

```
$
// Uma operação de Equivalência entre os dois primeiros
s[3] = Logic.Xnor(s[1], s[2]);
```

## Veja também

`Logic.Xnor3()`, `Logic.Xor()`, `Logic.And()`, `Logic.Or()`, `Logic.Not()`

MÉTODO

# Método: Logic.Xnor3()

## Sintaxe

```
Logic.Xnor3(Valor1, Valor2, Valor3)
```

## Descrição

Considera o valor dos parâmetros "Valor1", "Valor2" e "Valor3" como valores booleanos e executa uma operação de Equivalência (XNOR) entre esses três valores.

Alteração em relação ao DestinyPatch 1.0: se aqui for informada uma String como parâmetro, uma string vazia é considerada falsa (false) e uma string não vazia é considerada verdadeira (true). Assim, uma string que contenha "0" também é considerada verdadeira. Se isso não for desejado, a string deveria ser convertida antes para um número. Isso pode ser feito com o objeto `Convert`.

## Lista de parâmetros

### Parâmetro: `Valor1`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Parâmetro: `Valor2`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Parâmetro: `Valor3`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** O resultado da operação de Equivalência (XNOR) de Valor1, Valor2 e Valor3

## Tabela-verdade

| Valor1 | Valor2 | Valor3 | Retorno |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

## Exemplo

```
$
// Uma operação de Equivalência entre os três primeiros
s[4] = Logic.Xnor3(s[1], s[2], s[3]);
```

## Veja também

`Logic.Xnor()`, `Logic.Xor3()`, `Logic.And()`, `Logic.Or()`, `Logic.Not()`

MÉTODO

# Método: Logic.Xor()

## Sintaxe

```
Logic.Xor(Valor1, Valor2)
```

## Descrição

Considera o valor dos parâmetros "Valor1" e "Valor2" como valores booleanos e executa uma operação OU-EXCLUSIVO (também chamada de Antivalência) entre esses dois valores.

Alteração em relação ao DestinyPatch 1.0: se aqui for informada uma String como parâmetro, uma string vazia é considerada falsa (false) e uma string não vazia é considerada verdadeira (true). Assim, uma string que contenha "0" também é considerada verdadeira. Se isso não for desejado, a string deveria ser convertida antes para um número. Isso pode ser feito com o objeto `Convert`.

## Lista de parâmetros

### Parâmetro: `Valor1`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Parâmetro: `Valor2`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** O resultado da operação OU-EXCLUSIVO (também chamada de Antivalência) de Valor1 e Valor2

## Tabela-verdade

| Valor1 | Valor2 | Retorno |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

## Exemplo

```
$
// Uma operação OU-EXCLUSIVO entre os dois primeiros
s[3] = Logic.Xor(s[1], s[2]);
```

## Veja também

`Logic.Xor3()`, `Logic.Xnor()`, `Logic.And()`, `Logic.Or()`, `Logic.Not()`

MÉTODO

# Método: Logic.Xor3()

## Sintaxe

```
Logic.Xor3(Valor1, Valor2, Valor3)
```

## Descrição

Considera o valor dos parâmetros "Valor1", "Valor2" e "Valor3" como valores booleanos e executa uma operação OU-EXCLUSIVO (também chamada de Antivalência) entre esses três valores.

Alteração em relação ao DestinyPatch 1.0: se aqui for informada uma String como parâmetro, uma string vazia é considerada falsa (false) e uma string não vazia é considerada verdadeira (true). Assim, uma string que contenha "0" também é considerada verdadeira. Se isso não for desejado, a string deveria ser convertida antes para um número. Isso pode ser feito com o objeto `Convert`.

## Lista de parâmetros

### Parâmetro: `Valor1`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Parâmetro: `Valor2`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Parâmetro: `Valor3`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** O resultado da operação OU-EXCLUSIVO (também chamada de Antivalência) de Valor1, Valor2 e Valor3

## Tabela-verdade

| Valor1 | Valor2 | Valor3 | Retorno |
|---|---|---|---|
| 0 | 0 | 0 | 0 |

| Valor1 | Valor2 | Valor3 | Retorno |
|---|---|---|---|
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

## Exemplo

```
$
// Uma operação OU-EXCLUSIVO entre os três primeiros
s[4] = Logic.Xor3(s[1], s[2], s[3]);
```

## Veja também

`Logic.Xor()`, `Logic.Xnor3()`, `Logic.And()`, `Logic.Or()`, `Logic.Not()`

CATEGORIA

# Categoria: Comparações

## Descrição

Nesta categoria estão reunidas todas as comparações do objeto `Logic`. Todos os métodos de comparação desta categoria já são suportados pela linguagem de script e, por isso, devem ser considerados obsoletos, já que, ao chamar esses métodos, é executado o mesmo código dos respectivos `operadores de comparação`. O único método útil desta categoria é a ramificação simples por meio de `Logic.If()`.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Above()` | Método | Compara "Valor1" com "Valor2" e devolve true, caso "Valor1" seja maior que "Valor2". |
| `AboveEqual()` | Método | Compara "Valor1" com "Valor2" e devolve true, caso "Valor1" seja maior/igual a "Valor2". |
| `Below()` | Método | Compara "Valor1" com "Valor2" e devolve true, caso "Valor1" seja menor que "Valor2". |
| `BelowEqual()` | Método | Compara "Valor1" com "Valor2" e devolve true, caso "Valor1" seja menor/igual a "Valor2". |
| `Equal()` | Método | Compara "Valor1" com "Valor2" e devolve true, caso "Valor1" seja igual a "Valor2". |
| `If()` | Método | Devolve, dependendo do estado booleano do parâmetro "Expressão", o parâmetro "ValorVerdadeiro" ou o parâmetro "ValorFalso". |
| `Unequal()` | Método | Compara "Valor1" com "Valor2" e devolve true, caso "Valor1" seja diferente de "Valor2". |

MÉTODO

# Método: Logic.Above()

## Sintaxe

```
Logic.Above(Valor1, Valor2)
```

## Descrição

Compara "Valor1" com "Valor2" e devolve true, caso "Valor1" seja maior que "Valor2".

## Propriedades

- **Obsoleto:** sim

## Lista de parâmetros

### Parâmetro: `Valor1`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Parâmetro: `Valor2`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** Devolve true, caso o primeiro valor seja maior que o segundo; caso contrário, false.

## Exemplo

```
$
// Realizar uma comparação "maior que" entre v[1] e v[2]
s[1] = Logic.Above(v[1], v[2]);
```

## Observações adicionais

Este método deve ser considerado obsoleto, já que o mesmo efeito pode ser obtido com o `operador de comparação maior que`.

```
$
// Realizar uma comparação maior que entre v[1] e
s[1] = v[1] > v[2];
```

## Veja também

`Logic.AboveEqual()`, `Logic.Below()`, `Logic.BelowEqual()`, `Logic.Equal()`, `Logic.Unequal()`

MÉTODO

# Método: Logic.AboveEqual()

## Sintaxe

```
Logic.AboveEqual(Valor1, Valor2)
```

## Descrição

Compara "Valor1" com "Valor2" e devolve true, caso "Valor1" seja maior/igual a "Valor2".

## Propriedades

- **Obsoleto:** sim

## Lista de parâmetros

### Parâmetro: `Valor1`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Parâmetro: `Valor2`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** Devolve true, caso o primeiro valor seja maior/igual a o segundo; caso contrário, false.

## Exemplo

```
$
// Realizar uma comparação "maior/igual" entre v[1] e v[2]
s[1] = Logic.AboveEqual(v[1], v[2]);
```

## Observações adicionais

Este método deve ser considerado obsoleto, já que o mesmo efeito pode ser obtido com o `operador de comparação maior/igual`.

```
$
// Realizar uma comparação maior/igual entre v[1] e
s[1] = v[1] >= v[2];
```

## Veja também

`Logic.Above()`, `Logic.Below()`, `Logic.BelowEqual()`, `Logic.Equal()`, `Logic.Unequal()`

MÉTODO

# Método: Logic.Below()

## Sintaxe

```
Logic.Below(Valor1, Valor2)
```

## Descrição

Compara "Valor1" com "Valor2" e devolve true, caso "Valor1" seja menor que "Valor2".

## Propriedades

- **Obsoleto:** sim

## Lista de parâmetros

### Parâmetro: `Valor1`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Parâmetro: `Valor2`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** Devolve true, caso o primeiro valor seja menor que o segundo; caso contrário, false.

## Exemplo

```
$
// Realizar uma comparação "menor que" entre v[1] e v[2]
s[1] = Logic.Below(v[1], v[2]);
```

## Observações adicionais

Este método deve ser considerado obsoleto, já que o mesmo efeito pode ser obtido com o `operador de comparação menor que`.

```
$
// Realizar uma comparação menor que entre v[1] e
s[1] = v[1] < v[2];
```

## Veja também

`Logic.Above()`, `Logic.AboveEqual()`, `Logic.BelowEqual()`, `Logic.Equal()`, `Logic.Unequal()`

MÉTODO

# Método: Logic.BelowEqual()

## Sintaxe

```
Logic.BelowEqual(Valor1, Valor2)
```

## Descrição

Compara "Valor1" com "Valor2" e devolve true, caso "Valor1" seja menor/igual a "Valor2".

## Propriedades

- **Obsoleto:** sim

## Lista de parâmetros

### Parâmetro: `Valor1`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Parâmetro: `Valor2`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** Devolve true, caso o primeiro valor seja menor/igual a o segundo; caso contrário, false.

## Exemplo

```
$
// Realizar uma comparação "menor/igual" entre v[1] e v[2]
s[1] = Logic.BelowEqual(v[1], v[2]);
```

## Observações adicionais

Este método deve ser considerado obsoleto, já que o mesmo efeito pode ser obtido com o `operador de comparação menor/igual`.

```
$
// Realizar uma comparação menor/igual entre v[1] e
s[1] = v[1] <= v[2];
```

## Veja também

`Logic.Above()`, `Logic.AboveEqual()`, `Logic.Below()`, `Logic.Equal()`, `Logic.Unequal()`

MÉTODO

# Método: Logic.Equal()

## Sintaxe

```
Logic.Equal(Valor1, Valor2)
```

## Descrição

Compara "Valor1" com "Valor2" e devolve true, caso "Valor1" seja igual a "Valor2".

Alteração em relação ao DestinyPatch 1.0: agora também é possível comparar Strings e Booleans com isso, já que para isso é executado o mesmo código do respectivo operador de comparação.

## Propriedades

- **Obsoleto:** sim

## Lista de parâmetros

### Parâmetro: `Valor1`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Parâmetro: `Valor2`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** Devolve true, caso o primeiro valor seja igual ao segundo; caso contrário, false.

## Exemplo

```
$
// Realizar uma comparação "igual" entre v[1] e v[2]
s[1] = Logic.Equal(v[1], v[2]);
```

## Observações adicionais

Este método deve ser considerado obsoleto, já que o mesmo efeito pode ser obtido com o `operador de comparação igual`.

```
$
// Realizar uma comparação "igual" por meio do operador
s[1] = v[1] == v[2];
```

## Veja também

`Logic.Above()`, `Logic.AboveEqual()`, `Logic.Below()`, `Logic.BelowEqual()`, `Logic.Unequal()`

MÉTODO
# Método: Logic.If()

## Sintaxe

```
Logic.If(Expressão, ValorVerdadeiro, ValorFalso)
```

## Descrição

Devolve, dependendo do estado booleano do parâmetro "Expressão", o parâmetro "ValorVerdadeiro" ou o parâmetro "ValorFalso". Como parâmetro "Expressão" também pode ser usada uma expressão de comparação (veja o exemplo).

> **Informação:** Os dados dos parâmetros são assumidos junto com o tipo de dado. No entanto, este método não serve para acesso de escrita.

> **Informação:** Como os parâmetros informados são sempre avaliados antes da chamada do método, este método não pode proteger contra erros de cálculo. Para isso, deveria ser usada a `ramificação If`.

## Lista de parâmetros

### Parâmetro: `Expressão`

- **Tipo de dado:** `bool`
- **Opcional:** não

- **Descrição:** Um valor booleano qualquer, que decide se o parâmetro ValorVerdadeiro ou ValorFalso é devolvido.

### Parâmetro: `ValorVerdadeiro`

- **Tipo de dado:** `any`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Parâmetro: `ValorFalso`

- **Tipo de dado:** `any`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Valor de retorno

- **Tipo de dado:** `any`
- **Descrição:** O parâmetro ValorVerdadeiro ou ValorFalso.

## Exemplo

```
$
// Fazer uma comparação entre v[1] e v[2]
a[1] = Logic.If(v[1] > v[2], "Kaffee", "Kuchen");
```

## Observações adicionais

Este método serve apenas para fazer pequenas comparações entre dois valores. Para distinções mais complexas com vários comandos a executar, deveria ser usada a `ramificação If` ou a `ramificação Switch`.

```
$
// Fazer uma comparação entre v[1] e v[2]
if(v[1] > v[2])
  a[1] = "Kaffee";
else
  a[1] = "Kuchen";
endif;
```

MÉTODO

# Método: Logic.Unequal()

## Sintaxe

```
Logic.Unequal(Valor1, Valor2)
```

## Descrição

Compara "Valor1" com "Valor2" e devolve true, caso "Valor1" seja diferente de "Valor2".

Alteração em relação ao DestinyPatch 1.0: agora também é possível comparar Strings e Booleans com isso, já que para isso é executado o mesmo código do respectivo operador de comparação.

## Propriedades

- **Obsoleto:** sim

## Lista de parâmetros

### Parâmetro: `Valor1`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Parâmetro: `Valor2`

- **Tipo de dado:** `scalar`
- **Opcional:** não
- **Descrição:** Um valor qualquer

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** Devolve true, caso o primeiro valor seja diferente do segundo; caso contrário, false.

## Exemplo

```
$
// Realizar uma comparação "diferente" entre v[1] e v
s[1] = Logic.Unequal(v[1], v[2]);
```

## Observações adicionais

Este método deve ser considerado obsoleto, já que o mesmo efeito pode ser obtido com o `operador de comparação diferente`.

```
$
// Realizar uma comparação "diferente" por meio do
s[1] = v[1] != v[2];
```

## Veja também

`Logic.Above()`, `Logic.AboveEqual()`, `Logic.Below()`, `Logic.BelowEqual()`, `Logic.Equal()`

OBJETO

# Objeto: Map

## Sintaxe

```
Map
```

## Descrição

O objeto Map serve para acessar o mapa atual do jogo.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Chipset do mapa` | Categoria | Contém tudo relacionado ao chipset usado pelo mapa. |
| `AuroraSheet` | Objeto | Com o objeto AuroraSheet do objeto Map é possível editar o gráfico atualmente carregado do chipset. |
| `Informações gerais do mapa` | Categoria | Contém todas as informações gerais sobre o mapa. |
| `Conteúdo do mapa` | Categoria | Contém tudo relacionado ao conteúdo do mapa. |

CATEGORIA

# Categoria: Chipset do mapa

## Descrição

Contém tudo relacionado ao chipset usado pelo mapa.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Chipset` | Propriedade | Com a propriedade Chipset é possível ler/gravar a Id do chipset atualmente usado. |

PROPRIEDADE

# Propriedade: Map.Chipset

## Sintaxe

```
Map.Chipset
```

## Descrição

Com a propriedade Chipset é possível ler/gravar a Id do chipset atualmente usado.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

OBJETO

# Objeto: Map.AuroraSheet

## Sintaxe

```
Map.AuroraSheet
```

## Descrição

Com o objeto AuroraSheet do objeto Map é possível editar o gráfico atualmente carregado do chipset. Como o objeto AuroraSheet, sendo padronizado, está disponível como subobjeto em vários objetos, sua descrição foi escrita apenas uma vez e colocada como subordinada ao objeto `Picture[]`. A descrição se encontra aqui: `Picture[].AuroraSheet`.

## Exemplo

```
$
// Os campos individuais do chipset usado
for(v[1] = 0; v[1] < Map.AuroraSheet.Width; v[1]
  for(v[2] = 0; v[2] < Map.AuroraSheet.Height; v[
    for(v[3] = 0; v[3] < 8; v[3]++)
      v[4] = Map.AuroraSheet.Pixel[v[1] + 15 - v[
      Map.AuroraSheet.Pixel[v[1] + 15 - v[3], v[2
      Map.AuroraSheet.Pixel[v[1] + v[3], v[2]] =
    next;
  next;
next;
```

## Veja também

`Map`, `Picture[].AuroraSheet`

CATEGORIA

# Categoria: Informações gerais do mapa

## Descrição

Contém todas as informações gerais sobre o mapa.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Height` | Propriedade | Com a propriedade Height é possível descobrir a altura do mapa. |
| `HeroX` | Propriedade | Com a propriedade HeroX é possível descobrir a coordenada X atual do herói. |

| Nome | Tipo | Resumo |
| --- | --- | --- |
| `HeroY` | Propriedade | Com a propriedade HeroY é possível descobrir a coordenada Y atual do herói. |
| `Id` | Propriedade | Com a propriedade Id é possível descobrir a Map-Id atual. |
| `NumberOfSaves` | Propriedade | Indica quantas vezes o mapa já foi salvo pelo Maker. |
| `ScreenX` | Propriedade | Com a propriedade ScreenX é possível ler/gravar a coordenada X atual da tela no mapa. |
| `ScreenY` | Propriedade | Com a propriedade ScreenY é possível ler/gravar a coordenada Y atual da tela no mapa. |
| `ScrollType` | Propriedade | Com a propriedade ScrollType é possível ler/gravar o comportamento de exibição das bordas do mapa. |
| `Width` | Propriedade | Com a propriedade Width é possível descobrir a largura do mapa. |

PROPRIEDADE

# Propriedade: Map.Height

## Sintaxe

```
Map.Height
```

## Descrição

Com a propriedade Height é possível descobrir a altura do mapa.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Map.HeroX

## Sintaxe

```
Map.HeroX
```

## Descrição

Com a propriedade HeroX é possível descobrir a coordenada X atual do herói.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Volátil:** sim

PROPRIEDADE

## Propriedade: Map.HeroY

### Sintaxe

```
Map.HeroY
```

### Descrição

Com a propriedade HeroY é possível descobrir a coordenada Y atual do herói.

### Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Map.Id

### Sintaxe

```
Map.Id
```

### Descrição

Com a propriedade Id é possível descobrir a Map-Id atual. Ela começa em 1 e é atribuída pelo RPG Maker.

### Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Map.NumberOfSaves

### Sintaxe

```
Map.NumberOfSaves
```

### Descrição

Indica quantas vezes o mapa já foi salvo pelo Maker.

### Propriedades

- **Tipo de dado:** `dword`

- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Map.ScreenX

### Sintaxe

```
Map.ScreenX
```

### Descrição

Com a propriedade ScreenX é possível ler/gravar a coordenada X atual da tela no mapa. As coordenadas se referem ao canto superior esquerdo do mapa (veja a figura):

Nesta figura, a tela teria, por exemplo, o valor 48.

### Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

### Exemplo

```
$
// Deslocar a posição atual da tela em 1 campo para
Map.ScreenX += 16;
```

### Veja também

`Map`, `Map.ScreenY`

PROPRIEDADE

## Propriedade: Map.ScreenY

### Sintaxe

```
Map.ScreenY
```

### Descrição

Com a propriedade ScreenY é possível ler/gravar a coordenada Y atual da tela no mapa. As coordenadas se referem ao canto superior esquerdo do mapa (veja a figura):

Nesta figura, a tela teria, por exemplo, o valor 80.

### Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

## Exemplo

```
$
// Deslocar a posição atual da tela em 1 campo para
Map.ScreenY -= 16;
```

## Veja também

`Map`, `Map.ScreenX`

PROPRIEDADE

# Propriedade: Map.ScrollType

## Sintaxe

```
Map.ScrollType
```

## Descrição

Com a propriedade ScrollType é possível ler/gravar o comportamento de exibição das bordas do mapa. Para isso são usadas as constantes `ScrollType`:

`SCROLLTYPE_NORMAL`: o mapa é exibido normalmente; ao chegar na borda do mapa, não se avança mais. `SCROLLTYPE_VERTICAL`: o mapa é exibido normalmente nas bordas horizontais, mas as bordas verticais formam um laço. `SCROLLTYPE_HORIZONTAL`: o mapa é exibido normalmente nas bordas verticais, mas as bordas horizontais formam um laço. `SCROLLTYPE_BOTH`: o mapa é exibido como um laço em todas as bordas.

## Propriedades

- **Tipo de dado:** `byte`
- **Somente leitura:** não
- **Faixa de valores:** 0 ... 3
- **Volátil:** sim

## Veja também

`Map`

PROPRIEDADE

# Propriedade: Map.Width

## Sintaxe

```
Map.Width
```

## Descrição

Com a propriedade Width é possível descobrir a largura do mapa.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

CATEGORIA

# Categoria: Conteúdo do mapa

## Descrição

Contém tudo relacionado ao conteúdo do mapa.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `EventCount` | Propriedade | Com a propriedade EventCount é possível descobrir a quantidade de eventos no mapa. |
| `Lower[]` | Propriedade | Com a propriedade Lower é possível ler/gravar o Lower-Chip em uma determinada posição do mapa. |
| `Upper[]` | Propriedade | Com a propriedade Upper é possível ler/gravar o Upper-Chip em uma determinada posição do mapa. |

PROPRIEDADE

# Propriedade: Map.EventCount

## Sintaxe

```
Map.EventCount
```

## Descrição

Com a propriedade EventCount é possível descobrir a quantidade de eventos no mapa. Essa propriedade pode ser usada junto com o objeto `MapEvent`, para executar um laço sobre todos os eventos do mapa. Na propriedade EventCount, os eventos especiais, como o herói ou os veículos, não estão incluídos.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Map.Lower[]

## Sintaxe

```
Map.Lower[X, Y]
```

## Descrição

Com a propriedade Lower é possível ler/gravar o Lower-Chip em uma determinada posição do mapa. O `cálculo dos chips inferiores/superiores` pode ser consultado no apêndice.

## Propriedades

- **Tipo de dado:** `word`
- **Somente leitura:** não
- **Volátil:** sim

## Lista de parâmetros

### Parâmetro: `X`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... (Map.Width - 1)
- **Descrição:** Coordenada X na faixa de 0 ... (`Map.Width` - 1)

### Parâmetro: `Y`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... (Map.Height - 1)
- **Descrição:** Coordenada Y na faixa de 0 ... (`Map.Height` - 1)

## Erros possíveis

`DESTINYERROR_ARRAYBOUND`

PROPRIEDADE

# Propriedade: Map.Upper[]

## Sintaxe

```
Map.Upper[X, Y]
```

## Descrição

Com a propriedade Upper é possível ler/gravar o Upper-Chip em uma determinada posição do mapa. O `cálculo dos chips inferiores/superiores` pode ser consultado no apêndice.

## Propriedades

- **Tipo de dado:** `word`
- **Somente leitura:** não
- **Volátil:** sim

## Lista de parâmetros

### Parâmetro: `X`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... (Map.Width - 1)
- **Descrição:** Coordenada X na faixa de 0 ... (`Map.Width` - 1)

### Parâmetro: `Y`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... (Map.Height - 1)
- **Descrição:** Coordenada Y na faixa de 0 ... (`Map.Height` - 1)

## Erros possíveis

`DESTINYERROR_ARRAYBOUND`

OBJETO

# Objeto: MapEvent

## Sintaxe

```
MapEvent
```

## Descrição

O objeto MapEvent serve para tornar acessíveis os eventos do mapa. Diferente do objeto `Event`, aqui é usado um índice contínuo baseado em zero para acessar um evento específico do mapa. Como todas as propriedades e métodos do objeto MapEvent são idênticos aos do objeto `Event`, a descrição deles deve ser consultada lá.

## Exemplo

```
$
// Um laço sobre todos os eventos do mapa (exceto o
// e inverter o status Exists
for(v[1] = 0; v[1] < MapEvent.Count; v[1]++)
  MapEvent[v[1]].Exists = !MapEvent[v[1]].Exists;
next;
```

## Veja também

`Event`, `Event[]`

OBJETO

# Objeto: Math

## Sintaxe

```
Math
```

## Descrição

Com o objeto Math é possível realizar diversas operações matemáticas (trigonometria, logaritmação, radiciação, arredondamento...).

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Constantes` | Categoria | Contém diversas constantes matemáticas. |
| `Funções de potência` | Categoria | Contém funções que trabalham com potências. |
| `Funções de arredondamento` | Categoria | Contém funções para arredondamento. |
| `Outras funções` | Categoria | Contém todas as demais funções matemáticas. |
| `Funções de ângulo` | Categoria | Contém funções de ângulo. |
| `Funções hiperbólicas inversas` | Categoria | Contém funções hiperbólicas inversas (funções de área). |
| `Funções hiperbólicas` | Categoria | Contém funções hiperbólicas. |
| `Funções trigonométricas inversas` | Categoria | Contém funções arco-trigonométricas (funções trigonométricas inversas). |
| `Funções trigonométricas` | Categoria | Contém funções trigonométricas. |
| `Números aleatórios` | Categoria | Contém funções para gerar números aleatórios. |

CATEGORIA

# Categoria: Constantes

## Descrição

Contém diversas constantes matemáticas.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `E` | Propriedade | Esta propriedade representa o número de Euler (e = 2,718281828...). |
| `Pi` | Propriedade | Esta propriedade representa o número de Ludolph, ou seja, Pi (π = 3,141592653589...). |

PROPRIEDADE

# Propriedade: Math.E

## Sintaxe

```
Math.E
```

## Descrição

Esta propriedade representa o número de Euler (e = 2,718281828...).

## Propriedades

- **Tipo de dado:** `float`
- **Somente leitura:** sim

## Exemplo

```
$
f[1] = Math.E;     // f[1] = 2.718281828...
```

## Erros possíveis

`DESTINYERROR_FLOATERROR`

PROPRIEDADE

# Propriedade: Math.Pi

## Sintaxe

```
Math.Pi
```

## Descrição

Esta propriedade representa o número de Ludolph, ou seja, Pi (π = 3,141592653589...).

## Propriedades

- **Tipo de dado:** `float`
- **Somente leitura:** sim

## Exemplo

```
$
f[1] = Math.Pi;    // f[1] = 3.141592653589...
```

## Erros possíveis

`DESTINYERROR_FLOATERROR`

CATEGORIA

# Categoria: Funções de potência

## Descrição

Contém funções que trabalham com potências.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Exp()` | Método | Calcula a potência de 10 elevado ao expoente. |
| `Lb()` | Método | Calcula o logaritmo binário (base 2). |
| `Lg()` | Método | Calcula o logaritmo decimal (base 10). |
| `Ln()` | Método | Calcula o logaritmo natural (a base é o número de Euler). |
| `Log()` | Método | Calcula o logaritmo em uma base qualquer. |
| `Power()` | Método | Eleva uma base a um expoente. |
| `Sqrt()` | Método | Calcula a raiz (quadrada) de um número (radiciação). |

MÉTODO

# Método: Math.Exp()

## Sintaxe

```
Math.Exp(Expoente)
```

## Descrição

Calcula a potência de 10 elevado ao expoente.

## Lista de parâmetros

### Parâmetro: `Expoente`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** Indica o expoente na base 10.

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** O resultado de 10 elevado ao expoente

## Exemplo

```
$
// Elevar o número 10 à sexta potência
f[1] = Math.Exp(6);     // = 1000000
```

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.Power()`, `Math.Log()`, `Math.Lg()`, `Math.Ln()`, `Math.Lb()`, `Math.Sqrt()`

MÉTODO

# Método: Math.Lb()

## Sintaxe

```
Math.Lb(Número)
```

## Descrição

Calcula o logaritmo binário (base 2).

## Lista de parâmetros

### Parâmetro: `Número`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Faixa de valores:** > 0
- **Descrição:** Indica o número do qual o logaritmo deve ser calculado. Esse número precisa ser maior que 0, pois o DestinyPatch não consegue calcular com números complexos.

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** O expoente na base 2 necessário para formar o número.

## Exemplo

```
$
// Determinar o logaritmo do número 1024 na base 2
f[1] = Math.Lb(1024);       // = 10
```

## Erros possíveis

`DESTINYERROR_FLOATERROR`, `DESTINYERROR_VALUE`

## Veja também

`Math.Power()`, `Math.Exp()`, `Math.Log()`, `Math.Lg()`, `Math.Ln()`, `Math.Sqrt()`

MÉTODO

# Método: Math.Lg()

## Sintaxe

```
Math.Lg(Número)
```

## Descrição

Calcula o logaritmo decimal (base 10).

## Lista de parâmetros

### Parâmetro: `Número`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Faixa de valores:** > 0
- **Descrição:** Indica o número do qual o logaritmo deve ser calculado. Esse número precisa ser maior que 0, pois o DestinyPatch não consegue calcular com números complexos.

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** O expoente na base 10 necessário para formar o número.

## Exemplo

```
$
// Determinar o logaritmo do número 10000 na base 10
f[1] = Math.Lg(10000.0);         // = 4 (pois 10
```

## Erros possíveis

`DESTINYERROR_FLOATERROR`, `DESTINYERROR_VALUE`

## Veja também

`Math.Power()`, `Math.Exp()`, `Math.Log()`, `Math.Lb()`, `Math.Ln()`, `Math.Sqrt()`

MÉTODO

# Método: Math.Ln()

## Sintaxe

```
Math.Ln(Número)
```

## Descrição

Calcula o logaritmo natural (a base é o `número de Euler`).

## Lista de parâmetros

### Parâmetro: `Número`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Faixa de valores:** > 0
- **Descrição:** Indica o número do qual o logaritmo deve ser calculado. Esse número precisa ser maior que 0, pois o DestinyPatch não consegue calcular com números complexos.

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** O expoente na base e necessário para formar o número.

## Exemplo

```
$
// Determinar o logaritmo de 1 / e na base e
f[1] = Math.Ln(1 / Math.E);      // = -1
```

## Erros possíveis

`DESTINYERROR_FLOATERROR`, `DESTINYERROR_VALUE`

## Veja também

`Math.Power()`, `Math.Exp()`, `Math.Log()`, `Math.Lg()`, `Math.Lb()`, `Math.Sqrt()`

MÉTODO

# Método: Math.Log()

## Sintaxe

```
Math.Log(Número, Base)
```

## Descrição

Calcula o logaritmo em uma base qualquer.

## Lista de parâmetros

### Parâmetro: `Número`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Faixa de valores:** > 0
- **Descrição:** Indica o número do qual o logaritmo deve ser calculado. Esse número precisa ser maior que 0, pois o DestinyPatch não consegue calcular com números complexos.

### Parâmetro: `Base`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Faixa de valores:** > 0
- **Descrição:** Indica a base à qual o logaritmo deve se referir. Esse número precisa ser maior que 0, pois o DestinyPatch não consegue calcular com números complexos.

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** O expoente na base indicada necessário para formar o número.

## Exemplo

```
$
// Determinar o logaritmo do número 25 na base 5
f[1] = Math.Log(25.0, 5.0);         // = 2 (pois
```

## Observações adicionais

Para calcular o logaritmo na base 10, 2 ou e, devem ser usados os métodos `Math.Lg()`, `Math.Lb()` e `Math.Ln()`, pois eles são mais rápidos por exigirem menos parâmetros.

## Erros possíveis

`DESTINYERROR_FLOATERROR`, `DESTINYERROR_VALUE`

## Veja também

`Math.Power()`, `Math.Exp()`, `Math.Lg()`, `Math.Lb()`, `Math.Ln()`, `Math.Sqrt()`

MÉTODO

# Método: Math.Power()

## Sintaxe

```
Math.Power(Base, Expoente)
```

## Descrição

Eleva uma base a um expoente.

## Lista de parâmetros

### Parâmetro: `Base`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** Indica a base do cálculo.

### Parâmetro: `Expoente`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Faixa de valores:** com 0 como base > 0; com números negativos como base, qualquer número sem parte decimal
- **Descrição:** Indica o expoente do cálculo. A faixa de valores permitida para este parâmetro depende da base usada:

Com base positiva, o expoente precisa ser maior que -∞ e menor que +∞ (ou seja, qualquer número é permitido). Com 0 como base, o expoente precisa ser maior que 0; o resultado então é sempre 0 (portanto, 0 elevado a 0, como de costume na matemática, não é permitido!). Com base negativa, o expoente não pode ter parte decimal, pois, caso contrário, o resultado estaria no espaço dos números complexos.

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** O resultado da base elevada ao expoente

## Exemplo

```
$
// Elevar o número 3 à quarta potência
f[1] = Math.Power(3.0, 4.0);        // = 81 (= 3
// Calcular a raiz cúbica de 8
f[2] = Math.Power(8.0, 1.0 / 3.0);  // = 2 (pois
```

## Observações adicionais

Se o expoente for uma fração, é calculada uma raiz. Por exemplo, 0,5 para a raiz quadrada, 0,125 para a raiz oitava etc. O índice da raiz calculada corresponde ao denominador do expoente, quando o numerador é igual a 1. Por

simplicidade, também é possível informar o parâmetro opcional em `Math.Sqrt()`. Nesse caso, o inverso é calculado automaticamente, o que facilita indicar qual raiz se deseja obter.

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.Exp()`, `Math.Log()`, `Math.Lg()`, `Math.Lb()`, `Math.Ln()`, `Math.Sqrt()`

MÉTODO

# Método: Math.Sqrt()

## Sintaxe

```
Math.Sqrt(Zahl, [Expoente])
```

## Descrição

Calcula a raiz (quadrada) de um número (radiciação).

## Lista de parâmetros

### Parâmetro: `Número`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** Indica o número do qual a raiz deve ser calculada.

### Parâmetro: `Expoente`

- **Tipo de dado:** `float`
- **Opcional:** sim
- **Valor padrão:** 2.0
- **Faixa de valores:** != 0
- **Descrição:** Indica o índice da raiz a ser calculada. O valor padrão é 2, por isso normalmente é calculada a raiz quadrada. Esse valor não pode ser 0, pois qualquer número (exceto 0) elevado a 0 é igual a 1. A faixa de valores permitida para este parâmetro também depende do número do qual a raiz é extraída:

Com número positivo, o expoente precisa ser maior que -∞ e menor que +∞ (ou seja, qualquer número é permitido). Com 0 como número, o expoente precisa ser maior que 0; o resultado então é sempre 0 (portanto, 0 elevado a 0, como de costume na matemática, não é permitido!). Com número negativo, o inverso do expoente não pode ter parte decimal, pois, caso contrário, o resultado estaria no espaço dos números complexos.

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** A base do expoente necessária para formar o número.

## Exemplo

```
$
// Calcular a raiz quadrada de 1000000
f[1] = Math.Sqrt(1000000.0);       // = 1000.0
// Calcular a raiz cúbica de 1000000
f[2] = Math.Sqrt(1000000.0, 3.0);  // = 100.0
```

## Observações adicionais

Se o parâmetro opcional "Expoente" for informado, é calculado o seu inverso, e os parâmetros são repassados ao método `Math.Power()`.

## Erros possíveis

`DESTINYERROR_FLOATERROR`, `DESTINYERROR_VALUE`

## Veja também

`Math.Power()`, `Math.Exp()`, `Math.Log()`, `Math.Lg()`, `Math.Ln()`, `Math.Lb()`

CATEGORIA

# Categoria: Funções de arredondamento

## Descrição

Contém funções para arredondamento.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Decimal()` | Método | Remove a parte inteira de um número. |
| `Integer()` | Método | Remove a parte decimal de um número. |
| `Round()` | Método | Arredonda um número de forma comercial (ou seja, de 5 a 9 arredonda para cima) até a casa indicada. |
| `RoundDown()` | Método | Arredonda um número em direção a -∞ (ou seja, de 1 a 9 arredonda para baixo) até a casa indicada. |
| `RoundUp()` | Método | Arredonda um número em direção a +∞ (ou seja, de 1 a 9 arredonda para cima) até a casa indicada. |

MÉTODO

# Método: Math.Decimal()

## Sintaxe

```
Math.Decimal(Número)
```

## Descrição

Remove a parte inteira de um número.

## Sinônimos

Scale() Dec()

## Lista de parâmetros

### Parâmetro: `Número`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** O número do qual a parte inteira deve ser removida

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** A parte decimal do número indicado

## Exemplo

```
$
// Remover a parte inteira de vários números
f[1] = Math.Decimal( 1273.48);     // =  0.48
f[2] = Math.Decimal(-1273.48);     // = -0.48
```

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.Integer()`, `Math.Round()`, `Math.RoundUp()`, `Math.RoundDown()`

MÉTODO

# Método: Math.Integer()

## Sintaxe

```
Math.Integer(Número)
```

## Descrição

Remove a parte decimal de um número.

## Sinônimos

Int()

## Lista de parâmetros

### Parâmetro: `Número`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** O número do qual a parte decimal deve ser removida

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** A parte inteira do número indicado

## Exemplo

```
$
// Remover a parte decimal de vários números
f[1] = Math.Integer( 1273.48);     // =  1273.0
f[2] = Math.Integer(-1273.48);     // = -1273.0
```

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.Decimal()`, `Math.Round()`, `Math.RoundUp()`, `Math.RoundDown()`

MÉTODO

# Método: Math.Round()

## Sintaxe

```
Math.Round(Zahl, [Posição])
```

## Descrição

Arredonda um número de forma comercial (ou seja, de 5 a 9 arredonda para cima) até a casa indicada.

## Lista de parâmetros

### Parâmetro: `Número`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** O número a arredondar

### Parâmetro: `Posição`

- **Tipo de dado:** `float`

- **Opcional:** sim
- **Valor padrão:** 0.0
- **Descrição:** Indica a distância da casa (em relação à vírgula) até a qual se arredonda.

Valores < 0 indicam que o arredondamento é feito na respectiva casa antes da vírgula (por exemplo, -1 para a segunda casa antes da vírgula). O valor 0 indica que a parte decimal é arredondada para a parte inteira (0 para a primeira casa antes da vírgula). Valores > 0 indicam que o arredondamento é feito na respectiva casa depois da vírgula (por exemplo, 1 para a primeira casa depois da vírgula).

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** O número arredondado de forma comercial

## Exemplo

```
$
// Arredondar vários números
f[1] = Math.Round(1273.48);     // = 1273.0 (para baixo
f[2] = Math.Round(1273.58);     // = 1274.0 (para cima
f[3] = Math.Round(1273.48, -1); // = 1270.0 (para baixo
f[4] = Math.Round(1273.48, -2); // = 1300.0 (para cima
f[5] = Math.Round(1273.48, 1);  // = 1273.5 (para cima
```

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.RoundUp()`, `Math.RoundDown()`, `Math.Integer()`, `Math.Decimal()`

MÉTODO

# Método: Math.RoundDown()

## Sintaxe

```
Math.RoundDown(Zahl, [Posição])
```

## Descrição

Arredonda um número em direção a -∞ (ou seja, de 1 a 9 arredonda para baixo) até a casa indicada. Números negativos, nesse caso, acabam ficando mais negativos (ou permanecem iguais).

## Sinônimos

Floor()

## Lista de parâmetros

### Parâmetro: `Número`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** O número a arredondar

### Parâmetro: `Posição`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 0
- **Descrição:** Indica a distância da casa (em relação à vírgula) até a qual se arredonda.

Valores < 0 indicam que o arredondamento é feito na respectiva casa antes da vírgula (por exemplo, -1 para a segunda casa antes da vírgula). O valor 0 indica que a parte decimal é arredondada para a parte inteira (0 para a primeira casa antes da vírgula). Valores > 0 indicam que o arredondamento é feito na respectiva casa depois da vírgula (por exemplo, 1 para a primeira casa depois da vírgula).

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** O número arredondado para baixo

## Exemplo

```
$
// Arredondar vários números para baixo
f[1] = Math.RoundDown( 1273.48);     // =  1273.0
f[2] = Math.RoundDown(-1273.48);     // = -1274.0
f[3] = Math.RoundDown( 1273.48, -1); // =  1270.0
f[4] = Math.RoundDown( 1273.48, -2); // =  1200.0
f[5] = Math.RoundDown( 1273.48,  1); // =  1273.4
```

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.RoundUp()`, `Math.Round()`, `Math.Integer()`, `Math.Decimal()`

MÉTODO

# Método: Math.RoundUp()

## Sintaxe

```
Math.RoundUp(Zahl, [Posição])
```

## Descrição

Arredonda um número em direção a +∞ (ou seja, de 1 a 9 arredonda para cima) até a casa indicada. Números negativos, nesse caso, acabam ficando mais positivos (ou permanecem iguais).

## Sinônimos

Ceil()

## Lista de parâmetros

### Parâmetro: `Número`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** O número a arredondar

### Parâmetro: `Posição`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 0
- **Descrição:** Indica a distância da casa (em relação à vírgula) até a qual se arredonda.

Valores < 0 indicam que o arredondamento é feito na respectiva casa antes da vírgula (por exemplo, -1 para a segunda casa antes da vírgula). O valor 0 indica que a parte decimal é arredondada para a parte inteira (0 para a primeira casa antes da vírgula). Valores > 0 indicam que o arredondamento é feito na respectiva casa depois da vírgula (por exemplo, 1 para a primeira casa depois da vírgula).

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** O número arredondado para cima

## Exemplo

```
$
// Arredondar vários números para cima
f[1] = Math.RoundUp( 1273.48);     // =  1274.0
f[2] = Math.RoundUp(-1273.48);     // = -1273.0
f[3] = Math.RoundUp( 1273.48, -1); // =  1280.0 (
f[4] = Math.RoundUp( 1273.48, -2); // =  1300.0 (
f[5] = Math.RoundUp( 1273.48,  1); // =  1273.5 (
```

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.RoundDown()`, `Math.Round()`, `Math.Integer()`, `Math.Decimal()`

CATEGORIA

# Categoria: Outras funções

## Descrição

Contém todas as demais funções matemáticas.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Abs()` | Método | Devolve o valor absoluto de um número (ou seja, o sinal do número é sempre definido como positivo). |
| `Cmp()` | Método | Compara dois números entre si. |

MÉTODO

# Método: Math.Abs()

## Sintaxe

```
Math.Abs(Número)
```

## Descrição

Devolve o valor absoluto de um número (ou seja, o sinal do número é sempre definido como positivo).

## Lista de parâmetros

### Parâmetro: `Número`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** O número a converter

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** O valor absoluto do número

## Exemplo

```
$
// Obter vários valores absolutos
f[1] = Math.Abs(-2);   // = 2
f[2] = Math.Abs(-1);   // = 1
f[3] = Math.Abs(0);    // = 0
f[4] = Math.Abs(1);    // = 1
f[5] = Math.Abs(2);    // = 2
```

## Erros possíveis

`DESTINYERROR_FLOATERROR`

MÉTODO

# Método: Math.Cmp()

## Sintaxe

```
Math.Cmp(Número1, Número2)
```

## Descrição

Compara dois números entre si. O resultado dessa função pode ser bem usado no RPG Maker para comparar, por meio de uma ramificação, o valor de uma variável com 0. Para comparações dentro de um DestinyScript, este método deve ser visto como algo rudimentar, já que na versão 1.0 do patch não havia operadores de comparação disponíveis.

## Lista de parâmetros

### Parâmetro: `Número1`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** O primeiro número a comparar

### Parâmetro: `Número2`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** O segundo número a comparar

### Valor de retorno

- **Tipo de dado:** `dword`
- **Descrição:** Resulta em -1 se Número1 < Número2. Resulta em 0 se Número1 = Número2. Resulta em 1 se Número1 > Número2.

## Exemplo

```
$
// Comparar o número 10.0 com o número 11.0 e
v[1] = Math.Cmp(10.0, 11.0);  // = -1
```

## Observações adicionais

Para comparações dentro de um DestinyScript, devem ser usados os operadores de comparação:

```
$
if(f[1] < f[2])
  // ...
```

```
elseif(f[1] == f[2])
  // ...
else // if(f[1] > f[2])
  // ...
endif;
```

## Erros possíveis

`DESTINYERROR_FLOATERROR`

CATEGORIA

# Categoria: Funções de ângulo

## Descrição

Contém funções de ângulo.

CATEGORIA

# Categoria: Funções hiperbólicas inversas

## Descrição

Contém funções hiperbólicas inversas (funções de área). Elas são as funções inversas das `funções hiperbólicas`.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `ACosH()` | Método | Calcula o cosseno hiperbólico inverso de um valor. |
| `ACotH()` | Método | Calcula a cotangente hiperbólica inversa de um valor. |
| `ACscH()` | Método | Calcula a cossecante hiperbólica inversa de um valor. |
| `ASecH()` | Método | Calcula a secante hiperbólica inversa de um valor. |
| `ASinH()` | Método | Calcula o seno hiperbólico inverso de um valor. |
| `ATanH()` | Método | Calcula a tangente hiperbólica inversa de um valor. |

MÉTODO

# Método: Math.ACosH()

## Sintaxe

```
Math.ACosH(CosH)
```

## Descrição

Calcula o cosseno hiperbólico inverso de um valor.

## Lista de parâmetros

### Parâmetro: `CosH`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** Indica o cosseno hiperbólico a converter.

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** O cosseno hiperbólico inverso do valor indicado

## Exemplo

Chamar o método:

```
$
// Calcular o cosseno hiperbólico inverso de 10.067661995
f[1] = Math.ACosH(10.0676619957777);   // = 3
```

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.CosH()`

MÉTODO

# Método: Math.ACotH()

## Sintaxe

```
Math.ACotH(CotH)
```

## Descrição

Calcula a cotangente hiperbólica inversa de um valor.

## Lista de parâmetros

### Parâmetro: `CotH`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** Indica a cotangente hiperbólica a converter.

### Valor de retorno

- **Tipo de dado:** `float`

- **Descrição:** A cotangente hiperbólica inversa do valor indicado

## Exemplo

Chamar o método:

```
$
// Calcular a cotangente hiperbólica inversa de 1.00496982
f[1] = Math.ACotH(1.00496982331368);   // = 3
```

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.CotH()`

MÉTODO

# Método: Math.ACscH()

## Sintaxe

```
Math.ACscH(CscH)
```

## Descrição

Calcula a cossecante hiperbólica inversa de um valor.

## Lista de parâmetros

### Parâmetro: `CscH`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** Indica a cossecante hiperbólica a converter.

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** A cossecante hiperbólica inversa do valor indicado

## Exemplo

Chamar o método:

```
$
// Calcular a cossecante hiperbólica inversa de 0.099821569
f[1] = Math.ACscH(0.0998215696688227);   // = 3
```

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.CscH()`

MÉTODO

# Método: Math.ASecH()

## Sintaxe

```
Math.ASecH(SecH)
```

## Descrição

Calcula a secante hiperbólica inversa de um valor.

## Lista de parâmetros

### Parâmetro: `SecH`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** Indica a secante hiperbólica a converter.

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** A secante hiperbólica inversa do valor indicado

## Exemplo

Chamar o método:

```
$
// Calcular a secante hiperbólica inversa de 0.09932792741
f[1] = Math.ASecH(0.0993279274194332);   // = 3
```

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.SecH()`

MÉTODO

# Método: Math.ASinH()

## Sintaxe

```
Math.ASinH(SinH)
```

## Descrição

Calcula o seno hiperbólico inverso de um valor.

## Lista de parâmetros

### Parâmetro: `SinH`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** Indica o seno hiperbólico a converter.

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** O seno hiperbólico inverso do valor indicado

## Exemplo

Chamar o método:

```
$
// Calcular o seno hiperbólico inverso de 10.01787492740
f[1] = Math.ASinH(10.0178749274099);   // = 3
```

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.SinH()`

MÉTODO

# Método: Math.ATanH()

## Sintaxe

```
Math.ATanH(TanH)
```

## Descrição

Calcula a tangente hiperbólica inversa de um valor.

## Lista de parâmetros

### Parâmetro: `TanH`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** Indica a tangente hiperbólica a converter.

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** A tangente hiperbólica inversa do valor indicado

## Exemplo

Chamar o método:

```
$
// Calcular a tangente hiperbólica inversa de 0.9950547536
f[1] = Math.ATanH(0.99505475368673);   // = 3
```

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.TanH()`

CATEGORIA

# Categoria: Funções hiperbólicas

## Descrição

Contém funções hiperbólicas.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `CosH()` | Método | Calcula o cosseno hiperbólico de um valor. |
| `CotH()` | Método | Calcula a cotangente hiperbólica de um valor. |
| `CscH()` | Método | Calcula a cossecante hiperbólica de um valor. |
| `SecH()` | Método | Calcula a secante hiperbólica de um valor. |
| `SinH()` | Método | Calcula o seno hiperbólico de um valor. |
| `TanH()` | Método | Calcula a tangente hiperbólica de um valor. |

MÉTODO

# Método: Math.CosH()

## Sintaxe

```
Math.CosH(Valor)
```

## Descrição

Calcula o cosseno hiperbólico de um valor.

## Lista de parâmetros

### Parâmetro: `Valor`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** Indica o valor a converter.

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** O cosseno hiperbólico do valor indicado

## Exemplo

Chamar o método:

```
$
// Calcular o cosseno hiperbólico de 3
f[1] = Math.CosH(3.0);   // = 10.0676619957777
```

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.ACosH()`

MÉTODO

# Método: Math.CotH()

## Sintaxe

```
Math.CotH(Valor)
```

## Descrição

Calcula a cotangente hiperbólica de um valor.

## Lista de parâmetros

### Parâmetro: `Valor`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** Indica o valor a converter.

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** A cotangente hiperbólica do valor indicado

## Exemplo

Chamar o método:

```
$
// Calcular a cotangente hiperbólica de 3
f[1] = Math.CotH(3.0);   // = 1.00496982331368
```

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.ACotH()`

MÉTODO

# Método: Math.CscH()

## Sintaxe

```
Math.CscH(Valor)
```

## Descrição

Calcula a cossecante hiperbólica de um valor.

## Lista de parâmetros

### Parâmetro: `Valor`

- **Tipo de dado:** `float`
- **Opcional:** não

- **Descrição:** Indica o valor a converter.

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** A cossecante hiperbólica do valor indicado

## Exemplo

Chamar o método:

```
$
// Calcular a cossecante hiperbólica de 3
f[1] = Math.CscH(3.0);   // = 0.0998215696688227
```

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.ACscH()`

MÉTODO

# Método: Math.SecH()

## Sintaxe

```
Math.SecH(Valor)
```

## Descrição

Calcula a secante hiperbólica de um valor.

## Lista de parâmetros

### Parâmetro: `Valor`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** Indica o valor a converter.

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** A secante hiperbólica do valor indicado

## Exemplo

Chamar o método:

```
$
// Calcular a secante hiperbólica de 3
f[1] = Math.SecH(3.0);   // = 0.0993279274194332
```

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.ASecH()`

MÉTODO

# Método: Math.SinH()

## Sintaxe

```
Math.SinH(Valor)
```

## Descrição

Calcula o seno hiperbólico de um valor.

## Lista de parâmetros

### Parâmetro: `Valor`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** Indica o valor a converter.

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** O seno hiperbólico do valor indicado

## Exemplo

Chamar o método:

```
$
// Calcular o seno hiperbólico de 3
f[1] = Math.SinH(3.0);   // = 10.0178749274099
```

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.ASinH()`

MÉTODO

# Método: Math.TanH()

## Sintaxe

```
Math.TanH(Valor)
```

## Descrição

Calcula a tangente hiperbólica de um valor.

## Lista de parâmetros

### Parâmetro: `Valor`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** Indica o valor a converter.

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** A tangente hiperbólica do valor indicado

## Exemplo

Chamar o método:

```
$
// Calcular a tangente hiperbólica de 3
f[1] = Math.TanH(3.0);   // = 0.99505475368673
```

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.ATanH()`

CATEGORIA

# Categoria: Funções trigonométricas inversas

## Descrição

Contém funções arco-trigonométricas (funções trigonométricas inversas).

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `ACos()` | Método | Calcula o ângulo a partir do cosseno (= arcocosseno). |
| `ACot()` | Método | Calcula o ângulo a partir da cotangente (= arcocotangente). |
| `ACsc()` | Método | Calcula o ângulo a partir da cossecante (= arcocossecante). |
| `ASec()` | Método | Calcula o ângulo a partir da secante (= arcosecante). |
| `ASin()` | Método | Calcula o ângulo a partir do seno (= arcoseno). |
| `ATan()` | Método | Calcula o ângulo a partir da tangente (= arcotangente). |

MÉTODO

# Método: Math.ACos()

## Sintaxe

```
Math.ACos(Cosinus, [Formato])
```

## Descrição

Calcula o ângulo a partir do cosseno (= arcocosseno).

## Lista de parâmetros

### Parâmetro: `Cosseno`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** Indica o cosseno a converter.

### Parâmetro: `Formato`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 2 (= RAD)
- **Faixa de valores:** 1 ... 4
- **Descrição:** Indica o formato em que o ângulo é devolvido. Para isso são usadas as `constantes de formato de ângulo`. Os seguintes formatos estão definidos:

1. `DEG`: significa Degree (grau) e representa o ângulo de 0 a 360° (360° = um círculo completo) 2. `RAD`: significa Radiant (radiano) e representa o ângulo de 0 a 2 × π (2 × π = um círculo completo) 3. `GRAD`: significa grado e

representa o ângulo de 0 a 400 (400 = um círculo completo) 4. `RPG`: significa um formato específico do RPG Maker e representa o ângulo de 0 a 256 (256 = um círculo completo)

O próprio ângulo também pode estar fora dessa faixa. Nesse caso, ele é interpretado como se estivesse dentro da faixa de um círculo completo. Para entender melhor, pode-se somar/subtrair o valor de um círculo completo até que a faixa correspondente seja novamente alcançada. Por exemplo: -1° corresponde a 359°, 361° corresponde a 1°, e assim por diante.

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** Ângulo do cosseno indicado

## Exemplo

Chamar o método:

```
$
// Converter vários valores de cosseno em ângulo (DEG, R
f[1]  = Math.ACos(Math.Cos(  0.0, DEG), DEG);   /
f[2]  = Math.ACos(Math.Cos( 90.0, DEG), DEG);   /
f[3]  = Math.ACos(Math.Cos(180.0, DEG), DEG);   /
f[4]  = Math.ACos(Math.Cos(270.0, DEG), DEG);   /
f[5]  = Math.ACos(Math.Cos(  0.0, DEG), RAD);   /
f[6]  = Math.ACos(Math.Cos( 90.0, DEG), RAD);   /
f[7]  = Math.ACos(Math.Cos(180.0, DEG), RAD);   /
f[8]  = Math.ACos(Math.Cos(270.0, DEG), RAD);   /
f[9]  = Math.ACos(Math.Cos(  0.0, DEG), GRAD);  /
f[10] = Math.ACos(Math.Cos( 90.0, DEG), GRAD);  /
f[11] = Math.ACos(Math.Cos(180.0, DEG), GRAD);  /
f[12] = Math.ACos(Math.Cos(270.0, DEG), GRAD);  /
f[13] = Math.ACos(Math.Cos(  0.0, DEG), RPG);   /
f[14] = Math.ACos(Math.Cos( 90.0, DEG), RPG);   /
f[15] = Math.ACos(Math.Cos(180.0, DEG), RPG);   /
f[16] = Math.ACos(Math.Cos(270.0, DEG), RPG);   /
```

> **Observação:** Para converter ângulos, deve-se usar o método `Angle` do objeto `Convert`.

## Observações adicionais

O cálculo subjacente usa a arco-tangente parcial. Isso está sujeito a algumas limitações, já que, por exemplo, Math.Cos(90.0, DEG) é o mesmo que Math.Cos(270.0, DEG), e por isso não é possível distinguir qual era o ângulo original nesses valores extremos.

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.Cos()`, `Math.Pi`, `Convert.Angle()`

MÉTODO

# Método: Math.ACot()

## Sintaxe

```
Math.ACot(Cotangens, [Formato])
```

## Descrição

Calcula o ângulo a partir da cotangente (= arcocotangente).

## Lista de parâmetros

### Parâmetro: `Cotangente`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** Indica a cotangente a converter.

### Parâmetro: `Formato`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 2 (= RAD)
- **Faixa de valores:** 1 ... 4
- **Descrição:** Indica o formato em que o ângulo é devolvido. Para isso são usadas as `constantes de formato de ângulo`. Os seguintes formatos estão definidos:

1. `DEG`: significa Degree (grau) e representa o ângulo de 0 a 360° (360° = um círculo completo) 2. `RAD`: significa Radiant (radiano) e representa o ângulo de 0 a 2 × π (2 × π = um círculo completo) 3. `GRAD`: significa grado e representa o ângulo de 0 a 400 (400 = um círculo completo) 4. `RPG`: significa um formato específico do RPG Maker e representa o ângulo de 0 a 256 (256 = um círculo completo)

O próprio ângulo também pode estar fora dessa faixa. Nesse caso, ele é interpretado como se estivesse dentro da faixa de um círculo completo. Para entender melhor, pode-se somar/subtrair o valor de um círculo completo até que a faixa correspondente seja novamente alcançada. Por exemplo: -1° corresponde a 359°, 361° corresponde a 1°, e assim por diante.

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** Ângulo da cotangente indicada

## Exemplo

Chamar o método:

```
$
// Converter vários valores de cotangente em ângulo (DEG, R
f[1]  = Math.ACot(Math.Cot( 45.0, DEG), DEG);   /
f[2]  = Math.ACot(Math.Cot( 90.0, DEG), DEG);   /
f[3]  = Math.ACot(Math.Cot(135.0, DEG), DEG);   /
f[4]  = Math.ACot(Math.Cot(270.0, DEG), DEG);   /
```

```
f[5]  = Math.ACot(Math.Cot( 45.0, DEG), RAD);   /
f[6]  = Math.ACot(Math.Cot( 90.0, DEG), RAD);   /
f[7]  = Math.ACot(Math.Cot(135.0, DEG), RAD);   /
f[8]  = Math.ACot(Math.Cot(270.0, DEG), RAD);   /
f[9]  = Math.ACot(Math.Cot( 45.0, DEG), GRAD);  /
f[10] = Math.ACot(Math.Cot( 90.0, DEG), GRAD);  /
f[11] = Math.ACot(Math.Cot(135.0, DEG), GRAD);  /
f[12] = Math.ACot(Math.Cot(270.0, DEG), GRAD);  /
f[13] = Math.ACot(Math.Cot( 45.0, DEG), RPG);   /
f[14] = Math.ACot(Math.Cot( 90.0, DEG), RPG);   /
f[15] = Math.ACot(Math.Cot(135.0, DEG), RPG);   /
f[16] = Math.ACot(Math.Cot(270.0, DEG), RPG);   /
```

> **Observação:** Para converter ângulos, deve-se usar o método `Angle` do objeto `Convert`.

## Observações adicionais

O cálculo subjacente usa a arco-tangente parcial. Isso está sujeito a algumas limitações, já que, por exemplo, Math.Cot(90.0, DEG) é o mesmo que Math.Cot(270.0, DEG), e por isso não é possível distinguir qual era o ângulo original nesses valores extremos.

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.Cot()`, `Math.Pi`, `Convert.Angle()`

MÉTODO

# Método: Math.ACsc()

## Sintaxe

```
Math.ACsc(Cosekans, [Formato])
```

## Descrição

Calcula o ângulo a partir da cossecante (= arcocossecante).

## Lista de parâmetros

### Parâmetro: `Cossecante`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** Indica a cossecante a converter.

### Parâmetro: `Formato`

- **Tipo de dado:** `dword`
- **Opcional:** sim

- **Valor padrão:** 2 (= RAD)
- **Faixa de valores:** 1 ... 4
- **Descrição:** Indica o formato em que o ângulo é devolvido. Para isso são usadas as `constantes de formato de ângulo`. Os seguintes formatos estão definidos:

1. `DEG`: significa Degree (grau) e representa o ângulo de 0 a 360° (360° = um círculo completo) 2. `RAD`: significa Radiant (radiano) e representa o ângulo de 0 a 2 × π (2 × π = um círculo completo) 3. `GRAD`: significa grado e representa o ângulo de 0 a 400 (400 = um círculo completo) 4. `RPG`: significa um formato específico do RPG Maker e representa o ângulo de 0 a 256 (256 = um círculo completo)

O próprio ângulo também pode estar fora dessa faixa. Nesse caso, ele é interpretado como se estivesse dentro da faixa de um círculo completo. Para entender melhor, pode-se somar/subtrair o valor de um círculo completo até que a faixa correspondente seja novamente alcançada. Por exemplo: -1° corresponde a 359°, 361° corresponde a 1°, e assim por diante.

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** Ângulo da cossecante indicada

## Exemplo

Chamar o método:

```
$
// Converter vários valores de cossecante em ângulo (DEG, R
f[1]  = Math.ACsc(Math.Csc( 30.0, DEG), DEG);   /
f[2]  = Math.ACsc(Math.Csc( 90.0, DEG), DEG);   /
f[3]  = Math.ACsc(Math.Csc(210.0, DEG), DEG);   /
f[4]  = Math.ACsc(Math.Csc(270.0, DEG), DEG);   /
f[5]  = Math.ACsc(Math.Csc( 30.0, DEG), RAD);   /
f[6]  = Math.ACsc(Math.Csc( 90.0, DEG), RAD);   /
f[7]  = Math.ACsc(Math.Csc(210.0, DEG), RAD);   /
f[8]  = Math.ACsc(Math.Csc(270.0, DEG), RAD);   /
f[9]  = Math.ACsc(Math.Csc( 30.0, DEG), GRAD);  /
f[10] = Math.ACsc(Math.Csc( 90.0, DEG), GRAD);  /
f[11] = Math.ACsc(Math.Csc(210.0, DEG), GRAD);  /
f[12] = Math.ACsc(Math.Csc(270.0, DEG), GRAD);  /
f[13] = Math.ACsc(Math.Csc( 30.0, DEG), RPG);   /
f[14] = Math.ACsc(Math.Csc( 90.0, DEG), RPG);   /
f[15] = Math.ACsc(Math.Csc(210.0, DEG), RPG);   /
f[16] = Math.ACsc(Math.Csc(270.0, DEG), RPG);   /
```

> **Observação:** Para converter ângulos, deve-se usar o método `Angle` do objeto `Convert`.

## Observações adicionais

O cálculo subjacente usa a arco-tangente parcial. Isso está sujeito a algumas limitações, já que, por exemplo, Math.Csc(45.0, DEG) é o mesmo que Math.Csc(135.0, DEG), e por isso não é possível distinguir qual era o ângulo original nesses valores extremos.

A cossecante 0 não pode ser convertida em ângulo, pois isso exigiria matematicamente uma divisão por 0.

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.Csc()`, `Math.Pi`, `Convert.Angle()`

MÉTODO

# Método: Math.ASec()

## Sintaxe

```
Math.ASec(Sekans, [Formato])
```

## Descrição

Calcula o ângulo a partir da secante (= arcosecante).

## Lista de parâmetros

### Parâmetro: `Secante`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** Indica a secante a converter.

### Parâmetro: `Formato`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 2 (= RAD)
- **Faixa de valores:** 1 ... 4
- **Descrição:** Indica o formato em que o ângulo é devolvido. Para isso são usadas as `constantes de formato de ângulo`. Os seguintes formatos estão definidos:

1. `DEG`: significa Degree (grau) e representa o ângulo de 0 a 360° (360° = um círculo completo) 2. `RAD`: significa Radiant (radiano) e representa o ângulo de 0 a 2 × π (2 × π = um círculo completo) 3. `GRAD`: significa grado e representa o ângulo de 0 a 400 (400 = um círculo completo) 4. `RPG`: significa um formato específico do RPG Maker e representa o ângulo de 0 a 256 (256 = um círculo completo)

O próprio ângulo também pode estar fora dessa faixa. Nesse caso, ele é interpretado como se estivesse dentro da faixa de um círculo completo. Para entender melhor, pode-se somar/subtrair o valor de um círculo completo até que a faixa correspondente seja novamente alcançada. Por exemplo: -1° corresponde a 359°, 361° corresponde a 1°, e assim por diante.

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** Ângulo da secante indicada

## Exemplo

Chamar o método:

```
$
// Converter vários valores de secante em ângulo (DEG, R
f[1]  = Math.ASec(Math.Sec(  0.0, DEG), DEG);   /
f[2]  = Math.ASec(Math.Sec( 60.0, DEG), DEG);   /
f[3]  = Math.ASec(Math.Sec(180.0, DEG), DEG);   /
f[4]  = Math.ASec(Math.Sec(240.0, DEG), DEG);   /
f[5]  = Math.ASec(Math.Sec(  0.0, DEG), RAD);   /
f[6]  = Math.ASec(Math.Sec( 60.0, DEG), RAD);   /
f[7]  = Math.ASec(Math.Sec(180.0, DEG), RAD);   /
f[8]  = Math.ASec(Math.Sec(240.0, DEG), RAD);   /
f[9]  = Math.ASec(Math.Sec(  0.0, DEG), GRAD);  /
f[10] = Math.ASec(Math.Sec( 60.0, DEG), GRAD);  /
f[11] = Math.ASec(Math.Sec(180.0, DEG), GRAD);  /
f[12] = Math.ASec(Math.Sec(240.0, DEG), GRAD);  /
f[13] = Math.ASec(Math.Sec(  0.0, DEG), RPG);   /
f[14] = Math.ASec(Math.Sec( 60.0, DEG), RPG);   /
f[15] = Math.ASec(Math.Sec(180.0, DEG), RPG);   /
f[16] = Math.ASec(Math.Sec(240.0, DEG), RPG);   /
```

> **Observação:** Para converter ângulos, deve-se usar o método `Angle` do objeto `Convert`.

## Observações adicionais

O cálculo subjacente usa a arco-tangente parcial. Isso está sujeito a algumas limitações, já que, por exemplo, Math.Sec(120.0, DEG) é o mesmo que Math.Sec(240.0, DEG), e por isso não é possível distinguir qual era o ângulo original nesses valores extremos.

A secante 0 não pode ser convertida em ângulo, pois isso exigiria matematicamente uma divisão por 0.

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.Sec()`, `Math.Pi`, `Convert.Angle()`

MÉTODO

# Método: Math.ASin()

## Sintaxe

```
Math.ASin(Sinus, [Formato])
```

## Descrição

Calcula o ângulo a partir do seno (= arcoseno).

## Lista de parâmetros

### Parâmetro: `Seno`

- **Tipo de dado:** `float`

- **Opcional:** não
- **Descrição:** Indica o seno a converter.

### Parâmetro: `Formato`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 2 (= RAD)
- **Faixa de valores:** 1 ... 4
- **Descrição:** Indica o formato em que o ângulo é devolvido. Para isso são usadas as `constantes de formato de ângulo`. Os seguintes formatos estão definidos:

1. `DEG`: significa Degree (grau) e representa o ângulo de 0 a 360° (360° = um círculo completo) 2. `RAD`: significa Radiant (radiano) e representa o ângulo de 0 a 2 × π (2 × π = um círculo completo) 3. `GRAD`: significa grado e representa o ângulo de 0 a 400 (400 = um círculo completo) 4. `RPG`: significa um formato específico do RPG Maker e representa o ângulo de 0 a 256 (256 = um círculo completo)

O próprio ângulo também pode estar fora dessa faixa. Nesse caso, ele é interpretado como se estivesse dentro da faixa de um círculo completo. Para entender melhor, pode-se somar/subtrair o valor de um círculo completo até que a faixa correspondente seja novamente alcançada. Por exemplo: -1° corresponde a 359°, 361° corresponde a 1°, e assim por diante.

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** Ângulo do seno indicado

## Exemplo

Chamar o método:

```
$
// Converter vários valores de seno em ângulo (DEG, R
f[1]  = Math.ASin(Math.Sin(  0.0, DEG), DEG);   /
f[2]  = Math.ASin(Math.Sin( 90.0, DEG), DEG);   /
f[3]  = Math.ASin(Math.Sin(180.0, DEG), DEG);   /
f[4]  = Math.ASin(Math.Sin(270.0, DEG), DEG);   /
f[5]  = Math.ASin(Math.Sin(  0.0, DEG), RAD);   /
f[6]  = Math.ASin(Math.Sin( 90.0, DEG), RAD);   /
f[7]  = Math.ASin(Math.Sin(180.0, DEG), RAD);   /
f[8]  = Math.ASin(Math.Sin(270.0, DEG), RAD);   /
f[9]  = Math.ASin(Math.Sin(  0.0, DEG), GRAD);  /
f[10] = Math.ASin(Math.Sin( 90.0, DEG), GRAD);  /
f[11] = Math.ASin(Math.Sin(180.0, DEG), GRAD);  /
f[12] = Math.ASin(Math.Sin(270.0, DEG), GRAD);  /
f[13] = Math.ASin(Math.Sin(  0.0, DEG), RPG);   /
f[14] = Math.ASin(Math.Sin( 90.0, DEG), RPG);   /
f[15] = Math.ASin(Math.Sin(180.0, DEG), RPG);   /
f[16] = Math.ASin(Math.Sin(270.0, DEG), RPG);   /
```

> **Observação:** Para converter ângulos, deve-se usar o método `Angle` do objeto `Convert`.

## Observações adicionais

O cálculo subjacente usa a arco-tangente parcial. Isso está sujeito a algumas limitações, já que, por exemplo, Math.Sin(0.0, DEG) é o mesmo que Math.Sin(180.0, DEG), e por isso não é possível distinguir qual era o ângulo

original nesses valores extremos.

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.Sin()`, `Math.Pi`, `Convert.Angle()`

MÉTODO

# Método: Math.ATan()

## Sintaxe

```
Math.ATan(Tangens, [Formato])
```

## Descrição

Calcula o ângulo a partir da tangente (= arcotangente).

## Lista de parâmetros

### Parâmetro: `Tangente`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** Indica a tangente a converter.

### Parâmetro: `Formato`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 2 (= RAD)
- **Faixa de valores:** 1 ... 4
- **Descrição:** Indica o formato em que o ângulo é devolvido. Para isso são usadas as `constantes de formato de ângulo`. Os seguintes formatos estão definidos:

1. `DEG`: significa Degree (grau) e representa o ângulo de 0 a 360° (360° = um círculo completo) 2. `RAD`: significa Radiant (radiano) e representa o ângulo de 0 a 2 × π (2 × π = um círculo completo) 3. `GRAD`: significa grado e representa o ângulo de 0 a 400 (400 = um círculo completo) 4. `RPG`: significa um formato específico do RPG Maker e representa o ângulo de 0 a 256 (256 = um círculo completo)

O próprio ângulo também pode estar fora dessa faixa. Nesse caso, ele é interpretado como se estivesse dentro da faixa de um círculo completo. Para entender melhor, pode-se somar/subtrair o valor de um círculo completo até que a faixa correspondente seja novamente alcançada. Por exemplo: -1° corresponde a 359°, 361° corresponde a 1°, e assim por diante.

### Valor de retorno

- **Tipo de dado:** `float`

- **Descrição:** Ângulo da tangente indicada

## Exemplo

Chamar o método:

```
$
// Converter vários valores de tangente em ângulo (DEG, R
f[1]  = Math.ATan(Math.Tan(  0.0, DEG), DEG);   /
f[2]  = Math.ATan(Math.Tan( 45.0, DEG), DEG);   /
f[3]  = Math.ATan(Math.Tan(135.0, DEG), DEG);   /
f[4]  = Math.ATan(Math.Tan(180.0, DEG), DEG);   /
f[5]  = Math.ATan(Math.Tan(  0.0, DEG), RAD);   /
f[6]  = Math.ATan(Math.Tan( 45.0, DEG), RAD);   /
f[7]  = Math.ATan(Math.Tan(135.0, DEG), RAD);   /
f[8]  = Math.ATan(Math.Tan(180.0, DEG), RAD);   /
f[9]  = Math.ATan(Math.Tan(  0.0, DEG), GRAD);  /
f[10] = Math.ATan(Math.Tan( 45.0, DEG), GRAD);  /
f[11] = Math.ATan(Math.Tan(135.0, DEG), GRAD);  /
f[12] = Math.ATan(Math.Tan(180.0, DEG), GRAD);  /
f[13] = Math.ATan(Math.Tan(  0.0, DEG), RPG);   /
f[14] = Math.ATan(Math.Tan( 45.0, DEG), RPG);   /
f[15] = Math.ATan(Math.Tan(135.0, DEG), RPG);   /
f[16] = Math.ATan(Math.Tan(180.0, DEG), RPG);   /
```

> **Observação:** Para converter ângulos, deve-se usar o método `Angle` do objeto `Convert`.

## Observações adicionais

O cálculo subjacente usa a arco-tangente parcial. Isso está sujeito a algumas limitações, já que, por exemplo, Math.Tan(0.0, DEG) é o mesmo que Math.Tan(180.0, DEG), e por isso não é possível distinguir qual era o ângulo original nesses valores extremos.

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.Tan()`, `Math.Pi`, `Convert.Angle()`

CATEGORIA

# Categoria: Funções trigonométricas

## Descrição

Contém funções trigonométricas.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Cos()` | Método | Calcula o cosseno de um ângulo (= cateto adjacente / hipotenusa). |

| Nome | Tipo | Resumo |
|---|---|---|
| `Cot()` | Método | Calcula a cotangente de um ângulo (= cateto adjacente / cateto oposto). |
| `Csc()` | Método | Calcula a cossecante de um ângulo (= hipotenusa / cateto oposto). |
| `Sec()` | Método | Calcula a secante de um ângulo (= hipotenusa / cateto adjacente). |
| `Sin()` | Método | Calcula o seno de um ângulo (= cateto oposto / hipotenusa). |
| `Tan()` | Método | Calcula a tangente de um ângulo (= cateto oposto / cateto adjacente). |

MÉTODO

# Método: Math.Cos()

## Sintaxe

```
Math.Cos(Winkel, [Formato])
```

## Descrição

Calcula o cosseno de um ângulo (= cateto adjacente / hipotenusa).

## Lista de parâmetros

### Parâmetro: `Ângulo`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** Indica o ângulo a converter. O formato em que o ângulo é informado é determinado pelo segundo parâmetro.

### Parâmetro: `Formato`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 2 (= RAD)
- **Faixa de valores:** 1 ... 4
- **Descrição:** Indica o formato em que o ângulo é informado. Para isso são usadas as `constantes de formato de ângulo`. Os seguintes formatos estão definidos:

1. `DEG`: significa Degree (grau) e representa o ângulo de 0 a 360° (360° = um círculo completo) 2. `RAD`: significa Radiant (radiano) e representa o ângulo de 0 a 2 × π (2 × π = um círculo completo) 3. `GRAD`: significa grado e representa o ângulo de 0 a 400 (400 = um círculo completo) 4. `RPG`: significa um formato específico do RPG Maker e representa o ângulo de 0 a 256 (256 = um círculo completo)

O próprio ângulo também pode estar fora dessa faixa. Nesse caso, ele é interpretado como se estivesse dentro da faixa de um círculo completo. Para entender melhor, pode-se somar/subtrair o valor de um círculo completo até que a faixa correspondente seja novamente alcançada. Por exemplo: -1° corresponde a 359°, 361° corresponde a 1°, e assim por diante.

### Valor de retorno

- **Tipo de dado:** `float`

- **Descrição:** O cosseno do ângulo indicado

## Exemplo

Chamar o método:

```
$
// Converter vários ângulos (DEG, RAD, GRAD
f[1]  = Math.Cos(  0.0, DEG);          // =  1.0
f[2]  = Math.Cos( 90.0, DEG);          // =  0.0
f[3]  = Math.Cos(180.0, DEG);          // = -1.0
f[4]  = Math.Cos(270.0, DEG);          // =  0.0
f[5]  = Math.Cos(Math.Pi * 0.0, RAD);  // =  1.0
f[6]  = Math.Cos(Math.Pi * 0.5, RAD);  // =  0.0
f[7]  = Math.Cos(Math.Pi * 1.0, RAD);  // = -1.0
f[8]  = Math.Cos(Math.Pi * 1.5, RAD);  // =  0.0
f[9]  = Math.Cos(  0.0, GRAD);         // =  1.0
f[10] = Math.Cos(100.0, GRAD);         // =  0.0
f[11] = Math.Cos(200.0, GRAD);         // = -1.0
f[12] = Math.Cos(300.0, GRAD);         // =  0.0
f[13] = Math.Cos(  0.0, RPG);          // =  1.0
f[14] = Math.Cos( 64.0, RPG);          // =  0.0
f[15] = Math.Cos(128.0, RPG);          // = -1.0
f[16] = Math.Cos(192.0, RPG);          // =  0.0
```

Desenhar um círculo:

```
$
// Criar uma Picture vazia de tamanho 100 x 100
Picture[1].AuroraSheet.Create(100, 100);
Picture[1].AuroraSheet.Palette[0] = 0xFFFFFF;
Picture[1].AuroraSheet.Palette[1] = 0x000000;
Picture[1].AuroraSheet.FlushPalette();
Picture[1].AuroraSheet.UseMaskColor = false;
Picture[1].X = 160;
Picture[1].Y = 120;
Picture[1].Transparency = 0;
Picture[1].Magnification = 100;
// Desenhar um círculo na imagem (centralizado; r
for(f[1] = 0.0; f[1] < 360.0; f[1] += 0.5)
  f[2] = Math.Cos(f[1], DEG) * 30.0 + 50.0;
  f[3] = Math.Sin(f[1], DEG) * 30.0 + 50.0;
  Picture[1].AuroraSheet.Pixel[f[2], f[3]] = 1;
next;
```

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.Sin()`, `Math.ACos()`, `Math.Pi`, `Picture`, `Convert.Angle()`

MÉTODO

# Método: Math.Cot()

## Sintaxe

```
Math.Cot(Winkel, [Formato])
```

## Descrição

Calcula a cotangente de um ângulo (= cateto adjacente / cateto oposto).

## Lista de parâmetros

### Parâmetro: `Ângulo`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** Indica o ângulo a converter. O formato em que o ângulo é informado é determinado pelo segundo parâmetro.

### Parâmetro: `Formato`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 2 (= RAD)
- **Faixa de valores:** 1 ... 4
- **Descrição:** Indica o formato em que o ângulo é informado. Para isso são usadas as `constantes de formato de ângulo`. Os seguintes formatos estão definidos:

1. `DEG`: significa Degree (grau) e representa o ângulo de 0 a 360° (360° = um círculo completo) 2. `RAD`: significa Radiant (radiano) e representa o ângulo de 0 a 2 × π (2 × π = um círculo completo) 3. `GRAD`: significa grado e representa o ângulo de 0 a 400 (400 = um círculo completo) 4. `RPG`: significa um formato específico do RPG Maker e representa o ângulo de 0 a 256 (256 = um círculo completo)

O próprio ângulo também pode estar fora dessa faixa. Nesse caso, ele é interpretado como se estivesse dentro da faixa de um círculo completo. Para entender melhor, pode-se somar/subtrair o valor de um círculo completo até que a faixa correspondente seja novamente alcançada. Por exemplo: -1° corresponde a 359°, 361° corresponde a 1°, e assim por diante.

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** A cotangente do ângulo indicado

## Exemplo

Chamar o método:

```
$
// Converter vários ângulos (DEG, RAD, GRAD
f[1]  = Math.Cot( 45.0, DEG);          // =  1.0
f[2]  = Math.Cot( 90.0, DEG);          // =  0.0
f[3]  = Math.Cot(135.0, DEG);          // = -1.0
```

```
f[4]  = Math.Cot(270.0, DEG);          // =  0.0
f[5]  = Math.Cot(Math.Pi * 0.25, RAD); // =  1.0
f[6]  = Math.Cot(Math.Pi * 0.50, RAD); // =  0.0
f[7]  = Math.Cot(Math.Pi * 0.75, RAD); // = -1.0
f[8]  = Math.Cot(Math.Pi * 1.50, RAD); // =  0.0
f[9]  = Math.Cot( 50.0, GRAD);         // =  1.0
f[10] = Math.Cot(100.0, GRAD);         // =  0.0
f[11] = Math.Cot(150.0, GRAD);         // = -1.0
f[12] = Math.Cot(300.0, GRAD);         // =  0.0
f[13] = Math.Cot( 32.0, RPG);          // =  1.0
f[14] = Math.Cot( 64.0, RPG);          // =  0.0
f[15] = Math.Cot( 96.0, RPG);          // = -1.0
f[16] = Math.Cot(192.0, RPG);          // =  0.0
```

## Observações adicionais

Os ângulos 0° e 180° (ou equivalentes) não podem ser convertidos em cotangente, pois isso exigiria matematicamente uma divisão por 0.

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.Tan()`, `Math.ACot()`, `Math.Pi`, `Convert.Angle()`

MÉTODO

# Método: Math.Csc()

## Sintaxe

```
Math.Csc(Winkel, [Formato])
```

## Descrição

Calcula a cossecante de um ângulo (= hipotenusa / cateto oposto).

## Lista de parâmetros

### Parâmetro: `Ângulo`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** Indica o ângulo a converter. O formato em que o ângulo é informado é determinado pelo segundo parâmetro.

### Parâmetro: `Formato`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 2 (= RAD)

- **Faixa de valores:** 1 ... 4
- **Descrição:** Indica o formato em que o ângulo é informado. Para isso são usadas as `constantes de formato de ângulo`. Os seguintes formatos estão definidos:

1. `DEG`: significa Degree (grau) e representa o ângulo de 0 a 360° (360° = um círculo completo) 2. `RAD`: significa Radiant (radiano) e representa o ângulo de 0 a 2 × π (2 × π = um círculo completo) 3. `GRAD`: significa grado e representa o ângulo de 0 a 400 (400 = um círculo completo) 4. `RPG`: significa um formato específico do RPG Maker e representa o ângulo de 0 a 256 (256 = um círculo completo)

O próprio ângulo também pode estar fora dessa faixa. Nesse caso, ele é interpretado como se estivesse dentro da faixa de um círculo completo. Para entender melhor, pode-se somar/subtrair o valor de um círculo completo até que a faixa correspondente seja novamente alcançada. Por exemplo: -1° corresponde a 359°, 361° corresponde a 1°, e assim por diante.

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** A cossecante do ângulo indicado

## Exemplo

Chamar o método:

```
$
// Converter vários ângulos (DEG, RAD, GRAD
f[1]  = Math.Csc( 30.0, DEG);             // =  2
f[2]  = Math.Csc( 90.0, DEG);             // =  1
f[3]  = Math.Csc(210.0, DEG);             // = -2
f[4]  = Math.Csc(270.0, DEG);             // = -1
f[5]  = Math.Csc(Math.Pi * 0.1666, RAD);  // =  2
f[6]  = Math.Csc(Math.Pi * 0.5,    RAD);  // =  1
f[7]  = Math.Csc(Math.Pi * 1.1666, RAD);  // = -2
f[8]  = Math.Csc(Math.Pi * 1.5,    RAD);  // = -1
f[9]  = Math.Csc( 33.3, GRAD);            // =  2
f[10] = Math.Csc(100.0, GRAD);            // =  1
f[11] = Math.Csc(233.3, GRAD);            // = -2
f[12] = Math.Csc(300.0, GRAD);            // = -1
f[13] = Math.Csc( 21.3, RPG);             // =  2
f[14] = Math.Csc( 64.0, RPG);             // =  1
f[15] = Math.Csc(149.3, RPG);             // = -2
f[16] = Math.Csc(192.0, RPG);             // = -1
```

## Observações adicionais

Os ângulos 0° e 180° (ou equivalentes) não podem ser convertidos em cossecante, pois isso exigiria matematicamente uma divisão por 0.

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.Cos()`, `Math.ASin()`, `Convert.Angle()`

MÉTODO

# Método: Math.Sec()

## Sintaxe

```
Math.Sec(Winkel, [Formato])
```

## Descrição

Calcula a secante de um ângulo (= hipotenusa / cateto adjacente).

## Lista de parâmetros

### Parâmetro: `Ângulo`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** Indica o ângulo a converter. O formato em que o ângulo é informado é determinado pelo segundo parâmetro.

### Parâmetro: `Formato`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 2 (= RAD)
- **Faixa de valores:** 1 ... 4
- **Descrição:** Indica o formato em que o ângulo é informado. Para isso são usadas as `constantes de formato de ângulo`. Os seguintes formatos estão definidos:

1. `DEG`: significa Degree (grau) e representa o ângulo de 0 a 360° (360° = um círculo completo) 2. `RAD`: significa Radiant (radiano) e representa o ângulo de 0 a 2 × π (2 × π = um círculo completo) 3. `GRAD`: significa grado e representa o ângulo de 0 a 400 (400 = um círculo completo) 4. `RPG`: significa um formato específico do RPG Maker e representa o ângulo de 0 a 256 (256 = um círculo completo)

O próprio ângulo também pode estar fora dessa faixa. Nesse caso, ele é interpretado como se estivesse dentro da faixa de um círculo completo. Para entender melhor, pode-se somar/subtrair o valor de um círculo completo até que a faixa correspondente seja novamente alcançada. Por exemplo: -1° corresponde a 359°, 361° corresponde a 1°, e assim por diante.

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** A secante do ângulo indicado

## Exemplo

Chamar o método:

```
$
// Converter vários ângulos (DEG, RAD, GRAD
f[1]  = Math.Sec(  0.0, DEG);             // =  1
f[2]  = Math.Sec( 60.0, DEG);             // =  2
f[3]  = Math.Sec(180.0, DEG);             // = -1
```

```
f[4]  = Math.Sec(240.0, DEG);             // = -2
f[5]  = Math.Sec(Math.Pi * 0.0,    RAD);  // =  1
f[6]  = Math.Sec(Math.Pi * 0.3333, RAD);  // =  2
f[7]  = Math.Sec(Math.Pi * 1.0,    RAD);  // = -1
f[8]  = Math.Sec(Math.Pi * 1.3333, RAD);  // = -2
f[9]  = Math.Sec(  0.0, GRAD);            // =  1
f[10] = Math.Sec( 66.6, GRAD);            // =  2
f[11] = Math.Sec(200.0, GRAD);            // = -1
f[12] = Math.Sec(266.6, GRAD);            // = -2
f[13] = Math.Sec(  0.0, RPG);             // =  1
f[14] = Math.Sec( 42.6, RPG);             // =  2
f[15] = Math.Sec(128.0, RPG);             // = -1
f[16] = Math.Sec(170.6, RPG);             // = -2
```

## Observações adicionais

Os ângulos 90° e 270° (ou equivalentes) não podem ser convertidos em secante, pois isso exigiria matematicamente uma divisão por 0.

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.Csc()`, `Math.ASec()`, `Math.Pi`, `Convert.Angle()`

MÉTODO

# Método: Math.Sin()

## Sintaxe

```
Math.Sin(Winkel, [Formato])
```

## Descrição

Calcula o seno de um ângulo (= cateto oposto / hipotenusa).

## Lista de parâmetros

### Parâmetro: `Ângulo`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** Indica o ângulo a converter. O formato em que o ângulo é informado é determinado pelo segundo parâmetro.

### Parâmetro: `Formato`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 2 (= RAD)

- **Faixa de valores:** 1 ... 4
- **Descrição:** Indica o formato em que o ângulo é informado. Para isso são usadas as `constantes de formato de ângulo`. Os seguintes formatos estão definidos:

1. `DEG`: significa Degree (grau) e representa o ângulo de 0 a 360° (360° = um círculo completo) 2. `RAD`: significa Radiant (radiano) e representa o ângulo de 0 a 2 × π (2 × π = um círculo completo) 3. `GRAD`: significa grado e representa o ângulo de 0 a 400 (400 = um círculo completo) 4. `RPG`: significa um formato específico do RPG Maker e representa o ângulo de 0 a 256 (256 = um círculo completo)

O próprio ângulo também pode estar fora dessa faixa. Nesse caso, ele é interpretado como se estivesse dentro da faixa de um círculo completo. Para entender melhor, pode-se somar/subtrair o valor de um círculo completo até que a faixa correspondente seja novamente alcançada. Por exemplo: -1° corresponde a 359°, 361° corresponde a 1°, e assim por diante.

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** O seno do ângulo indicado

## Exemplo

Chamar o método:

```
$
// Converter vários ângulos (DEG, RAD, GRAD
f[1]  = Math.Sin(  0.0, DEG);          // =  0.0
f[2]  = Math.Sin( 90.0, DEG);          // =  1.0
f[3]  = Math.Sin(180.0, DEG);          // =  0.0
f[4]  = Math.Sin(270.0, DEG);          // = -1.0
f[5]  = Math.Sin(Math.Pi * 0.0, RAD);  // =  0.0
f[6]  = Math.Sin(Math.Pi * 0.5, RAD);  // =  1.0
f[7]  = Math.Sin(Math.Pi * 1.0, RAD);  // =  0.0
f[8]  = Math.Sin(Math.Pi * 1.5, RAD);  // = -1.0
f[9]  = Math.Sin(  0.0, GRAD);         // =  0.0
f[10] = Math.Sin(100.0, GRAD);         // =  1.0
f[11] = Math.Sin(200.0, GRAD);         // =  0.0
f[12] = Math.Sin(300.0, GRAD);         // = -1.0
f[13] = Math.Sin(  0.0, RPG);          // =  0.0
f[14] = Math.Sin( 64.0, RPG);          // =  1.0
f[15] = Math.Sin(128.0, RPG);          // =  0.0
f[16] = Math.Sin(192.0, RPG);          // = -1.0
```

Desenhar um círculo:

```
$
// Criar uma Picture vazia de tamanho 100 x 100
Picture[1].AuroraSheet.Create(100, 100);
Picture[1].AuroraSheet.Palette[0] = 0xFFFFFF;
Picture[1].AuroraSheet.Palette[1] = 0x000000;
Picture[1].AuroraSheet.FlushPalette();
Picture[1].AuroraSheet.UseMaskColor = false;
Picture[1].X = 160;
Picture[1].Y = 120;
Picture[1].Transparency = 0;
Picture[1].Magnification = 100;
// Desenhar um círculo na imagem (centralizado; r
for(f[1] = 0.0; f[1] < 360.0; f[1] += 0.5)
  f[2] = Math.Cos(f[1], DEG) * 30.0 + 50.0;
  f[3] = Math.Sin(f[1], DEG) * 30.0 + 50.0;
  Picture[1].AuroraSheet.Pixel[f[2], f[3]] = 1;
next;
```

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.Cos()`, `Math.ASin()`, `Math.Pi`, `Picture`, `Convert.Angle()`

MÉTODO

# Método: Math.Tan()

## Sintaxe

```
Math.Tan(Winkel, [Formato])
```

## Descrição

Calcula a tangente de um ângulo (= cateto oposto / cateto adjacente).

## Lista de parâmetros

### Parâmetro: `Ângulo`

- **Tipo de dado:** `float`
- **Opcional:** não
- **Descrição:** Indica o ângulo a converter. O formato em que o ângulo é informado é determinado pelo segundo parâmetro.

### Parâmetro: `Formato`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 2 (= RAD)
- **Faixa de valores:** 1 ... 4
- **Descrição:** Indica o formato em que o ângulo é informado. Para isso são usadas as `constantes de formato de ângulo`. Os seguintes formatos estão definidos:

1. `DEG`: significa Degree (grau) e representa o ângulo de 0 a 360° (360° = um círculo completo) 2. `RAD`: significa Radiant (radiano) e representa o ângulo de 0 a 2 × π (2 × π = um círculo completo) 3. `GRAD`: significa grado e representa o ângulo de 0 a 400 (400 = um círculo completo) 4. `RPG`: significa um formato específico do RPG Maker e representa o ângulo de 0 a 256 (256 = um círculo completo)

O próprio ângulo também pode estar fora dessa faixa. Nesse caso, ele é interpretado como se estivesse dentro da faixa de um círculo completo. Para entender melhor, pode-se somar/subtrair o valor de um círculo completo até que a faixa correspondente seja novamente alcançada. Por exemplo: -1° corresponde a 359°, 361° corresponde a 1°, e assim por diante.

### Valor de retorno

- **Tipo de dado:** `float`
- **Descrição:** A tangente do ângulo indicado

## Exemplo

Chamar o método:

```
$
// Converter vários ângulos (DEG, RAD, GRAD
f[1]  = Math.Tan(  0.0, DEG);          // =  0.0
f[2]  = Math.Tan( 45.0, DEG);          // =  1.0
f[3]  = Math.Tan(135.0, DEG);          // = -1.0
f[4]  = Math.Tan(180.0, DEG);          // =  0.0
f[5]  = Math.Tan(Math.Pi * 0.00, RAD); // =  0.0
f[6]  = Math.Tan(Math.Pi * 0.25, RAD); // =  1.0
f[7]  = Math.Tan(Math.Pi * 0.75, RAD); // = -1.0
f[8]  = Math.Tan(Math.Pi * 1.00, RAD); // =  0.0
f[9]  = Math.Tan(  0.0, GRAD);         // =  0.0
f[10] = Math.Tan( 50.0, GRAD);         // =  1.0
f[11] = Math.Tan(150.0, GRAD);         // = -1.0
f[12] = Math.Tan(200.0, GRAD);         // =  0.0
f[13] = Math.Tan(  0.0, RPG);          // =  0.0
f[14] = Math.Tan( 32.0, RPG);          // =  1.0
f[15] = Math.Tan( 96.0, RPG);          // = -1.0
f[16] = Math.Tan(128.0, RPG);          // =  0.0
```

## Observações adicionais

Os ângulos 90° e 270° (ou equivalentes) não podem ser convertidos em tangente, pois isso exigiria matematicamente uma divisão por 0.

## Erros possíveis

`DESTINYERROR_FLOATERROR`

## Veja também

`Math.Cot()`, `Math.ATan()`, `Math.Pi`, `Convert.Angle()`

CATEGORIA

# Categoria: Números aleatórios

## Descrição

Contém funções para gerar números aleatórios.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Rand()` | Método | Gera um número aleatório entre um mínimo e um máximo (o mínimo e o máximo também podem ser devolvidos como resultado). |
| `RandSeed` | Propriedade | Indica o valor-base para o cálculo de números aleatórios. |

MÉTODO

# Método: Math.Rand()

## Sintaxe

```
Math.Rand(Mínimo, Máximo)
```

## Descrição

Gera um número aleatório entre um mínimo e um máximo (o mínimo e o máximo também podem ser devolvidos como resultado). Como esses números são calculados, eles são apenas pseudoaleatórios. O cálculo se baseia no valor de `Math.RandSeed`, que é inicializado com `Time.Tick` quando a Destiny.dll é carregada.

## Lista de parâmetros

### Parâmetro: `Mínimo`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** O menor valor possível do número aleatório gerado

### Parâmetro: `Máximo`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** O maior valor possível do número aleatório gerado

### Valor de retorno

- **Tipo de dado:** `dword`
- **Descrição:** O número aleatório gerado entre o mínimo e o máximo

## Exemplo

```
$
// Gerar vários números aleatórios
v[1] = Math.Rand(1, 100);   // = ? (um número entre
v[2] = Math.Rand(5, 5);     // = 5 (já que nenhum jogo
```

## Observações adicionais

Se o mínimo for maior que o máximo, os valores são simplesmente trocados entre si.

## Veja também

`Math.RandSeed`

PROPRIEDADE

## Propriedade: Math.RandSeed

### Sintaxe

```
Math.RandSeed
```

### Descrição

Indica o valor-base para o cálculo de números aleatórios. Esse valor é alterado a cada chamada de `Math.Rand()`. O valor dessa propriedade é inicializado com `Time.Tick` quando a Destiny.dll é carregada.

### Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não
- **Volátil:** sim

### Exemplo

```
$
// Tornar o "aleatório" previsível
Math.RandSeed = 1;
v[1] = Math.Rand(1, 100);   // = 8 (em Math.Rand
Math.RandSeed = 1;
v[2] = Math.Rand(1, 100);   // = 8 (em Math.Rand
```

### Observações adicionais

Normalmente não é necessário alterar o valor dessa propriedade. Mas essa possibilidade fica em aberto, por exemplo, para guardar um determinado estado aleatório e poder retomá-lo mais tarde.

### Veja também

`Math.Rand()`

OBJETO

# Objeto: Mouse

O objeto Mouse serve para descobrir ou alterar as propriedades atuais do mouse (como, por exemplo, a posição do ponteiro do mouse).

### Exemplo

```
$
// Define a posição atual da primeira Picture no
// (como o posicionamento de uma imagem no RPG Maker
// aqui ainda se considera a largura/altura da imagem
Picture[1].X = Mouse.X - Picture[1].Width / 2;
Picture[1].Y = Mouse.Y - Picture[1].Height / 2;
```

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| X | Propriedade | Com a propriedade X é possível ler/gravar a coordenada X atual do ponteiro do mouse. |
| Y | Propriedade | Com a propriedade Y é possível ler/gravar a coordenada Y atual do ponteiro do mouse. |

PROPRIEDADE

# Propriedade: Mouse.X

Com a propriedade X é possível ler/gravar a coordenada X atual do ponteiro do mouse. O valor dessa propriedade é informado em pixels e se refere aos pixels da tela do jogo. Se o jogo, por exemplo, estiver rodando em modo janela, os pixels da tela são automaticamente convertidos em pixels do jogo. Como esse valor é medido a partir da borda esquerda da tela (ou da borda esquerda da janela do jogo), vale o seguinte: "Quanto menor o valor, mais à esquerda está o ponteiro do mouse."

## Exemplo

```
$
// A coordenada X atual do ponteiro do mouse, guardada em v
v[1] = Mouse.X;
// A coordenada X atual do ponteiro do mouse na tela
Mouse.X = 100;
```

PROPRIEDADE

# Propriedade: Mouse.Y

Com a propriedade Y é possível ler/gravar a coordenada Y atual do ponteiro do mouse. O valor dessa propriedade é informado em pixels e se refere aos pixels da tela do jogo. Se o jogo, por exemplo, estiver rodando em modo janela, os pixels da tela são automaticamente convertidos em pixels do jogo. Como esse valor é medido a partir da borda superior da tela (ou da borda superior da janela do jogo), vale o seguinte: "Quanto menor o valor, mais acima está o ponteiro do mouse."

## Exemplo

```
$
// A coordenada Y atual do ponteiro do mouse, guardada em v
v[1] = Mouse.Y;
// A coordenada Y atual do ponteiro do mouse na tela
Mouse.Y = 50;
```

OBJETO

# Objeto: Panorama

## Sintaxe

```
Panorama
```

## Descrição

O objeto Panorama serve para tornar acessíveis as propriedades do panorama (imagem de fundo) do mapa. Através do subobjeto AuroraSheet, também é possível alterar o panorama.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `AuroraSheet` | Objeto | Com o objeto AuroraSheet é possível acessar o gráfico do panorama atualmente carregado. |
| `Height` | Propriedade | Com a propriedade Height é possível descobrir a altura do panorama. |
| `LoadFromFile()` | Método | Com o método LoadFromFile é possível carregar um determinado panorama. |
| `Name` | Propriedade | Com a propriedade Name é possível acessar o nome do arquivo (sem caminho/extensão) do panorama atualmente usado. |
| `PanHorizontal` | Propriedade | Com a propriedade PanHorizontal é possível ler/gravar se o panorama permanece parado (True) ou se move junto (False) durante o scroll horizontal do mapa. |
| `PanVertical` | Propriedade | Com a propriedade PanVertical é possível ler/gravar se o panorama permanece parado (True) ou se move junto (False) durante o scroll vertical do mapa. |
| `ScrollPosHorizontal` | Propriedade | Com a propriedade ScrollPosHorizontal é possível ler/gravar em que posição o panorama se encontra na direção horizontal. |
| `ScrollPosVertical` | Propriedade | Com a propriedade ScrollPosVertical é possível ler/gravar em que posição o panorama se encontra na direção vertical. |
| `ScrollSpeedHorizontal` | Propriedade | Com a propriedade ScrollSpeedHorizontal é possível ler/gravar a velocidade com que o panorama se move na direção horizontal. |
| `ScrollSpeedVertical` | Propriedade | Com a propriedade ScrollSpeedVertical é possível ler/gravar a velocidade com que o panorama se move na direção vertical. |
| `Used` | Propriedade | Com a propriedade Used é possível ler/gravar se o panorama está sendo usado. |
| `Width` | Propriedade | Com a propriedade Width é possível descobrir a largura do panorama. |

OBJETO

# Objeto: Panorama.AuroraSheet

## Sintaxe

```
Panorama.AuroraSheet
```

## Descrição

Com o objeto AuroraSheet é possível acessar o gráfico do panorama atualmente carregado. Como o objeto AuroraSheet, sendo padronizado, está disponível como subobjeto em vários objetos, sua descrição foi escrita apenas uma vez e colocada como subordinada ao objeto `Picture[]`. A descrição se encontra aqui: `Picture[].AuroraSheet`.

## Exemplo

```
$
// Desenhar listras horizontais no panorama
for(v[1] = 0; v[1] < Panorama.AuroraSheet.Height;
  Panorama.AuroraSheet.DrawLine(0, v[1], Panorama
  Panorama.AuroraSheet.DrawLine(0, v[1] + 1, Pano
next;
```

## Veja também

`Panorama`, `Picture[].AuroraSheet`

PROPRIEDADE

# Propriedade: Panorama.Height

## Sintaxe

```
Panorama.Height
```

## Descrição

Com a propriedade Height é possível descobrir a altura do panorama. Diferente da propriedade `Panorama.AuroraSheet.Height`, aqui o valor 16 é subtraído, já que o RPG_RT sempre faz pequenas alterações ao carregar um panorama, para facilitar o scroll.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** >= 0

## Exemplo

```
$
// Descobrir a largura do panorama e guardar em v[1]
v[1] = Panorama.Width;
// Descobrir a altura do panorama e guardar em v[2]
v[2] = Panorama.Height;
```

## Veja também

`Panorama`, `Panorama.Width`, `Picture[].AuroraSheet`, `Panorama.AuroraSheet.Height`

MÉTODO

# Método: Panorama.LoadFromFile()

## Sintaxe

```
Panorama.LoadFromFile(Nome do arquivo)
```

## Descrição

Com o método LoadFromFile é possível carregar um determinado panorama. Com isso, todas as propriedades adicionais (`Panorama.PanHorizontal`, ...) são reiniciadas.

## Lista de parâmetros

### Parâmetro: `Nome do arquivo`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** Indica o nome do arquivo do panorama (sem caminho/extensão). Esse caminho é relativo, e por isso também pode, em princípio, ser usado para carregar imagens de fora da pasta Panorama. Se for necessário carregar um arquivo com uma extensão específica, pode-se informar a extensão incluindo o caractere NULL, para carregar explicitamente esse arquivo (veja o exemplo).

## Exemplo

```
$
// Primeira chamada com Panorama.LoadFromFile()
Panorama.LoadFromFile("Dawn1");
// Segunda chamada (totalmente idêntica) com Co
Command.SetPanorama("Dawn1", False, False, 0, 0);
// Terceira chamada com Panorama.AuroraSheet.LoadF
// Aqui nenhuma propriedade é reiniciada,
// e o nome do arquivo não é ajustado nem salvo
// Por isso, essa variante não deve, de forma alguma, ser
Panorama.AuroraSheet.LoadFromFile("Panorama\Dawn1
```

Para carregar um panorama bem específico, pode-se anexar um caractere NULL ao fim do nome do arquivo:

```
$
// Carregar o arquivo Panorama\Bild.bmp
Panorama.LoadFromFile("Bild.bmp" .. String.Chr(0)
// Carregar o arquivo Picture\Bild.bmp (outro dire
Panorama.LoadFromFile("..\Picture\Bild.bmp" .. St
```

## Erros possíveis

`DESTINYERROR_EXCEPTION`

## Veja também

`Panorama`, `Panorama.Name`, `Command.SetPanorama()`

PROPRIEDADE

# Propriedade: Panorama.Name

## Sintaxe

```
Panorama.Name
```

## Descrição

Com a propriedade Name é possível acessar o nome do arquivo (sem caminho/extensão) do panorama atualmente usado.

> **Informação:** Essa propriedade pode ser usada junto com o método `Panorama.LoadFromFile()`, para recarregar o panorama.

## Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** sim

## Exemplo

```
$
// Descobrir o nome do panorama atualmente usado
a[1] = Panorama.Name;
// Carregar um novo panorama
Panorama.LoadFromFile("Dawn1");
// Carregar de volta o panorama antigo (desta vez pelo
Command.SetPanorama(a[1], False, False, 0, 0);
```

## Veja também

`Panorama`, `Panorama.LoadFromFile()`, `Command.SetPanorama()`

PROPRIEDADE

# Propriedade: Panorama.PanHorizontal

## Sintaxe

```
Panorama.PanHorizontal
```

## Descrição

Com a propriedade PanHorizontal é possível ler/gravar se o panorama permanece parado (True) ou se move junto (False) durante o scroll horizontal do mapa.

> **Informação:** Esse valor precisa ser True para que `Panorama.ScrollSpeedHorizontal` tenha efeito.

## Sinônimos

PanX

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** não

## Exemplo

```
$
// Verificar se o panorama permanece parado durante o
if(Panorama.PanHorizontal)
  // A velocidade de scroll na direção horizontal
  Panorama.ScrollSpeedHorizontal++;
endif;
```

## Veja também

`Panorama`, `Panorama.PanVertical`, `Panorama.ScrollSpeedHorizontal`

PROPRIEDADE

# Propriedade: Panorama.PanVertical

## Sintaxe

```
Panorama.PanVertical
```

## Descrição

Com a propriedade PanVertical é possível ler/gravar se o panorama permanece parado (True) ou se move junto (False) durante o scroll vertical do mapa.

> **Informação:** Esse valor precisa ser True para que `Panorama.ScrollSpeedVertical` tenha efeito.

## Sinônimos

PanY

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** não

## Exemplo

```
$
// Verificar se o panorama permanece parado durante o
```

```
if(Panorama.PanVertical)
  // A velocidade de scroll na direção vertical
  Panorama.ScrollSpeedVertical++;
endif;
```

## Veja também

`Panorama`, `Panorama.PanHorizontal`, `Panorama.ScrollSpeedVertical`

PROPRIEDADE

# Propriedade: Panorama.ScrollPosHorizontal

## Sintaxe

```
Panorama.ScrollPosHorizontal
```

## Descrição

Com a propriedade ScrollPosHorizontal é possível ler/gravar em que posição o panorama se encontra na direção horizontal. A indicação é feita em pixels. Valores > 0 significam que o panorama está mais à esquerda. O valor = 0 significa que o panorama está na posição horizontal padrão. Valores < 0 significam que o panorama está mais à direita.

## Sinônimos

ScrollPosX

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

## Exemplo

```
$
// Deslocar o panorama 1 pixel para a esquerda
Panorama.ScrollPosHorizontal++;
```

## Veja também

`Panorama`, `Panorama.ScrollPosVertical`, `Panorama.ScrollSpeedHorizontal`, `Panorama.PanHorizontal`

PROPRIEDADE

# Propriedade: Panorama.ScrollPosVertical

## Sintaxe

```
Panorama.ScrollPosVertical
```

## Descrição

Com a propriedade ScrollPosVertical é possível ler/gravar em que posição o panorama se encontra na direção vertical. A indicação é feita em pixels. Valores > 0 significam que o panorama está mais acima. O valor = 0 significa que o panorama está na posição vertical padrão. Valores < 0 significam que o panorama está mais abaixo.

## Sinônimos

ScrollPosY

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

## Exemplo

```
$
// Deslocar o panorama 1 pixel para cima
Panorama.ScrollPosVertical++;
```

## Veja também

`Panorama`, `Panorama.ScrollPosHorizontal`, `Panorama.ScrollSpeedVertical`, `Panorama.PanVertical`

PROPRIEDADE

# Propriedade: Panorama.ScrollSpeedHorizontal

## Sintaxe

```
Panorama.ScrollSpeedHorizontal
```

## Descrição

Com a propriedade ScrollSpeedHorizontal é possível ler/gravar a velocidade com que o panorama se move na direção horizontal. Valores > 0 significam que o panorama se move para a direita. O valor = 0 significa que o panorama não se move horizontalmente. Valores < 0 significam que o panorama se move para a esquerda.

> **Informação:** `Panorama.PanHorizontal` precisa ser True para que essa propriedade tenha efeito.

## Sinônimos

ScrollSpeedX

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

## Exemplo

```
$
// Verificar se o panorama permanece parado durante o
if(Panorama.PanHorizontal)
  // A velocidade de scroll na direção horizontal
  Panorama.ScrollSpeedHorizontal++;
endif;
```

## Veja também

`Panorama`, `Panorama.ScrollSpeedVertical`, `Panorama.PanHorizontal`, `Panorama.ScrollPosHorizontal`

PROPRIEDADE

# Propriedade: Panorama.ScrollSpeedVertical

## Sintaxe

```
Panorama.ScrollSpeedVertical
```

## Descrição

Com a propriedade ScrollSpeedVertical é possível ler/gravar a velocidade com que o panorama se move na direção vertical. Valores > 0 significam que o panorama se move para baixo. O valor = 0 significa que o panorama não se move verticalmente. Valores < 0 significam que o panorama se move para cima.

> **Informação:** `Panorama.PanVertical` precisa ser True para que essa propriedade tenha efeito.

## Sinônimos

ScrollSpeedY

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

## Exemplo

```
$
// Verificar se o panorama permanece parado durante o
if(Panorama.PanVertical)
  // A velocidade de scroll na direção vertical
  Panorama.ScrollSpeedVertical++;
endif;
```

## Veja também

`Panorama`, `Panorama.ScrollSpeedHorizontal`, `Panorama.PanVertical`, `Panorama.ScrollPosVertical`

PROPRIEDADE

# Propriedade: Panorama.Used

## Sintaxe

```
Panorama.Used
```

## Descrição

Com a propriedade Used é possível ler/gravar se o panorama está sendo usado.

> **Informação:** Se esse valor for definido como False, nenhum panorama é mais exibido. O conteúdo do AuroraSheet, porém, permanece inalterado. Assim, é possível ligar/desligar o panorama sem descartar o conteúdo do AuroraSheet.

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** não

## Exemplo

```
$
// Inverter o status de exibição atual do panorama
Panorama.Used = !Panorama.Used;
```

## Veja também

`Panorama`, `Picture[].AuroraSheet`

PROPRIEDADE

# Propriedade: Panorama.Width

## Sintaxe

```
Panorama.Width
```

## Descrição

Com a propriedade Width é possível descobrir a largura do panorama. Diferente da propriedade `Panorama.AuroraSheet.Width`, aqui o valor 16 é subtraído, já que o RPG_RT sempre faz pequenas alterações ao carregar um panorama, para facilitar o scroll.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

- **Faixa de valores:** >= 0

## Exemplo

```
$
// Descobrir a largura do panorama e guardar em v[1]
v[1] = Panorama.Width;
// Descobrir a altura do panorama e guardar em v[2]
v[2] = Panorama.Height;
```

## Veja também

`Panorama`, `Panorama.Height`, `Picture[].AuroraSheet`, `Panorama.AuroraSheet.Width`

OBJETO

# Objeto: Picture

## Sintaxe

```
Picture
```

## Descrição

O objeto Picture serve para dar acesso às Pictures do jogo.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Count` | Propriedade | Com a propriedade Count é possível saber a quantidade de Pictures nesse array. |
| `Picture[]` | Objeto | O objeto Picture[] serve para reunir todas as propriedades de uma Picture. |

## Veja também

`Picture[]`

PROPRIEDADE

# Propriedade: Picture.Count

## Sintaxe

```
Picture.Count
```

## Descrição

Com a propriedade Count é possível saber a quantidade de Pictures nesse array.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

## Exemplo

No exemplo a seguir, partimos do princípio de que o jogo está usando as configurações padrão do RPG Maker 2000.

```
$
// Descobrir a quantidade de Pictures e guardar em v[1]
v[1] = Picture.Count;              // 20
// Executar um laço sobre todas as Pictures e
for(v[2] = 1; v[2] < Picture.Count; v[2]++)
  Picture[v[2]].AuroraSheet.Free();
next;
```

## Veja também

`Picture`, `Picture[]`, `Picture[].AuroraSheet.Free()`

OBJETO

# Objeto: Picture[]

## Sintaxe

```
Picture[Picture-Id]
```

## Descrição

O objeto Picture[] serve para reunir todas as propriedades de uma Picture. Ela é identificada pela Picture-Id. Além disso, é possível alterar o conteúdo da Picture por meio de vários métodos.

## Lista de parâmetros

### Parâmetro: `Picture-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... Picture.Count
- **Descrição:** A Id da Picture

## Erros possíveis

`DESTINYERROR_ARRAYBOUND`

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Action` | Propriedade | Com a propriedade Action é possível ler/gravar a ação de uma Picture. |
| `ActionStrength` | Propriedade | Com a propriedade ActionStrength é possível ler/gravar a intensidade da ação de uma Picture. |
| `ActionValue` | Propriedade | Com a propriedade ActionValue é possível ler/gravar o valor de ação de uma Picture. |
| `AuroraSheet` | Objeto | Com o objeto AuroraSheet dentro de Picture[] é possível acessar a imagem da Picture. |
| `BltRect()` | Método | Com o método BltRect, uma área de imagem de outra Picture pode ser copiada para esta Picture, sem copiar junto a cor de máscara indicada. |
| `Blue` | Propriedade | Com a propriedade Blue é possível ler/gravar a coloração azul de uma Picture, em porcentagem. |
| `Chroma` | Propriedade | Com a propriedade Chroma é possível ler/gravar a crominância (saturação de cor) de uma Picture, em porcentagem. |
| `CopyRect()` | Método | Com o método CopyRect, uma área de imagem de outra Picture pode ser copiada para esta Picture. |
| `DrawLine()` | Método | Com o método DrawLine é possível desenhar uma linha nesta Picture. |
| `Filename` | Propriedade | Com a propriedade Filename é possível obter o nome do arquivo da Picture. |
| `FillRect()` | Método | Com o método FillRect é desenhado um retângulo preenchido. |
| `FlushPalette()` | Método | Com o método FlushPalette é possível fazer com que a paleta atual passe a ser usada como a paleta efetiva. |
| `Green` | Propriedade | Com a propriedade Green é possível ler/gravar a coloração verde de uma Picture, em porcentagem. |
| `Height` | Propriedade | Com a propriedade Height é possível obter a altura de uma Picture em pixels. |
| `LoadFromFile()` | Método | Com o método LoadFromFile é possível carregar uma Picture a partir de um arquivo de imagem. |
| `Magnification` | Propriedade | Com a propriedade Magnification é possível ler/gravar o fator de ampliação de uma Picture, em porcentagem. |
| `MapMove` | Propriedade | Com a propriedade MapMove é possível ler/gravar a propriedade MoveWithMap de uma Picture. |
| `MapX` | Propriedade | Com a propriedade MapX é possível ler/gravar a coordenada X do ponto de referência de uma Picture que se move com o mapa. |
| `MapY` | Propriedade | Com a propriedade MapY é possível ler/gravar a coordenada Y do ponto de referência de uma Picture que se move com o mapa. |
| `MoveX` | Propriedade | Com a propriedade MoveX é possível ler/gravar a coordenada X atual de uma Picture durante o movimento. |
| `MoveY` | Propriedade | Com a propriedade MoveY é possível ler/gravar a coordenada Y atual de uma Picture durante o movimento. |
| `Palette[]` | Propriedade | Com a propriedade Palette é possível ler/gravar uma determinada entrada de paleta da Picture. |
| `Pixel[]` | Propriedade | Com a propriedade Pixel é possível ler/gravar um pixel da Picture. |
| `Red` | Propriedade | Com a propriedade Red é possível ler/gravar a coloração vermelha de uma Picture, em porcentagem. |
| `SaveAs()` | Método | Com o método SaveAs é possível salvar o conteúdo atual da Picture. |
| `Transparency` | Propriedade | Com a propriedade Transparency é possível ler/gravar a transparência de uma Picture, em porcentagem. |

| Nome | Tipo | Resumo |
|---|---|---|
| `UseMaskColor` | Propriedade | Com a propriedade UseMaskColor é possível ler/gravar se o RPG_RT, ao desenhar a Picture, usa a entrada de paleta 0 como cor de máscara (true) ou não (false). |
| `Width` | Propriedade | Com a propriedade Width é possível obter a largura de uma Picture em pixels. |
| `X` | Propriedade | Com a propriedade X é possível ler/gravar a coordenada X da Picture. |
| `Y` | Propriedade | Com a propriedade Y é possível ler/gravar a coordenada Y da Picture. |

PROPRIEDADE

# Propriedade: Picture[].Action

## Sintaxe

```
Picture[Picture-Id].Action
```

## Descrição

Com a propriedade Action é possível ler/gravar a ação de uma Picture. Para isso podem ser usadas as `constantes de ação`. Junto com as propriedades `Picture[].ActionStrength` e `Picture[].ActionValue`, é possível ler/gravar as diversas configurações de ação.

`ACTION_NONE`: A Picture não realiza nenhuma ação. `ACTION_ROTATION`: A Picture gira em torno do próprio eixo. `ACTION_RIPPLE`: A Picture realiza um efeito de onda.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não
- **Faixa de valores:** 0 ... 2
- **Volátil:** sim

## Exemplo

```
$
// Obter a ação atual da primeira Picture
v[1] = Picture[1].Action;

// Definir a ação da primeira Picture como rotação
Picture[1].Action         = ACTION_ROTATION;
Picture[1].ActionStrength = 0;                //
Picture[1].ActionValue    = Convert.Angle(90, DEG
```

## Veja também

`Picture`, `Picture[]`, `Picture[].ActionStrength`, `Picture[].ActionValue`, `Convert.Angle()`

PROPRIEDADE

## Propriedade: Picture[].ActionStrength

### Sintaxe

```
Picture[Picture-Id].ActionStrength
```

### Descrição

Com a propriedade ActionStrength é possível ler/gravar a intensidade da ação de uma Picture. O significado desse valor depende de `Picture[].Action`. Junto com as propriedades `Picture[].Action` e `Picture[].ActionValue`, é possível ler/gravar as diversas configurações de ação. O valor dessa propriedade é somado a `Picture[].ActionValue` cerca de 60 vezes por segundo.

### Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não
- **Volátil:** sim

### Exemplo

```
$
// Obter a intensidade de ação atual da primeira Picture
v[1] = Picture[1].ActionStrength;

// Definir a ação da primeira Picture como rotação
Picture[1].Action         = ACTION_ROTATION;
Picture[1].ActionStrength = 0;                //
Picture[1].ActionValue    = Convert.Angle(90, DEG
```

### Veja também

`Picture`, `Picture[]`, `Picture[].Action`, `Picture[].ActionValue`, `Convert.Angle()`

PROPRIEDADE

## Propriedade: Picture[].ActionValue

### Sintaxe

```
Picture[Picture-Id].ActionValue
```

### Descrição

Com a propriedade ActionValue é possível ler/gravar o valor de ação de uma Picture. O significado desse valor depende de `Picture[].Action`. Junto com as propriedades `Picture[].Action` e `Picture[].ActionStrength`, é possível ler/gravar as diversas configurações de ação. No tipo de ação de rotação, esse valor indica o ângulo da Picture, usando o formato de ângulo específico do RPG Maker. Para converter um ângulo normal para esse

formato, pode-se usar o método `Convert.Angle()`. Com `Picture[].Action` igual a `ACTION_RIPPLE`, esse valor não tem significado.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não
- **Volátil:** sim

## Exemplo

```
$
// Obter o valor de ação atual da primeira Picture
v[1] = Picture[1].ActionValue;

// Definir a ação da primeira Picture como rotação
Picture[1].Action         = ACTION_ROTATION;
Picture[1].ActionStrength = 0;                //
Picture[1].ActionValue    = Convert.Angle(90, DEG
```

## Veja também

`Picture`, `Picture[]`, `Picture[].Action`, `Picture[].ActionStrength`, `Convert.Angle()`

OBJETO

# Objeto: Picture[].AuroraSheet

## Sintaxe

```
Picture[Picture-Id].AuroraSheet
```

## Descrição

Com o objeto AuroraSheet dentro de `Picture[]` é possível acessar a imagem da Picture. Os métodos/propriedades documentados neste objeto valem igualmente para todos os objetos AuroraSheet. Através do `AuroraSheet-Handle`, áreas de imagem de AuroraSheets pertencentes a outros objetos podem ser reaproveitadas (por exemplo, copiar partes do Faceset atual para uma Picture). Objetos AuroraSheet estão definidos como subobjetos nos seguintes objetos:

`Event[]` / `MapEvent[]`, `Faceset`, `Map`, `Panorama`, `Picture[]`

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Clone()` | Método | Com o método Clone é possível copiar o conteúdo e a paleta de outro AuroraSheet. |
| `CopyPalette()` | Método | Com o método CopyPalette é possível copiar a paleta de outro AuroraSheet. |
| `Create()` | Método | Com o método Create é possível criar um novo AuroraSheet. |
| `FlushPalette()` | Método | Com o método FlushPalette é possível fazer com que a paleta atual passe a ser usada como a paleta efetiva. |

| Nome | Tipo | Resumo |
|---|---|---|
| `Free()` | Método | Com o método Free é possível esvaziar o conteúdo de um AuroraSheet. |
| `Handle` | Propriedade | Com a propriedade Handle é possível obter o handle de um AuroraSheet. |
| `Height` | Propriedade | Com a propriedade Height é possível obter a altura de um AuroraSheet. |
| `IsEmpty()` | Método | Com o método IsEmpty é possível verificar se um AuroraSheet está vazio no momento. |
| `LoadFromFile()` | Método | Com o método LoadFromFile é possível carregar um arquivo no AuroraSheet. |
| `Palette` | Objeto | O objeto Palette contém a paleta de um AuroraSheet. |
| `Palette[]` | Propriedade | Com a propriedade Palette[] é possível ler/gravar uma determinada entrada de paleta. |
| `Pixel[]` | Propriedade | Com a propriedade Pixel é possível ler/gravar os pixels de um AuroraSheet. |
| `SaveAs()` | Método | Com o método SaveAs é possível salvar o conteúdo de um AuroraSheet em um bitmap. |
| `UseMaskColor` | Propriedade | Com a propriedade UseMaskColor é possível ler/gravar se o RPG_RT, ao desenhar o AuroraSheet, usa a entrada de paleta 0 como cor de máscara (true) ou não (false). |
| `Width` | Propriedade | Com a propriedade Width é possível obter a largura de um AuroraSheet. |
| `Funções de desenho` | Categoria | Nesta categoria estão reunidas todas as funções de desenho dos objetos AuroraSheet. |

## Veja também

`Picture`, `Picture[]`, `Picture[].AuroraSheet.Handle`

MÉTODO

# Método: Picture[].AuroraSheet.Clone()

## Sintaxe

```
Picture[Picture-Id].AuroraSheet.Clone(Quell-
AuroraSheet)
```

## Descrição

Com o método Clone é possível copiar o conteúdo e a paleta de outro AuroraSheet. Depois de copiar a paleta, o método `FlushPalette` é chamado automaticamente. Informações adicionais, como `UseMaskColor`, não são copiadas.

## Lista de parâmetros

### Parâmetro: `AuroraSheet de origem`

- **Tipo de dado:** `handle`
- **Opcional:** não
- **Descrição:** Indica o `AuroraSheet-Handle` do AuroraSheet cujo conteúdo e paleta devem ser copiados.

## Exemplo

```
$
```

```
// Copiar a segunda Picture para a primeira
Picture[1].AuroraSheet.Clone(Picture[2].AuroraShe
```

## Erros possíveis

`DESTINYERROR_INVALIDHANDLE`, `DESTINYERROR_PICTURE`

## Veja também

`Picture`, `Picture[]`, `Picture[].AuroraSheet`, `Picture[].AuroraSheet.Create()`, `Picture[].AuroraSheet.Free()`, `Picture[].AuroraSheet.IsEmpty()`

MÉTODO

# Método: Picture[].AuroraSheet.CopyPalette()

## Sintaxe

```
Picture[Picture-Id].AuroraSheet.CopyPalette(Quell-
AuroraSheet)
```

## Descrição

Com o método CopyPalette é possível copiar a paleta de outro AuroraSheet. Depois de copiar a paleta, o método `FlushPalette` é chamado automaticamente.

## Lista de parâmetros

### Parâmetro: `AuroraSheet de origem`

- **Tipo de dado:** `handle`
- **Opcional:** não
- **Descrição:** Indica o `AuroraSheet-Handle` do AuroraSheet cuja paleta deve ser copiada.

## Exemplo

```
$

// Copiar a paleta da segunda Picture para a primeira
Picture[1].AuroraSheet.CopyPalette(Picture[2].Aur
```

## Erros possíveis

`DESTINYERROR_INVALIDHANDLE`

## Veja também

`Picture`, `Picture[]`, `Picture[].AuroraSheet`, `Picture[].AuroraSheet.Palette[]`, `Picture[].AuroraSheet.FlushPalette()`

MÉTODO

# Método: Picture[].AuroraSheet.Create()

## Sintaxe

```
Picture[Picture-Id].AuroraSheet.Create(Largura, Altura)
```

## Descrição

Com o método Create é possível criar um novo AuroraSheet. O AuroraSheet anterior é substituído. O AuroraSheet recém-criado é inicializado inteiramente com o índice de paleta 0.

## Lista de parâmetros

### Parâmetro: `Largura`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** > 0
- **Descrição:** Indica a largura do AuroraSheet. Esse valor precisa ser maior que 0.

### Parâmetro: `Altura`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** > 0
- **Descrição:** Indica a altura do AuroraSheet. Esse valor precisa ser maior que 0.

## Exemplo

```
$

// Criar o AuroraSheet e exibi-lo na primeira Pic
Picture[1].AuroraSheet.Create(8 * 8, 8);

// Definir as cores mais comuns
Picture[1].AuroraSheet.Palette[0] = 0x000000;  //
Picture[1].AuroraSheet.Palette[1] = 0xFFFFFF;  //
Picture[1].AuroraSheet.Palette[2] = 0x0000FF;  //
Picture[1].AuroraSheet.Palette[3] = 0x00FF00;  //
Picture[1].AuroraSheet.Palette[4] = 0xFF0000;  //
Picture[1].AuroraSheet.Palette[5] = 0xFF00FF;  //
Picture[1].AuroraSheet.Palette[6] = 0x00FFFF;  //
Picture[1].AuroraSheet.Palette[7] = 0xFFFF00;  //
Picture[1].AuroraSheet.FlushPalette();

// Exibir as cores (0 ... 7)
for(v[1] = 0; v[1] < 8; v[1]++)  // Índice de paleta
  for(v[2] = 0; v[2] < 8; v[2]++)  // X
    for(v[3] = 0; v[3] < 8; v[3]++)  // Y
      Picture[1].AuroraSheet.Pixel[v[1] * 8 + v[2
    next;
  next;
next;
```

```
// NÃO usar a cor de máscara
Picture[1].AuroraSheet.UseMaskColor = false;

// Exibir a primeira Picture centralizada
Picture[1].X = 160;
Picture[1].Y = 120;
```

## Erros possíveis

`DESTINYERROR_MEMORY`, `DESTINYERROR_VALUE`

## Veja também

`Picture`, `Picture[]`, `Picture[].AuroraSheet`, `Picture[].AuroraSheet.Clone()`, `Picture[].AuroraSheet.Free()`, `Picture[].AuroraSheet.IsEmpty()`

MÉTODO

# Método: Picture[].AuroraSheet.FlushPalette()

## Sintaxe

```
Picture[Picture-Id].AuroraSheet.FlushPalette()
```

## Descrição

Com o método FlushPalette é possível fazer com que a paleta atual passe a ser usada como a paleta efetiva. Esse método precisa ser chamado depois que todas as alterações na propriedade `Palette[]` forem feitas.

> **Informação:** Ao alterar uma `entrada de paleta`, a alteração só passa a valer depois que esse método for chamado. O motivo técnico é que o RPG_RT guarda a paleta duas vezes, para que alterações de cor (por exemplo, quando a imagem é exibida dessaturada) só precisem ser feitas na paleta efetiva. Normalmente, primeiro se ajusta a paleta como desejado através da propriedade `Palette[]` e, por fim, executa-se este método.

## Exemplo

```
$

// Criar o AuroraSheet e exibi-lo na primeira Pic
Picture[1].AuroraSheet.Create(8 * 8, 8);

// Definir as cores mais comuns
Picture[1].AuroraSheet.Palette[0] = 0x000000;  //
Picture[1].AuroraSheet.Palette[1] = 0xFFFFFF;  //
Picture[1].AuroraSheet.Palette[2] = 0x0000FF;  //
Picture[1].AuroraSheet.Palette[3] = 0x00FF00;  //
Picture[1].AuroraSheet.Palette[4] = 0xFF0000;  //
Picture[1].AuroraSheet.Palette[5] = 0xFF00FF;  //
Picture[1].AuroraSheet.Palette[6] = 0x00FFFF;  //
Picture[1].AuroraSheet.Palette[7] = 0xFFFF00;  //
Picture[1].AuroraSheet.FlushPalette();

// Exibir as cores (0 ... 7)
```

```
for(v[1] = 0; v[1] < 8; v[1]++)  // Índice de paleta
  for(v[2] = 0; v[2] < 8; v[2]++)  // X
    for(v[3] = 0; v[3] < 8; v[3]++)  // Y
      Picture[1].AuroraSheet.Pixel[v[1] * 8 + v[2
    next;
  next;
next;

// NÃO usar a cor de máscara
Picture[1].AuroraSheet.UseMaskColor = false;

// Exibir a primeira Picture centralizada
Picture[1].X = 160;
Picture[1].Y = 120;
```

## Veja também

`Picture`, `Picture[]`, `Picture[].AuroraSheet`, `Picture[].AuroraSheet.Palette[]`, `Picture[].AuroraSheet.Create()`, `Picture[].AuroraSheet.Pixel[]`, `Picture[].AuroraSheet.UseMaskColor`

MÉTODO

# Método: Picture[].AuroraSheet.Free()

## Sintaxe

```
Picture[Picture-Id].AuroraSheet.Free()
```

## Descrição

Com o método Free é possível esvaziar o conteúdo de um AuroraSheet. Para poder acessar novamente o conteúdo do AuroraSheet depois disso, ele precisa ser criado (`Picture[].AuroraSheet.Create()`), copiado ( `Picture[].AuroraSheet.Clone()`), carregado de um arquivo (`Picture[].AuroraSheet.LoadFromFile()`) ou criado pelo RPG_RT.

> **Informação:** Com o método IsEmpty é possível verificar se um AuroraSheet está vazio.

## Exemplo

```
$

// Esvaziar o conteúdo do AuroraSheet da primeira Pi
Picture[1].AuroraSheet.Free();

// Verificar se o AuroraSheet da primeira Picture
if(Picture[1].AuroraSheet.IsEmpty())  // Condição
  // ...
endif;
```

## Veja também

`Picture`, `Picture[]`, `Picture[].AuroraSheet`, `Picture[].AuroraSheet.Create()`, `Picture[].AuroraSheet.Clone()`, `Picture[].AuroraSheet.IsEmpty()`

PROPRIEDADE

# Propriedade: Picture[].AuroraSheet.Handle

## Sintaxe

```
Picture[Picture-Id].AuroraSheet.Handle
```

## Descrição

Com a propriedade Handle é possível obter o handle de um AuroraSheet. Esse handle é usado com outros métodos de AuroraSheet (por exemplo, para copiar áreas de imagem desse AuroraSheet).

## Exemplo

```
$

// Copiar o Faceset atual para a primeira Picture
// (copia tudo, incluindo a paleta)
Picture[1].AuroraSheet.Clone(Faceset.AuroraSheet.
// Ainda definir as coordenadas da Picture (centro)
Picture[1].X = 160;
Picture[1].Y = 120;
```

## Veja também

`Picture`, `Picture[]`, `Picture[].AuroraSheet`, `Picture[].AuroraSheet.Width`, `Picture[].AuroraSheet.Height`

PROPRIEDADE

# Propriedade: Picture[].AuroraSheet.Height

## Sintaxe

```
Picture[Picture-Id].AuroraSheet.Height
```

## Descrição

Com a propriedade Height é possível obter a altura de um AuroraSheet.

## Exemplo

```
$

// Colorir a primeira Picture de preto
Picture[1].AuroraSheet.Create(80, 80);
Picture[1].AuroraSheet.Palette[1] = 0x000000;  //
Picture[1].AuroraSheet.FlushPalette();
for(v[1] = 0; v[1] < Picture[1].AuroraSheet.Width
  for(v[2] = 0; v[2] < Picture[2].AuroraSheet.Hei
    Picture[1].AuroraSheet.Pixel[v[1], v[2]] = 1;
  next;
```

```
next;

// Ainda definir as coordenadas da Picture (centro)
Picture[1].X = 160;
Picture[1].Y = 120;
```

## Veja também

`Picture`, `Picture[]`, `Picture[].AuroraSheet`, `Picture[].AuroraSheet.Width`, `Picture[].AuroraSheet.Create()`, `Picture[].AuroraSheet.Palette[]`, `Picture[].AuroraSheet.Pixel[]`, `Picture[].AuroraSheet.Handle`

MÉTODO

# Método: Picture[].AuroraSheet.IsEmpty()

## Sintaxe

```
Picture[Picture-Id].AuroraSheet.IsEmpty()
```

## Descrição

Com o método IsEmpty é possível verificar se um AuroraSheet está vazio no momento. Isso é útil para verificar se o acesso a propriedades como `Pixel[]` falharia.

## Exemplo

```
$

// Esvaziar o conteúdo do AuroraSheet da primeira Pi
Picture[1].AuroraSheet.Free();

// Verificar se o AuroraSheet da primeira Picture
if(Picture[1].AuroraSheet.IsEmpty())  // Condição
  // ...
endif;
```

## Veja também

`Picture`, `Picture[]`, `Picture[].AuroraSheet`, `Picture[].AuroraSheet.Create()`, `Picture[].AuroraSheet.Clone()`, `Picture[].AuroraSheet.Free()`

MÉTODO

# Método: Picture[].AuroraSheet.LoadFromFile()

## Sintaxe

```
Picture[Picture-Id].AuroraSheet.LoadFromFile(Nome do arquivo)
```

## Descrição

Com o método LoadFromFile é possível carregar um arquivo no AuroraSheet. O arquivo a carregar pode ser um BMP (Bitmap), um PNG ou um XYZ. Acessos fora da pasta do jogo são inválidos.

## Lista de parâmetros

### Parâmetro: `Nome do arquivo`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** Indica o nome do arquivo (incluindo caminho relativo/extensão). O caminho é relativo à pasta do jogo. Para carregar a imagem, por exemplo, da pasta Picture, é preciso colocar "Picture\\" na frente.

## Exemplo

```
$

// Criar o AuroraSheet e exibi-lo na primeira Pic
Picture[1].AuroraSheet.Create(8 * 8, 8);

// Definir as cores mais comuns
Picture[1].AuroraSheet.Palette[0] = 0x000000;  //
Picture[1].AuroraSheet.Palette[1] = 0xFFFFFF;  //
Picture[1].AuroraSheet.Palette[2] = 0x0000FF;  //
Picture[1].AuroraSheet.Palette[3] = 0x00FF00;  //
Picture[1].AuroraSheet.Palette[4] = 0xFF0000;  //
Picture[1].AuroraSheet.Palette[5] = 0xFF00FF;  //
Picture[1].AuroraSheet.Palette[6] = 0x00FFFF;  //
Picture[1].AuroraSheet.Palette[7] = 0xFFFF00;  //
Picture[1].AuroraSheet.FlushPalette();

// Exibir as cores (0 ... 7)
for(v[1] = 0; v[1] < 8; v[1]++)  // Índice de paleta
  for(v[2] = 0; v[2] < 8; v[2]++)  // X
    for(v[3] = 0; v[3] < 8; v[3]++)  // Y
      Picture[1].AuroraSheet.Pixel[v[1] * 8 + v[2
    next;
  next;
next;

// Salvar o AuroraSheet como bitmap na pasta Pictu
Picture[1].AuroraSheet.SaveAs("Picture\Beispielpa

// Apagar o AuroraSheet
Picture[1].AuroraSheet.Free();

// Carregar o bitmap no AuroraSheet
Picture[1].AuroraSheet.LoadFromFile("Picture\Beis

// NÃO usar a cor de máscara
Picture[1].AuroraSheet.UseMaskColor = false;

// Exibir a primeira Picture centralizada
Picture[1].X = 160;
Picture[1].Y = 120;
```

## Erros possíveis

`DESTINYERROR_NOPERMISSION`, `DESTINYERROR_CANTRESOLVEPATH`, `DESTINYERROR_MEMORY`

## Veja também

`Picture`, `Picture[]`, `Picture[].AuroraSheet`, `Picture[].AuroraSheet.SaveAs()`, `Picture[].AuroraSheet.Palette[]`, `Picture[].AuroraSheet.FlushPalette()`, `Picture[].AuroraSheet.Create()`, `Picture[].AuroraSheet.Pixel[]`, `Picture[].AuroraSheet.UseMaskColor`

OBJETO

# Objeto: Picture[].AuroraSheet.Palette

## Sintaxe

```
Picture[Picture-Id].AuroraSheet.Palette
```

## Descrição

O objeto Palette contém a paleta de um AuroraSheet.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Count` | Propriedade | A propriedade Count sempre devolve 256, já que essa é a quantidade de cores na paleta de um AuroraSheet. |

## Veja também

`Picture`, `Picture[]`, `Picture[].AuroraSheet`, `Picture[].AuroraSheet.Palette[]`, `Picture[].AuroraSheet.FlushPalette()`

PROPRIEDADE

# Propriedade: Picture[].AuroraSheet.Palette.Count

## Sintaxe

```
Picture[Picture-Id].AuroraSheet.Palette.Count
```

## Descrição

A propriedade Count sempre devolve 256, já que essa é a quantidade de cores na paleta de um AuroraSheet.

## Propriedades

- **Tipo de dado:** `dword`

- **Somente leitura:** sim

## Veja também

`Picture`, `Picture[]`, `Picture[].AuroraSheet`, `Picture[].AuroraSheet.Palette`, `Picture[].AuroraSheet.Palette[]`

PROPRIEDADE

# Propriedade: Picture[].AuroraSheet.Palette[]

## Sintaxe

```
Picture[Picture-Id].AuroraSheet.Palette[Índice]
```

## Descrição

Com a propriedade Palette[] é possível ler/gravar uma determinada entrada de paleta. O valor é informado em RGB (vermelho, verde, azul) na notação little-endian. Assim, o último byte representa o vermelho, o penúltimo o verde e o antepenúltimo o azul. O primeiro byte (o mais à esquerda) não é usado e deve ser 0.

> **Informação:** Uma cor costuma ser informada em hexadecimal (por exemplo, 0xFFFFFF para branco). Se você copiar um valor de cor do HTML/CSS (por exemplo, #123456), é preciso corrigir a ordem de leitura para a notação little-endian (ou seja, 0x563412).

> **Informação:** Ao alterar uma entrada de paleta, a alteração só passa a valer depois que o método `FlushPalette` do AuroraSheet for chamado. O motivo técnico é que o RPG_RT guarda a paleta duas vezes, para que alterações de cor (por exemplo, quando a imagem é exibida dessaturada) só precisem ser feitas na paleta efetiva. Normalmente, primeiro se ajusta a paleta como desejado através dessa propriedade e, por fim, executa-se o método `FlushPalette`.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

## Lista de parâmetros

### Parâmetro: `Índice`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 255
- **Descrição:** O índice da entrada de paleta. Esse valor precisa estar na faixa de 0 a 255. As indicações no array `Pixel[]` fazem referência a esse índice.

## Exemplo

```
$

// Criar o AuroraSheet e exibi-lo na primeira Pic
Picture[1].AuroraSheet.Create(8 * 8, 8);

// Definir as cores mais comuns
Picture[1].AuroraSheet.Palette[0] = 0x000000;  //
Picture[1].AuroraSheet.Palette[1] = 0xFFFFFF;  //
Picture[1].AuroraSheet.Palette[2] = 0x0000FF;  //
Picture[1].AuroraSheet.Palette[3] = 0x00FF00;  //
Picture[1].AuroraSheet.Palette[4] = 0xFF0000;  //
Picture[1].AuroraSheet.Palette[5] = 0xFF00FF;  //
Picture[1].AuroraSheet.Palette[6] = 0x00FFFF;  //
Picture[1].AuroraSheet.Palette[7] = 0xFFFF00;  //
Picture[1].AuroraSheet.FlushPalette();

// Exibir as cores (0 ... 7)
for(v[1] = 0; v[1] < 8; v[1]++)  // Índice de paleta
  for(v[2] = 0; v[2] < 8; v[2]++)  // X
    for(v[3] = 0; v[3] < 8; v[3]++)  // Y
      Picture[1].AuroraSheet.Pixel[v[1] * 8 + v[2
    next;
  next;
next;

// NÃO usar a cor de máscara
Picture[1].AuroraSheet.UseMaskColor = false;

// Exibir a primeira Picture centralizada
Picture[1].X = 160;
Picture[1].Y = 120;
```

## Erros possíveis

`DESTINYERROR_PALETTERANGE`

## Veja também

`Picture`, `Picture[]`, `Picture[].AuroraSheet`, `Picture[].AuroraSheet.Palette`, `Picture[].AuroraSheet.FlushPalette()`, `Picture[].AuroraSheet.Create()`, `Picture[].AuroraSheet.Pixel[]`, `Picture[].AuroraSheet.UseMaskColor`

PROPRIEDADE

# Propriedade: Picture[].AuroraSheet.Pixel[]

## Sintaxe

```
Picture[Picture-Id].AuroraSheet.Pixel[X, Y]
```

## Descrição

Com a propriedade Pixel é possível ler/gravar os pixels de um AuroraSheet. O valor de um pixel fica na faixa de 0 a 255 e faz referência à entrada de paleta usada. Como cor de máscara costuma ser usada (dependendo do uso do AuroraSheet) a cor da entrada de paleta 0.

## Exemplo

```
$

// Colorir a primeira Picture de preto
Picture[1].AuroraSheet.Create(80, 80);
Picture[1].AuroraSheet.Palette[1] = 0x000000;  //
Picture[1].AuroraSheet.FlushPalette();
for(v[1] = 0; v[1] < Picture[1].AuroraSheet.Width
  for(v[2] = 0; v[2] < Picture[2].AuroraSheet.Hei
    Picture[1].AuroraSheet.Pixel[v[1], v[2]] = 1;
  next;
next;

// Ainda definir as coordenadas da Picture (centro)
Picture[1].X = 160;
Picture[1].Y = 120;
```

## Erros possíveis

`DESTINYERROR_PICTURE`, `DESTINYERROR_PIXELRANGE`

## Veja também

`Picture`, `Picture[]`, `Picture[].AuroraSheet`, `Picture[].AuroraSheet.Width`, `Picture[].AuroraSheet.Height`, `Picture[].AuroraSheet.Create()`, `Picture[].AuroraSheet.Palette[]`, `Picture[].AuroraSheet.FlushPalette()`, `Picture[].AuroraSheet.Handle`

MÉTODO

# Método: Picture[].AuroraSheet.SaveAs()

## Sintaxe

```
Picture[Picture-Id].AuroraSheet.SaveAs(Nome do arquivo)
```

## Descrição

Com o método SaveAs é possível salvar o conteúdo de um AuroraSheet em um bitmap. Acessos fora da pasta do jogo são inválidos.

## Lista de parâmetros

### Parâmetro: `Nome do arquivo`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** Indica o nome do arquivo (incluindo caminho relativo/extensão). O caminho é relativo à pasta do jogo. Para salvar a imagem, por exemplo, na pasta Picture, é preciso colocar "Picture\\" na frente.

## Exemplo

```
$

// Criar o AuroraSheet e exibi-lo na primeira Pic
Picture[1].AuroraSheet.Create(8 * 8, 8);

// Definir as cores mais comuns
Picture[1].AuroraSheet.Palette[0] = 0x000000;  //
Picture[1].AuroraSheet.Palette[1] = 0xFFFFFF;  //
Picture[1].AuroraSheet.Palette[2] = 0x0000FF;  //
Picture[1].AuroraSheet.Palette[3] = 0x00FF00;  //
Picture[1].AuroraSheet.Palette[4] = 0xFF0000;  //
Picture[1].AuroraSheet.Palette[5] = 0xFF00FF;  //
Picture[1].AuroraSheet.Palette[6] = 0x00FFFF;  //
Picture[1].AuroraSheet.Palette[7] = 0xFFFF00;  //
Picture[1].AuroraSheet.FlushPalette();

// Exibir as cores (0 ... 7)
for(v[1] = 0; v[1] < 8; v[1]++)  // Índice de paleta
  for(v[2] = 0; v[2] < 8; v[2]++)  // X
    for(v[3] = 0; v[3] < 8; v[3]++)  // Y
      Picture[1].AuroraSheet.Pixel[v[1] * 8 + v[2
    next;
  next;
next;

// Salvar o AuroraSheet como bitmap na pasta Pictu
Picture[1].AuroraSheet.SaveAs("Picture\Beispielpa

// Apagar o AuroraSheet
Picture[1].AuroraSheet.Free();

// Carregar o bitmap no AuroraSheet
Picture[1].AuroraSheet.LoadFromFile("Picture\Beis

// NÃO usar a cor de máscara
Picture[1].AuroraSheet.UseMaskColor = false;

// Exibir a primeira Picture centralizada
Picture[1].X = 160;
Picture[1].Y = 120;
```

## Erros possíveis

`DESTINYERROR_CANTOPENFILE`, `DESTINYERROR_NOPERMISSION`, `DESTINYERROR_CANTRESOLVEPATH`, `DESTINYERROR_MEMORY`, `DESTINYERROR_PICTURE`

## Veja também

`Picture`, `Picture[]`, `Picture[].AuroraSheet`, `Picture[].AuroraSheet.LoadFromFile()`, `Picture[].AuroraSheet.Palette[]`, `Picture[].AuroraSheet.FlushPalette()`, `Picture[].AuroraSheet.Create()`, `Picture[].AuroraSheet.Pixel[]`, `Picture[].AuroraSheet.UseMaskColor`

PROPRIEDADE

# Propriedade: Picture[].AuroraSheet.UseMaskColor

## Sintaxe

```
Picture[Picture-Id].AuroraSheet.UseMaskColor
```

## Descrição

Com a propriedade UseMaskColor é possível ler/gravar se o RPG_RT, ao desenhar o AuroraSheet, usa a entrada de paleta 0 como cor de máscara (true) ou não (false). Esse valor não é respeitado pelo RPG_RT em todos os objetos AuroraSheet. Se esse valor é respeitado, e qual é o comportamento padrão, pode ser visto na tabela a seguir:

Objeto AuroraSheet

Funciona?

Comportamento

não `Event[]` / (O valor desta propriedade é ignorado; em vez disso, a cor de máscara `MapEvent[]` é sempre usada)

`Faceset` sim

parcialmente (o RPG_RT verifica apenas uma vez se um chip é transparente. `Map` Se isso for alterado depois, o panorama por baixo deixa de ser desenhado)

sim (desde que a ordem de blitting não tenha sido alterada; `Panorama` isso, porém, pode ser ignorado, pois o panorama é desenhado primeiro de qualquer forma)

`Picture[]` sim

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** não

## Exemplo

```
$

// Criar o AuroraSheet e exibi-lo na primeira Pic
Picture[1].AuroraSheet.Create(100, 100);

// Definir as cores branco (índice = 0) e preto (índice
Picture[1].AuroraSheet.Palette[0] = 0xFFFFFF;  //
Picture[1].AuroraSheet.Palette[1] = 0x000000;  //
Picture[1].AuroraSheet.FlushPalette();

// Colorir o conteúdo do AuroraSheet de branco
Picture[1].AuroraSheet.FillRect(0, 0, 100, 100, 0

// Desenhar um quadrado preenchido de preto
Picture[1].AuroraSheet.FillRect(10, 10, 90, 90, 1

// Usar a cor de máscara (a cor 0 (aqui, branco)
Picture[1].AuroraSheet.UseMaskColor = true;
```

```
// Exibir a primeira Picture centralizada
Picture[1].X = 160;
Picture[1].Y = 120;
```

## Veja também

`Picture`, `Picture[]`, `Picture[].AuroraSheet`, `Picture[].AuroraSheet.FillRect()`, `Picture[].AuroraSheet.Create()`, `Picture[].AuroraSheet.Palette[]`, `Picture[].AuroraSheet.FlushPalette()`

PROPRIEDADE

# Propriedade: Picture[].AuroraSheet.Width

## Sintaxe

```
Picture[Picture-Id].AuroraSheet.Width
```

## Descrição

Com a propriedade Width é possível obter a largura de um AuroraSheet.

## Exemplo

```
$

// Colorir a primeira Picture de preto
Picture[1].AuroraSheet.Create(80, 80);
Picture[1].AuroraSheet.Palette[1] = 0x000000;  //
Picture[1].AuroraSheet.FlushPalette();
for(v[1] = 0; v[1] < Picture[1].AuroraSheet.Width
  for(v[2] = 0; v[2] < Picture[2].AuroraSheet.Hei
    Picture[1].AuroraSheet.Pixel[v[1], v[2]] = 1;
  next;
next;

// Ainda definir as coordenadas da Picture (centro)
Picture[1].X = 160;
Picture[1].Y = 120;
```

## Veja também

`Picture`, `Picture[]`, `Picture[].AuroraSheet`, `Picture[].AuroraSheet.Height`, `Picture[].AuroraSheet.Create()`, `Picture[].AuroraSheet.Palette[]`, `Picture[].AuroraSheet.Pixel[]`, `Picture[].AuroraSheet.Handle`

CATEGORIA

# Categoria: Funções de desenho

## Descrição

Nesta categoria estão reunidas todas as funções de desenho dos objetos AuroraSheet.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `BltRect()` | Método | Com o método BltRect, uma área de imagem de um AuroraSheet é copiada para uma determinada posição deste AuroraSheet, sem copiar junto uma determinada cor. |
| `CopyRect()` | Método | Com o método CopyRect, uma área de imagem de um AuroraSheet é copiada para uma determinada posição deste AuroraSheet. |
| `DrawLine()` | Método | Com o método DrawLine é desenhada uma linha reta entre dois pontos. |
| `FillRect()` | Método | Com o método FillRect é desenhado um retângulo preenchido. |

## Veja também

`Picture`, `Picture[]`, `Picture[].AuroraSheet`

MÉTODO

# Método: Picture[].AuroraSheet.BltRect()

## Sintaxe

```
Picture[Picture-Id].AuroraSheet.BltRect(X, Y, Quell-
AuroraSheet, Links, Oben, Rechts, Unten, Maskenfarbe)
```

## Descrição

Com o método BltRect, uma área de imagem de um AuroraSheet é copiada para uma determinada posição deste AuroraSheet, sem copiar junto uma determinada cor.

> **Informação:** A área de imagem copiada também pode vir do mesmo AuroraSheet, mas nesse caso as coordenadas não podem se sobrepor.

## Lista de parâmetros

### Parâmetro: `X`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... (Picture[].AuroraSheet.Width - 1)
- **Descrição:** A coordenada X no AuroraSheet onde a área de imagem será inserida. A largura da área de origem (Direita - Esquerda) precisa estar disponível.

### Parâmetro: `Y`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... (Picture[].AuroraSheet.Height - 1)
- **Descrição:** A coordenada Y no AuroraSheet onde a área de imagem será inserida. A altura da área de origem (Baixo - Cima) precisa estar disponível.

### Parâmetro: `AuroraSheet de origem`

- **Tipo de dado:** `handle`
- **Opcional:** não
- **Descrição:** O AuroraSheet de origem, do qual a imagem deve ser copiada.

### Parâmetro: `Esquerda`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... (Picture[].AuroraSheet.Width - 1)
- **Descrição:** A coordenada X do ponto superior esquerdo do retângulo do AuroraSheet de origem. Precisa estar na faixa de 0 (totalmente à esquerda) a (`Picture[].AuroraSheet.Width` - 1) (totalmente à direita).

### Parâmetro: `Cima`

- **Tipo de dado:** `dword`
- **Opcional:** nein Wertebereich: 0 ... (Picture[].AuroraSheet.Height - 1)
- **Descrição:** A coordenada Y do ponto superior esquerdo do retângulo do AuroraSheet de origem. Precisa estar na faixa de 0 (totalmente em cima) a (`Picture[].AuroraSheet.Height` - 1) (totalmente embaixo).

### Parâmetro: `Direita`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... (Picture[].AuroraSheet.Width)
- **Descrição:** A coordenada X do ponto inferior direito do AuroraSheet de origem. Precisa estar na faixa de 0 (totalmente à esquerda) a (`Picture[].AuroraSheet.Width`) (totalmente à direita + 1) e ser maior/igual a Esquerda. Essa coordenada é a soma da coordenada esquerda com a largura da área de imagem a copiar.

### Parâmetro: `Baixo`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... (Picture[].AuroraSheet.Height)
- **Descrição:** A coordenada Y do ponto inferior direito do AuroraSheet de origem. Precisa estar na faixa de 0 (totalmente em cima) a (`Picture[].AuroraSheet.Height`) (totalmente embaixo + 1) e ser maior/igual a Cima. Essa coordenada é a soma da coordenada de cima com a altura da área de imagem a copiar.

### Parâmetro: `Cor da máscara`

- **Tipo de dado:** `byte`
- **Opcional:** não
- **Descrição:** Indica o índice de paleta da cor a ser ignorada (= cor de máscara). Todos os pixels da área de imagem a copiar que usam essa entrada de paleta não são copiados.

## Exemplo

```
$

// Criar o AuroraSheet e exibi-lo na primeira Pic
Picture[1].AuroraSheet.Create(100, 100);

// As cores branco (índice = 1) e preto (índice
```

```
Picture[1].AuroraSheet.Palette[1] = 0xFFFFFF;  //
Picture[1].AuroraSheet.Palette[2] = 0x000000;  //
Picture[1].AuroraSheet.FlushPalette();

// Colorir o conteúdo do AuroraSheet de branco
Picture[1].AuroraSheet.FillRect(0, 0, 100, 100, 1

// Desenhar um quadrado preenchido de preto
Picture[1].AuroraSheet.FillRect(10, 10, 90, 90, 2

// Copiar o quarto superior esquerdo para o quarto
Picture[1].AuroraSheet.BltRect(75, 75, Picture[1]

// Exibir a primeira Picture centralizada
Picture[1].X = 160;
Picture[1].Y = 120;
```

## Erros possíveis

`DESTINYERROR_PICTURE`, `DESTINYERROR_PIXELRANGE`, `DESTINYERROR_SAMEPICTURE`, `DESTINYERROR_INVALIDHANDLE`

## Veja também

`Picture`, `Picture[]`, `Picture[].AuroraSheet`, `Picture[].AuroraSheet.CopyRect()`, `Picture[].AuroraSheet.FillRect()`, `Picture[].AuroraSheet.Create()`, `Picture[].AuroraSheet.Palette[]`, `Picture[].AuroraSheet.FlushPalette()`

MÉTODO

# Método: Picture[].AuroraSheet.CopyRect()

## Sintaxe

```
Picture[Picture-Id].AuroraSheet.CopyRect(X, Y, Quell-
AuroraSheet, Links, Oben, Rechts, Unten)
```

## Descrição

Com o método CopyRect, uma área de imagem de um AuroraSheet é copiada para uma determinada posição deste AuroraSheet.

> **Informação:** A área de imagem copiada também pode vir do mesmo AuroraSheet, mas nesse caso as coordenadas não podem se sobrepor.

## Lista de parâmetros

### Parâmetro: `X`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... (Picture[].AuroraSheet.Width - 1)

- **Descrição:** A coordenada X no AuroraSheet onde a área de imagem será inserida. A largura da área de origem (Direita - Esquerda) precisa estar disponível.

### Parâmetro: `Y`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... (Picture[].AuroraSheet.Height - 1)
- **Descrição:** A coordenada Y no AuroraSheet onde a área de imagem será inserida. A altura da área de origem (Baixo - Cima) precisa estar disponível.

### Parâmetro: `AuroraSheet de origem`

- **Tipo de dado:** `handle`
- **Opcional:** não
- **Descrição:** O AuroraSheet de origem, do qual a imagem deve ser copiada.

### Parâmetro: `Esquerda`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... (Picture[].AuroraSheet.Width - 1)
- **Descrição:** A coordenada X do ponto superior esquerdo do retângulo do AuroraSheet de origem. Precisa estar na faixa de 0 (totalmente à esquerda) a (`Picture[].AuroraSheet.Width` - 1) (totalmente à direita).

### Parâmetro: `Cima`

- **Tipo de dado:** `dword`
- **Opcional:** nein Wertebereich: 0 ... (Picture[].AuroraSheet.Height - 1)
- **Descrição:** A coordenada Y do ponto superior esquerdo do retângulo do AuroraSheet de origem. Precisa estar na faixa de 0 (totalmente em cima) a (`Picture[].AuroraSheet.Height` - 1) (totalmente embaixo).

### Parâmetro: `Direita`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... (Picture[].AuroraSheet.Width)
- **Descrição:** A coordenada X do ponto inferior direito do AuroraSheet de origem. Precisa estar na faixa de 0 (totalmente à esquerda) a (`Picture[].AuroraSheet.Width`) (totalmente à direita + 1) e ser maior/igual a Esquerda. Essa coordenada é a soma da coordenada esquerda com a largura da área de imagem a copiar.

### Parâmetro: `Baixo`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... (Picture[].AuroraSheet.Height)
- **Descrição:** A coordenada Y do ponto inferior direito do AuroraSheet de origem. Precisa estar na faixa de 0 (totalmente em cima) a (`Picture[].AuroraSheet.Height`) (totalmente embaixo + 1) e ser maior/igual a Cima. Essa coordenada é a soma da coordenada de cima com a altura da área de imagem a copiar.

## Exemplo

```
$

// Criar o AuroraSheet e exibi-lo na primeira Pic
Picture[1].AuroraSheet.Create(100, 100);

// As cores branco (índice = 1) e preto (índice
Picture[1].AuroraSheet.Palette[1] = 0xFFFFFF;  //
Picture[1].AuroraSheet.Palette[2] = 0x000000;  //
Picture[1].AuroraSheet.FlushPalette();

// Colorir o conteúdo do AuroraSheet de branco
Picture[1].AuroraSheet.FillRect(0, 0, 100, 100, 1

// Desenhar um quadrado preenchido de preto
Picture[1].AuroraSheet.FillRect(10, 10, 90, 90, 2

// Copiar o quarto superior esquerdo para o quarto
Picture[1].AuroraSheet.CopyRect(75, 75, Picture[1

// Exibir a primeira Picture centralizada
Picture[1].X = 160;
Picture[1].Y = 120;
```

## Erros possíveis

`DESTINYERROR_PICTURE`, `DESTINYERROR_PIXELRANGE`, `DESTINYERROR_SAMEPICTURE`, `DESTINYERROR_INVALIDHANDLE`

## Veja também

`Picture`, `Picture[]`, `Picture[].AuroraSheet`, `Picture[].AuroraSheet.BltRect()`, `Picture[].AuroraSheet.FillRect()`, `Picture[].AuroraSheet.Create()`, `Picture[].AuroraSheet.Palette[]`, `Picture[].AuroraSheet.FlushPalette()`

MÉTODO

# Método: Picture[].AuroraSheet.DrawLine()

## Sintaxe

```
Picture[Picture-Id].AuroraSheet.DrawLine(X1, Y1, X2,
Y2, Farbe)
```

## Descrição

Com o método DrawLine é desenhada uma linha reta entre dois pontos. Ambos os pontos recebem sempre a cor indicada.

## Lista de parâmetros

### Parâmetro: `X1`

- **Tipo de dado:** `dword`

- **Opcional:** não
- **Faixa de valores:** 0 ... (Picture[].AuroraSheet.Width - 1)
- **Descrição:** A coordenada X do primeiro ponto. Precisa estar na faixa de 0 (totalmente à esquerda) a ( `Picture[].AuroraSheet.Width` - 1) (totalmente à direita).

### Parâmetro: `Y1`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... (Picture[].AuroraSheet.Height - 1)
- **Descrição:** A coordenada Y do primeiro ponto. Precisa estar na faixa de 0 (totalmente em cima) a ( `Picture[].AuroraSheet.Height` - 1) (totalmente embaixo).

### Parâmetro: `X2`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... (Picture[].AuroraSheet.Width - 1)
- **Descrição:** A coordenada X do segundo ponto. Precisa estar na faixa de 0 (totalmente à esquerda) a ( `Picture[].AuroraSheet.Width` - 1) (totalmente à direita).

### Parâmetro: `Y2`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... (Picture[].AuroraSheet.Height - 1)
- **Descrição:** A coordenada Y do segundo ponto. Precisa estar na faixa de 0 (totalmente em cima) a ( `Picture[].AuroraSheet.Height` - 1) (totalmente embaixo).

### Parâmetro: `Cor`

- **Tipo de dado:** `byte Optional:nein`
- **Descrição:** O índice da entrada de paleta usada (= a cor usada). A cor em si pode ser definida através de `Picture[].AuroraSheet.Palette[]`.

## Exemplo

```
$

// Criar o AuroraSheet e exibi-lo na primeira Pic
Picture[1].AuroraSheet.Create(100, 100);

// As cores branco (índice = 1) e vermelho (índice = 2
Picture[1].AuroraSheet.Palette[1] = 0xFFFFFF;  //
Picture[1].AuroraSheet.Palette[2] = 0x0000FF;  //
Picture[1].AuroraSheet.FlushPalette();

// Colorir o conteúdo do AuroraSheet de branco
Picture[1].AuroraSheet.FillRect(0, 0,
                                Picture[1].Aurora
                                1);

// Desenhar um quadrado vermelho
Picture[1].AuroraSheet.DrawLine(5, 5,
                                Picture[1].Aurora
                                2);
```

```
Picture[1].AuroraSheet.DrawLine(Picture[1].Aurora
                                Picture[1].Aurora
                                2);
Picture[1].AuroraSheet.DrawLine(Picture[1].Aurora
                                5, Picture[1].Aur
                                2);
Picture[1].AuroraSheet.DrawLine(5, Picture[1].Aur
                                5, 5,
                                2);

// Exibir a primeira Picture centralizada
Picture[1].X = 160;
Picture[1].Y = 120;
```

## Erros possíveis

`DESTINYERROR_PICTURE`, `DESTINYERROR_PIXELRANGE`

## Veja também

`Picture`, `Picture[]`, `Picture[].AuroraSheet`, `Picture[].AuroraSheet.FillRect()`, `Picture[].AuroraSheet.Create()`, `Picture[].AuroraSheet.Palette[]`, `Picture[].AuroraSheet.FlushPalette()`

MÉTODO

# Método: Picture[].AuroraSheet.FillRect()

## Sintaxe

```
Picture[Picture-Id].AuroraSheet.FillRect(Links, Oben,
Rechts, Unten, Farbe)
```

## Descrição

Com o método FillRect é desenhado um retângulo preenchido. Esse retângulo começa em Esquerda/Cima e termina em (Direita - 1)/(Baixo - 1).

## Lista de parâmetros

### Parâmetro: `Esquerda`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... (Picture[].AuroraSheet.Width - 1)
- **Descrição:** A coordenada X do ponto superior esquerdo do retângulo. Precisa estar na faixa de 0 (totalmente à esquerda) a (`Picture[].AuroraSheet.Width` - 1) (totalmente à direita).

### Parâmetro: `Cima`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... (Picture[].AuroraSheet.Height - 1)

- **Descrição:** A coordenada Y do ponto superior esquerdo do retângulo. Precisa estar na faixa de 0 (totalmente em cima) a (`Picture[].AuroraSheet.Height` - 1) (totalmente embaixo).

### Parâmetro: `Direita`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... (Picture[].AuroraSheet.Width)
- **Descrição:** A coordenada X do ponto inferior direito. Precisa estar na faixa de 0 (totalmente à esquerda) a (`Picture[].AuroraSheet.Width`) (totalmente à direita + 1) e ser maior/igual a Esquerda. Essa coordenada é a soma da coordenada esquerda com a largura do retângulo. A coordenada aqui indicada não é colorida.

### Parâmetro: `Baixo`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... (Picture[].AuroraSheet.Height)
- **Descrição:** A coordenada Y do ponto inferior direito. Precisa estar na faixa de 0 (totalmente em cima) a

(`Picture[].AuroraSheet.Height`) (totalmente embaixo + 1) e ser maior/igual a Cima. Essa coordenada é a soma da coordenada de cima com a altura do retângulo. A coordenada aqui indicada não é colorida.

### Parâmetro: `Cor`

- **Tipo de dado:** `byte`
- **Opcional:** não
- **Descrição:** O índice da entrada de paleta usada (= a cor usada). A cor em si pode ser definida através de `Picture[].AuroraSheet.Palette[]`.

## Exemplo

```
$

// Criar o AuroraSheet e exibi-lo na primeira Pic
Picture[1].AuroraSheet.Create(100, 100);

// As cores branco (índice = 1) e vermelho (índice = 2
Picture[1].AuroraSheet.Palette[1] = 0xFFFFFF;  //
Picture[1].AuroraSheet.Palette[2] = 0x0000FF;  //
Picture[1].AuroraSheet.FlushPalette();

// Colorir o conteúdo do AuroraSheet de branco
Picture[1].AuroraSheet.FillRect(0, 0,
                                Picture[1].Aurora
                                1);

// Desenhar um quadrado vermelho
Picture[1].AuroraSheet.DrawLine(5, 5,
                                Picture[1].Aurora
                                2);
Picture[1].AuroraSheet.DrawLine(Picture[1].Aurora
                                Picture[1].Aurora
                                2);
Picture[1].AuroraSheet.DrawLine(Picture[1].Aurora
                                5, Picture[1].Aur
                                2);
Picture[1].AuroraSheet.DrawLine(5, Picture[1].Aur
                                5, 5,
```

```
                                2);

// Exibir a primeira Picture centralizada
Picture[1].X = 160;
Picture[1].Y = 120;
```

## Erros possíveis

`DESTINYERROR_PICTURE`, `DESTINYERROR_PIXELRANGE`

## Veja também

`Picture`, `Picture[]`, `Picture[].AuroraSheet`, `Picture[].AuroraSheet.DrawLine()`, `Picture[].AuroraSheet.Create()`, `Picture[].AuroraSheet.Palette[]`, `Picture[].AuroraSheet.FlushPalette()`

MÉTODO

# Método: Picture[].BltRect()

## Sintaxe

```
Picture[Picture-Id].BltRect(X, Y, QuellPicture-Id,
Links, Oben, Rechts, Unten, Maskenfarbe)
```

## Descrição

Com o método BltRect, uma área de imagem de outra Picture pode ser copiada para esta Picture, sem copiar junto a cor de máscara indicada. Este método só foi mantido por compatibilidade com o DestinyPatch V1.0. Por isso, recomenda-se usar o método `Picture[].AuroraSheet.BltRect()`, já que ele funciona com todos os objetos AuroraSheet. A diferença deste método para o método padronizado do AuroraSheet é que, em vez de um handle de AuroraSheet, é passada a Picture-Id da Picture de origem. Para a descrição deste método, veja portanto a descrição do método `BltRect()` do subobjeto `AuroraSheet`.

## Propriedades

- **Obsoleto:** sim

## Lista de parâmetros

### Parâmetro: `X`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A coordenada X do canto superior esquerdo, para onde a área de imagem deve ser copiada nesta Picture.

### Parâmetro: `Y`

- **Tipo de dado:** `dword`
- **Opcional:** não

- **Descrição:** A coordenada Y do canto superior esquerdo, para onde a área de imagem deve ser copiada nesta Picture.

### Parâmetro: `Picture-Id de origem`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A Picture-Id da Picture da qual a área de imagem deve ser copiada.

### Parâmetro: `Esquerda`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A coordenada esquerda da área de imagem a copiar da Picture de origem.

### Parâmetro: `Cima`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A coordenada de cima da área de imagem a copiar da Picture de origem.

### Parâmetro: `Direita`

- **Tipo de dado:** `dword`
- **Opcional:** nein Beschreibung: Die rechte Koordinate des zu kopierenden Bildbereichs aus dem Quell-Picture.

### Parâmetro: `Baixo`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A coordenada de baixo da área de imagem a copiar da Picture de origem.

### Parâmetro: `Cor da máscara`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** O índice de paleta da cor que não deve ser copiada.

## Exemplo

```
$
// Copiar uma área de imagem de 32 x 32 pixels da
Picture[1].BltRect(0, 0, 2, 0, 0, 32, 32, 255);
// Chamada totalmente equivalente via AuroraSh
Picture[1].AuroraSheet.BltRect(0, 0, Picture[2].A
```

## Erros possíveis

`DESTINYERROR_PICTURE`, `DESTINYERROR_PIXELRANGE`, `DESTINYERROR_SAMEPICTURE`

## Veja também

`Picture`, `Picture[]`, `Picture[].AuroraSheet.CopyRect()`, `Picture[].BltRect()`, `Picture[].AuroraSheet.BltRect()`

PROPRIEDADE

# Propriedade: Picture[].Blue

## Sintaxe

```
Picture[Picture-Id].Blue
```

## Descrição

Com a propriedade Blue é possível ler/gravar a coloração azul de uma Picture, em porcentagem. O valor 0 significa que a Picture não apresenta a cor azul. O valor 100 significa que a Picture exibe o azul normalmente. O valor 200 significa que a Picture é exibida com tons de azul especialmente fortes.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não
- **Faixa de valores:** 0 ... 200

## Exemplo

```
$
// Obter a coloração azul da primeira Picture
v[1] = Picture[1].Blue;

// Reduzir em 1% a coloração azul da primeira Pic
if(Picture[1].Blue > 0)
  Picture[1].Blue--;
endif;
```

## Veja também

`Picture`, `Picture[]`, `Picture[].Red`, `Picture[].Green`, `Picture[].Chroma`

PROPRIEDADE

# Propriedade: Picture[].Chroma

## Sintaxe

```
Picture[Picture-Id].Chroma
```

## Descrição

Com a propriedade Chroma é possível ler/gravar a crominância (saturação de cor) de uma Picture, em porcentagem. O valor 0 significa que a Picture é exibida dessaturada. O valor 100 significa que a Picture é exibida normalmente. O valor 200 significa que a Picture é exibida com diferenças de cor especialmente grandes.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não
- **Faixa de valores:** 0 ... 200

## Exemplo

```
$
// Obter a crominância da primeira Picture
v[1] = Picture[1].Chroma;

// Reduzir em 1% a crominância da primeira Pic
if(Picture[1].Chroma > 0)
  Picture[1].Chroma--;
endif;
```

## Veja também

`Picture`, `Picture[]`, `Picture[].Red`, `Picture[].Green`, `Picture[].Blue`

MÉTODO

# Método: Picture[].CopyRect()

## Sintaxe

```
Picture[Picture-Id].CopyRect(X, Y, QuellPicture-Id,
Links, Oben, Rechts, Unten)
```

## Descrição

Com o método CopyRect, uma área de imagem de outra Picture pode ser copiada para esta Picture. Este método só foi mantido por compatibilidade com o DestinyPatch V1.0. Por isso, recomenda-se usar o método `Picture[].AuroraSheet.CopyRect()`, já que ele funciona com todos os objetos AuroraSheet. A diferença deste método para o método padronizado do AuroraSheet é que, em vez de um handle de AuroraSheet, é passada a Picture-Id da Picture de origem. Para a descrição deste método, veja portanto a descrição do método `CopyRect()` do subobjeto `AuroraSheet`.

## Propriedades

- **Obsoleto:** sim

## Lista de parâmetros

### Parâmetro: `X`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A coordenada X do canto superior esquerdo, para onde a área de imagem deve ser copiada nesta Picture.

### Parâmetro: `Y`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A coordenada Y do canto superior esquerdo, para onde a área de imagem deve ser copiada nesta Picture.

### Parâmetro: `Picture-Id de origem`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A Picture-Id da Picture da qual a área de imagem deve ser copiada.

### Parâmetro: `Esquerda`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A coordenada esquerda da área de imagem a copiar da Picture de origem.

### Parâmetro: `Cima`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A coordenada de cima da área de imagem a copiar da Picture de origem.

### Parâmetro: `Direita`

- **Tipo de dado:** `dword`
- **Opcional:** nein Beschreibung: Die rechte Koordinate des zu kopierenden Bildbereichs aus dem Quell-Picture.

### Parâmetro: `Baixo`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Descrição:** A coordenada de baixo da área de imagem a copiar da Picture de origem.

## Exemplo

```
$
// Copiar uma área de imagem de 32 x 32 pixels da
Picture[1].CopyRect(0, 0, 2, 0, 0, 32, 32);
// Chamada totalmente equivalente via AuroraSh
Picture[1].AuroraSheet.CopyRect(0, 0, Picture[2].
```

## Erros possíveis

`DESTINYERROR_PICTURE`, `DESTINYERROR_PIXELRANGE`, `DESTINYERROR_SAMEPICTURE`

## Veja também

`Picture`, `Picture[]`, `Picture[].AuroraSheet.CopyRect()`, `Picture[].BltRect()`, `Picture[].AuroraSheet.BltRect()`

MÉTODO

## Método: Picture[].DrawLine()

### Sintaxe

```
Picture[Picture-Id].DrawLine(X1, Y1, X2, Y2,
Farbe)
```

### Descrição

Com o método DrawLine é possível desenhar uma linha nesta Picture. Este método só foi mantido por compatibilidade com o DestinyPatch V1.0. Por isso, recomenda-se usar o método `Picture[].AuroraSheet.DrawLine()`, já que ele está disponível em todos os objetos AuroraSheet. A documentação deste método também se encontra lá.

PROPRIEDADE

## Propriedade: Picture[].Filename

### Sintaxe

```
Picture[Picture-Id].Filename
```

### Descrição

Com a propriedade Filename é possível obter o nome do arquivo da Picture. O valor dessa propriedade é definido pelo comando ShowPicture.

### Exemplo

```
$
// Carregar o gráfico de nuvem do RTP como Picture 1 e exi
Command.ShowPicture(1, "CLOUD", 160, 120);

// Consultar o nome do arquivo da primeira Picture e
a[1] = Picture[1].Filename;  // "CLOUD"
```

### Veja também

`Picture`, `Picture[]`, `Command.ShowPicture()`

MÉTODO

## Método: Picture[].FillRect()

### Sintaxe

```
Picture[Picture-Id].FillRect(Links, Oben,
Rechts, Unten, Farbe)
```

## Descrição

Com o método FillRect é desenhado um retângulo preenchido. Esse retângulo começa em Esquerda/Cima e termina em (Direita - 1)/(Baixo - 1). Este método só foi mantido por compatibilidade com o DestinyPatch V1.0. Por isso, recomenda-se usar o método `Picture[].AuroraSheet.FillRect()`, já que ele está disponível em todos os objetos AuroraSheet. A documentação deste método também se encontra lá.

MÉTODO

# Método: Picture[].FlushPalette()

## Sintaxe

```
Picture[Picture-Id].FlushPalette()
```

## Descrição

Com o método FlushPalette é possível fazer com que a paleta atual passe a ser usada como a paleta efetiva. Este método só foi mantido por compatibilidade com o DestinyPatch V1.0. Por isso, recomenda-se usar o método `Picture[].AuroraSheet.FlushPalette()`, já que ele está disponível em todos os objetos AuroraSheet. A documentação deste método também se encontra lá.

PROPRIEDADE

# Propriedade: Picture[].Green

## Sintaxe

```
Picture[Picture-Id].Green
```

## Descrição

Com a propriedade Green é possível ler/gravar a coloração verde de uma Picture, em porcentagem. O valor 0 significa que a Picture não apresenta a cor verde. O valor 100 significa que a Picture exibe o verde normalmente. O valor 200 significa que a Picture é exibida com tons de verde especialmente fortes.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não
- **Faixa de valores:** 0 ... 200

## Exemplo

```
$
// Obter a coloração verde da primeira Picture
v[1] = Picture[1].Green;

// Reduzir em 1% a coloração verde da primeira Pic
if(Picture[1].Green > 0)
```

```
  Picture[1].Green--;
endif;
```

## Veja também

`Picture`, `Picture[]`, `Picture[].Red`, `Picture[].Blue`, `Picture[].Chroma`

PROPRIEDADE

# Propriedade: Picture[].Height

## Sintaxe

```
Picture[Picture-Id].Height
```

## Descrição

Com a propriedade Height é possível obter a altura de uma Picture em pixels. Se no momento nenhuma imagem estiver carregada na Picture, esse valor é 0.

> **Informação:** Essa propriedade existe apenas por compatibilidade com o DestinyPatch V1.0. Seu valor é idêntico ao da propriedade `Height` do subobjeto `AuroraSheet` (veja o exemplo).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Obsoleto:** sim

## Exemplo

```
$
// Obter a altura da primeira Picture e guardar em
v[1] = Picture[1].Height;

// Obter a altura da primeira Picture pelo AuroraSh
v[2] = Picture[1].AuroraSheet.Height;
```

## Veja também

`Picture`, `Picture[]`, `Picture[].Width`, `Picture[].AuroraSheet.Height`

MÉTODO

# Método: Picture[].LoadFromFile()

## Sintaxe

```
Picture[Picture-Id].LoadFromFile(Nome do arquivo)
```

## Descrição

Com o método LoadFromFile é possível carregar uma Picture a partir de um arquivo de imagem. Todas as demais configurações da Picture que não têm relação com a imagem são mantidas.

## Lista de parâmetros

### Parâmetro: `Nome do arquivo`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** Indica o nome do arquivo da Picture (sem caminho/extensão). Esse caminho é relativo, e por isso também pode, em princípio, ser usado para carregar imagens de fora da pasta Picture. Se for necessário carregar um arquivo com uma extensão específica, pode-se informar a extensão incluindo o caractere NULL, para carregar explicitamente esse arquivo (veja o exemplo).

## Exemplo

```
$
// Primeira chamada com Picture[].LoadFromFile()
Picture[1].LoadFromFile("Bild");
// Segunda chamada (totalmente idêntica) com Co
Command.ShowPicture(1, "Bild", Picture[1].X, Pict
                    Picture[1].Magnification, Pic
                    Picture[1].Red, Picture[1].Gr
                    Picture[1].Action, Picture[1]
// Terceira chamada com Picture[].AuroraSheet.LoadF
// Aqui nenhuma propriedade é redefinida
Picture[1].AuroraSheet.LoadFromFile("Picture\Dawn
```

Para carregar uma Picture bem específica, pode-se anexar um caractere NULL ao fim do nome do arquivo:

```
$
// Carregar o arquivo Picture\Bild.bmp
Picture[1].LoadFromFile("Bild.bmp" .. String.Chr(
// Carregar o arquivo Panorama\Bild.bmp (outro dire
Picture[1].LoadFromFile("..\Panorama\Bild.bmp" ..
```

## Erros possíveis

`DESTINYERROR_EXCEPTION`

## Veja também

`Picture`, `Picture[]`, `Command.ShowPicture()`, `Picture[].AuroraSheet.LoadFromFile()`, `Picture[].SaveAs()`

PROPRIEDADE

# Propriedade: Picture[].Magnification

## Sintaxe

```
Picture[Picture-Id].Magnification
```

## Descrição

Com a propriedade Magnification é possível ler/gravar o fator de ampliação de uma Picture, em porcentagem.

> **Atenção:** Se essa propriedade for definida como 0, isso pode causar uma travada do RPG_RT. Esse não é um erro do DestinyPatch, e sim do RPG_RT.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não
- **Faixa de valores:** >= 1

## Exemplo

```
$
// Obter o fator de ampliação da primeira Picture
v[1] = Picture[1].Magnification;

// Aumentar em 1 o fator de ampliação da primeira Pic
Picture[1].Magnification++;
```

## Veja também

`Picture`, `Picture[]`, `Picture[].Transparency`

PROPRIEDADE

# Propriedade: Picture[].MapMove

## Sintaxe

```
Picture[Picture-Id].MapMove
```

## Descrição

Com a propriedade MapMove é possível ler/gravar a propriedade MoveWithMap de uma Picture. Se esse valor for true, a Picture se move junto com o mapa. Se esse valor for false, a Picture permanece na posição da tela, mesmo que o mapa role.

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** não

## Exemplo

```
$
// Vincular a primeira Picture ao mapa
```

```
Picture[1].MapMove = true;
```

## Veja também

`Picture`, `Picture[]`, `Picture[].MapX`, `Picture[].MapY`

PROPRIEDADE

# Propriedade: Picture[].MapX

## Sintaxe

```
Picture[Picture-Id].MapX
```

## Descrição

Com a propriedade MapX é possível ler/gravar a coordenada X do ponto de referência de uma Picture que se move com o mapa. Esse é o valor usado no comando ShowPicture para exibir a imagem originalmente.

> **Informação:** Se o movimento de Picture for como no RPG Maker 2003, o valor dessa propriedade é praticamente sem importância.

## Propriedades

- **Tipo de dado:** `float`
- **Somente leitura:** não

## Exemplo

```
$
// Obter as coordenadas X e Y originais do primeir
// comando ShowPicture e guardar em v[1] e v
v[1] = Picture[1].MapX;
v[2] = Picture[1].MapY;
```

## Veja também

`Picture`, `Picture[]`, `Picture[].MapY`, `Picture[].MapMove`

PROPRIEDADE

# Propriedade: Picture[].MapY

## Sintaxe

```
Picture[Picture-Id].MapY
```

## Descrição

Com a propriedade MapY é possível ler/gravar a coordenada Y do ponto de referência de uma Picture que se move com o mapa. Esse é o valor usado no comando ShowPicture para exibir a imagem originalmente.

> **Informação:** Se o movimento de Picture for como no RPG Maker 2003, o valor dessa propriedade é praticamente sem importância.

## Propriedades

- **Tipo de dado:** `float`
- **Somente leitura:** não

## Exemplo

```
$
// Obter as coordenadas X e Y originais do primeir
// comando ShowPicture e guardar em v[1] e v
v[1] = Picture[1].MapX;
v[2] = Picture[1].MapY;
```

## Veja também

`Picture`, `Picture[]`, `Picture[].MapX`, `Picture[].MapMove`

PROPRIEDADE

# Propriedade: Picture[].MoveX

## Sintaxe

```
Picture[Picture-Id].MoveX
```

## Descrição

Com a propriedade MoveX é possível ler/gravar a coordenada X atual de uma Picture durante o movimento.

## Propriedades

- **Tipo de dado:** `float`
- **Somente leitura:** não

## Exemplo

```
$
// Obter as coordenadas X e Y atuais da primeira
v[1] = Picture[1].MoveX;
v[2] = Picture[1].MoveY;
```

## Veja também

`Picture`, `Picture[]`, `Picture[].MoveY`, `Picture[].X`, `Picture[].Y`

PROPRIEDADE

# Propriedade: Picture[].MoveY

## Sintaxe

```
Picture[Picture-Id].MoveY
```

## Descrição

Com a propriedade MoveY é possível ler/gravar a coordenada Y atual de uma Picture durante o movimento.

## Propriedades

- **Tipo de dado:** `float`
- **Somente leitura:** não

## Exemplo

```
$
// Obter as coordenadas X e Y atuais da primeira
v[1] = Picture[1].MoveX;
v[2] = Picture[1].MoveY;
```

## Veja também

`Picture`, `Picture[]`, `Picture[].MoveX`, `Picture[].X`, `Picture[].Y`

PROPRIEDADE

# Propriedade: Picture[].Palette[]

## Sintaxe

```
Picture[Picture-Id].Palette[Índice]
```

## Descrição

Com a propriedade Palette é possível ler/gravar uma determinada entrada de paleta da Picture. Essa propriedade só foi mantida por compatibilidade com o DestinyPatch V1.0. Por isso, recomenda-se usar a propriedade `Picture[].AuroraSheet.Palette[]`, já que ela está disponível em todos os objetos AuroraSheet. A documentação dessa propriedade também se encontra lá.

PROPRIEDADE

# Propriedade: Picture[].Pixel[]

## Sintaxe

```
Picture[Picture-Id].Pixel[X, Y]
```

## Descrição

Com a propriedade Pixel é possível ler/gravar um pixel da Picture. Essa propriedade só foi mantida por compatibilidade com o DestinyPatch V1.0. Por isso, recomenda-se usar a propriedade `Picture[].AuroraSheet.Pixel[]`, já que ela está disponível em todos os objetos AuroraSheet. A documentação dessa propriedade também se encontra lá.

PROPRIEDADE

# Propriedade: Picture[].Red

## Sintaxe

```
Picture[Picture-Id].Red
```

## Descrição

Com a propriedade Red é possível ler/gravar a coloração vermelha de uma Picture, em porcentagem. O valor 0 significa que a Picture não apresenta a cor vermelha. O valor 100 significa que a Picture exibe o vermelho normalmente. O valor 200 significa que a Picture é exibida com tons de vermelho especialmente fortes.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não
- **Faixa de valores:** 0 ... 200

## Exemplo

```
$
// Obter a coloração vermelha da primeira Picture e
v[1] = Picture[1].Red;

// Reduzir em 1% a coloração vermelha da primeira
if(Picture[1].Red > 0)
  Picture[1].Red--;
endif;
```

## Veja também

`Picture`, `Picture[]`, `Picture[].Green`, `Picture[].Blue`, `Picture[].Chroma`

MÉTODO

# Método: Picture[].SaveAs()

## Sintaxe

```
Picture[Picture-Id].SaveAs(Nome do arquivo)
```

## Descrição

Com o método SaveAs é possível salvar o conteúdo atual da Picture. Por padrão, isso é feito na pasta Picture do jogo. A extensão ".bmp" precisa ser informada junto (veja o exemplo).

## Lista de parâmetros

### Parâmetro: `Nome do arquivo`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** Indica o nome do arquivo de destino (sem caminho, mas com extensão!). Esse caminho é relativo e, por isso, também pode, em princípio, ser usado para salvar imagens fora da pasta Picture.

## Exemplo

```
$
// Salvar a primeira Picture no arquivo "Ergebnis.bmp
Picture[1].SaveAs("Ergebnis.bmp");

// Depois, a Picture pode ser carregada de volta com o método LoadFro
Picture[1].LoadFromFile("Ergebnis");
```

## Erros possíveis

`DESTINYERROR_PICTURE`, `DESTINYERROR_MEMORY`, `DESTINYERROR_CANTRESOLVEPATH`, `DESTINYERROR_NOPERMISSION`, `DESTINYERROR_CANTOPENFILE`

## Veja também

`Picture`, `Picture[]`, `Picture[].LoadFromFile()`, `Picture[].AuroraSheet.SaveAs()`

PROPRIEDADE

# Propriedade: Picture[].Transparency

## Sintaxe

```
Picture[Picture-Id].Transparency
```

## Descrição

Com a propriedade Transparency é possível ler/gravar a transparência de uma Picture, em porcentagem. Com valores maiores que 100 ocorrem erros de cálculo divertidos.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não
- **Faixa de valores:** 0 ... 100

## Exemplo

```
$
// Obter a transparência da primeira Picture
v[1] = Picture[1].Transparency;

// Aumentar em 1% a transparência da primeira Pic
if(Picture[1].Transparency < 100)
  Picture[1].Transparency++;
endif;
```

## Veja também

`Picture`, `Picture[]`, `Picture[].Magnification`

PROPRIEDADE

# Propriedade: Picture[].UseMaskColor

## Sintaxe

```
Picture[Picture-Id].UseMaskColor
```

## Descrição

Com a propriedade UseMaskColor é possível ler/gravar se o RPG_RT, ao desenhar a Picture, usa a entrada de paleta 0 como cor de máscara (true) ou não (false). Essa propriedade só foi mantida por compatibilidade com o DestinyPatch V1.0. Por isso, recomenda-se usar a propriedade `Picture[].AuroraSheet.UseMaskColor`, já que ela está disponível em todos os objetos AuroraSheet. A documentação dessa propriedade também se encontra lá.

PROPRIEDADE

# Propriedade: Picture[].Width

## Sintaxe

```
Picture[Picture-Id].Width
```

## Descrição

Com a propriedade Width é possível obter a largura de uma Picture em pixels. Se no momento nenhuma imagem estiver carregada na Picture, esse valor é 0.

> **Informação:** Essa propriedade existe apenas por compatibilidade com o DestinyPatch V1.0. Seu valor é idêntico ao da propriedade `Width` do subobjeto `AuroraSheet` (veja o exemplo).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Obsoleto:** sim

## Exemplo

```
$
// Obter a largura da primeira Picture e guardar em
v[1] = Picture[1].Width;

// Obter a largura da primeira Picture pelo AuroraSh
v[2] = Picture[1].AuroraSheet.Width;
```

## Veja também

`Picture`, `Picture[]`, `Picture[].Height`, `Picture[].AuroraSheet.Width`

PROPRIEDADE

# Propriedade: Picture[].X

## Sintaxe

```
Picture[Picture-Id].X
```

## Descrição

Com a propriedade X é possível ler/gravar a coordenada X da Picture. No RPG Maker 2000, essa coordenada sempre se refere ao centro da Picture.

> **Informação:** `Picture[].X` sempre se refere à coordenada de destino de uma Picture. Se uma Picture não estiver se movendo, essa também é a coordenada atual. `Picture[].MoveX` se refere à coordenada atual de uma Picture durante um movimento. `Picture[].MapX` sempre se refere ao valor originalmente usado no comando ShowPicture. Esse valor só é relevante para o RPG Maker 2000, desde que o movimento "MoveWithMap" das Pictures não funcione como no RPG Maker 2003.

## Exemplo

```
$
// Obter a coordenada X da primeira Picture
```

```
v[1] = Picture[1].X;
// Obter a coordenada Y da primeira Picture
v[2] = Picture[1].Y;
// Posicionar a primeira Picture centralizada (largura d
Picture[1].X = (320 - Picture[1].Width) / 2;
Picture[1].Y = (240 - Picture[1].Height) / 2;
```

## Veja também

`Picture`, `Picture[]`, `Picture[].Y`, `Picture[].Width`, `Picture[].Height`, `Picture[].MoveX`, `Picture[].MapX`

PROPRIEDADE

# Propriedade: Picture[].Y

## Sintaxe

```
Picture[Picture-Id].Y
```

## Descrição

Com a propriedade Y é possível ler/gravar a coordenada Y da Picture. No RPG Maker 2000, essa coordenada sempre se refere ao centro da Picture.

> **Informação:** `Picture[].Y` sempre se refere à coordenada de destino de uma Picture. Se uma Picture não estiver se movendo, essa também é a coordenada atual. `Picture[].MoveY` se refere à coordenada atual de uma Picture durante um movimento. `Picture[].MapY` sempre se refere ao valor originalmente usado no comando ShowPicture. Esse valor só é relevante para o RPG Maker 2000, desde que o movimento "MoveWithMap" das Pictures não funcione como no RPG Maker 2003.

## Exemplo

```
$
// Obter a coordenada X da primeira Picture
v[1] = Picture[1].X;
// Obter a coordenada Y da primeira Picture
v[2] = Picture[1].Y;
// Posicionar a primeira Picture centralizada (largura d
Picture[1].X = (320 - Picture[1].Width) / 2;
Picture[1].Y = (240 - Picture[1].Height) / 2;
```

## Veja também

`Picture`, `Picture[]`, `Picture[].X`, `Picture[].Width`, `Picture[].Height`, `Picture[].MoveY`, `Picture[].MapY`

OBJETO

# Objeto: s

## Sintaxe

```
s
```

## Descrição

O objeto s serve para tornar acessíveis os switches (valores booleanos, "sim"/"não") do RPG_RT. Os switches do objeto s são gerenciados pelo RPG_RT.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Count` | Propriedade | Devolve a quantidade de switches no objeto s. |
| `s[]` | Propriedade | Permite o acesso a um determinado switch do objeto s. |

## Veja também

`a`, `d`, `f`, `v`

PROPRIEDADE

# Propriedade: s.Count

## Sintaxe

```
s.Count
```

## Descrição

Devolve a quantidade de switches no objeto `s`. Isso pode ser usado para fazer um laço sobre todos os switches.

> **Informação:** A quantidade de switches disponível só é definida em tempo de execução do jogo. Somente quando um switch (ou um com Switch-Id maior) é acessado pela primeira vez, ele é criado (com o valor false). Esse comportamento vem do RPG_RT e foi mantido assim no DestinyPatch. Por esse motivo, o objeto `s` também não tem um método ReAlloc.

Exemplo:

Se, depois de iniciado o jogo, o terceiro switch for acessado, então a partir desse momento existem três switches ( `s.Count` = 3). Se depois disso o quinto switch for acessado, então a partir desse momento existem cinco switches ( `s.Count` = 5). Se depois disso o terceiro switch for acessado de novo, continuam existindo cinco switches (`s.Count` = 5).

## Propriedades

- **Tipo de dado:** `dword`

- **Somente leitura:** sim

## Exemplo

```
$
// Definir todos os switches como true
for(v[1] = 1; v[1] <= s.Count; v[1]++)
  s[v[1]] = true;
next;
```

## Veja também

`s`, `s[]`

PROPRIEDADE

# Propriedade: s[]

## Sintaxe

```
s[Switch-Id]
```

## Descrição

Permite o acesso a um determinado switch do objeto `s`.

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** não

## Lista de parâmetros

### Parâmetro: `Switch-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 1 && <= 100000000
- **Descrição:** A Switch-Id do switch. Esse valor precisa ser maior/igual a 1. Se for informada uma Switch-Id maior que `s.Count`, a quantidade de switches aumenta automaticamente.

## Exemplo

```
$
// Definir o valor do primeiro switch como false
s[1] = false;
```

## Erros possíveis

`DESTINYERROR_ARRAYBOUND`

## Veja também

`s`, `s.Count`

OBJETO

# Objeto: Screen

## Sintaxe

```
Screen
```

## Descrição

O objeto Screen serve para disponibilizar a tela atual do jogo.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Blue` | Propriedade | Com a propriedade Blue é possível ler/gravar a coloração azul da tela, em porcentagem. |
| `Chroma` | Propriedade | Com a propriedade Chroma é possível ler/gravar a crominância (saturação de cor) da tela, em porcentagem. |
| `Green` | Propriedade | Com a propriedade Green é possível ler/gravar a coloração verde da tela, em porcentagem. |
| `HighColor` | Propriedade | Com a propriedade HighColor é possível descobrir se a tela tem profundidade de cor de 16 bits (modo 565). |
| `Red` | Propriedade | Com a propriedade Red é possível ler/gravar a coloração vermelha da tela, em porcentagem. |
| `TrueColor` | Propriedade | Com a propriedade TrueColor é possível descobrir se a tela tem profundidade de cor de 32 bits (modo XRGB). |

PROPRIEDADE

# Propriedade: Screen.Blue

## Sintaxe

```
Screen.Blue
```

## Descrição

Com a propriedade Blue é possível ler/gravar a coloração azul da tela, em porcentagem. O valor 0 significa que a tela não apresenta a cor azul. O valor 100 significa que a tela exibe o azul normalmente. O valor 200 significa que a tela é exibida com tons de azul especialmente fortes.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

## Exemplo

```
$
// Obter a coloração azul da tela e guardar em
v[1] = Screen.Blue;

// Reduzir em 1% a coloração azul da tela
if(Screen.Blue > 0)
  Screen.Blue--;
endif;
```

## Veja também

`Screen`, `Screen.Red`, `Screen.Green`, `Screen.Chroma`, `Command.SetScreenTone()`

PROPRIEDADE

# Propriedade: Screen.Chroma

## Sintaxe

```
Screen.Chroma
```

## Descrição

Com a propriedade Chroma é possível ler/gravar a crominância (saturação de cor) da tela, em porcentagem. O valor 0 significa que a tela é exibida dessaturada. O valor 100 significa que a tela é exibida normalmente. O valor 200 significa que a tela é exibida com diferenças de cor especialmente grandes.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

## Exemplo

```
$
// Obter a crominância da tela e guardar em v[1]
v[1] = Screen.Chroma;

// Reduzir em 1% a crominância da tela
if(Screen.Chroma > 0)
  Screen.Chroma--;
endif;
```

## Veja também

`Screen`, `Screen.Red`, `Screen.Green`, `Screen.Blue`, `Command.SetScreenTone()`

PROPRIEDADE

# Propriedade: Screen.Green

## Sintaxe

```
Screen.Green
```

## Descrição

Com a propriedade Green é possível ler/gravar a coloração verde da tela, em porcentagem. O valor 0 significa que a tela não apresenta a cor verde. O valor 100 significa que a tela exibe o verde normalmente. O valor 200 significa que a tela é exibida com tons de verde especialmente fortes.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

## Exemplo

```
$
// Obter a coloração verde da tela e guardar em
v[1] = Screen.Green;

// Reduzir em 1% a coloração verde da tela
if(Screen.Green > 0)
  Screen.Green--;
endif;
```

## Veja também

`Screen`, `Screen.Red`, `Screen.Blue`, `Screen.Chroma`, `Command.SetScreenTone()`

PROPRIEDADE

# Propriedade: Screen.HighColor

## Sintaxe

```
Screen.HighColor
```

## Descrição

Com a propriedade HighColor é possível descobrir se a tela tem profundidade de cor de 16 bits (modo 565). False: a profundidade de cor é de 32 bits. True: a profundidade de cor é de 16 bits.

> **Informação:** A profundidade de cor do jogo não influencia a quantidade de cores por gráfico (256), apenas a qualidade de exibição delas.

> **Informação:** A profundidade de cor pode ser alterada pelo DestinyPatcher.

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

## Exemplo

```
$
// Verificar se a tela tem profundidade de cor de 1
if(Screen.HighColor)
  // ...
endif;
```

## Veja também

`Screen`, `Screen.TrueColor`

PROPRIEDADE

# Propriedade: Screen.Red

## Sintaxe

```
Screen.Red
```

## Descrição

Com a propriedade Red é possível ler/gravar a coloração vermelha da tela, em porcentagem. O valor 0 significa que a tela não apresenta a cor vermelha. O valor 100 significa que a tela exibe o vermelho normalmente. O valor 200 significa que a tela é exibida com tons de vermelho especialmente fortes.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

## Exemplo

```
$
// Obter a coloração vermelha da tela e guardar em
v[1] = Screen.Red;

// Reduzir em 1% a coloração vermelha da tela
if(Screen.Red > 0)
  Screen.Red--;
endif;
```

## Veja também

`Screen`, `Screen.Green`, `Screen.Blue`, `Screen.Chroma`, `Command.SetScreenTone()`

PROPRIEDADE

# Propriedade: Screen.TrueColor

## Sintaxe

```
Screen.TrueColor
```

## Descrição

Com a propriedade TrueColor é possível descobrir se a tela tem profundidade de cor de 32 bits (modo XRGB). False: a profundidade de cor é de 16 bits. True: a profundidade de cor é de 32 bits.

> **Informação:** A profundidade de cor do jogo não influencia a quantidade de cores por gráfico (256), apenas a qualidade de exibição delas.

> **Informação:** A profundidade de cor pode ser alterada pelo DestinyPatcher.

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

## Exemplo

```
$
// Verificar se a tela tem profundidade de cor de 3
if(Screen.TrueColor)
  // ...
endif;
```

## Veja também

`Screen`, `Screen.HighColor`

OBJETO

# Objeto: Server

## Sintaxe

```
Server
```

## Descrição

O objeto Server serve para aceitar conexões de entrada pela internet ou pela rede. Depois de aceita, a conexão é acessada através do objeto `Client[]`.

> **Informação:** Em conexões de entrada pela internet, podem ocorrer problemas ao usar um firewall ou roteador. Como configurar o firewall/roteador para permitir conexões de entrada pode ser consultado na internet.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Accept()` | Método | Com o método Accept é possível ao objeto Server aceitar uma conexão de entrada. |
| `Close()` | Método | Com o método Close é possível fechar o objeto Server. |
| `Listen()` | Método | Com o método Listen é possível fazer o objeto Server passar a esperar por conexões de entrada. |
| `State` | Propriedade | Com a propriedade State é possível descobrir o status de conexão atual do servidor. |
| `Type` | Propriedade | Com a propriedade Type é possível descobrir o tipo de socket das conexões aceitas. |

## Veja também

`Client`

MÉTODO

# Método: Server.Accept()

## Sintaxe

```
Server.Accept([Cliente-Índice])
```

## Descrição

Com o método Accept é possível ao objeto Server aceitar uma conexão de entrada. Para isso, o objeto Server precisa antes ser colocado no estado de espera por conexões de entrada, por meio de `Server.Listen()`. As conexões aceitas são acessadas através do objeto `Client[]`. Com este método também se verifica se há uma conexão de entrada pendente. Se não for o caso, a chamada termina imediatamente e é devolvido -1.

> **Informação:** Em conexões de entrada pela internet, podem ocorrer problemas ao usar um firewall ou roteador. Como configurar o firewall/roteador para permitir conexões de entrada pode ser consultado na internet.

## Lista de parâmetros

### Parâmetro: `Cliente-Índice`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** NEXT_FREE_SOCKET
- **Faixa de valores:** -1 ... (Client.Count - 1)
- **Descrição:** Indica o índice do socket cliente dentro do objeto `Client[]` pelo qual a conexão deve ser aceita. Se o respectivo socket cliente não estiver fechado, ele é fechado automaticamente antes de aceitar a conexão. Em vez de um índice de cliente, também pode ser informada a constante `NEXT_FREE_SOCKET` (= -1), para que um socket cliente livre seja escolhido automaticamente (se houver). Se não houver nenhum socket cliente livre, é lançado um erro. A faixa de valores deste parâmetro fica, portanto, entre -1 e (`Client.Count` - 1).

Para manter dinâmica a quantidade máxima de conexões a aceitar, também pode ser usado como parâmetro o valor de retorno do método `NextFree` do objeto `Client`.

### Valor de retorno

- **Tipo de dado:** `dword`
- **Descrição:** O índice dentro do objeto `Client[]`, ou -1, caso não houvesse conexão de entrada pendente.

## Exemplo

```
$
// Aceitar conexões de entrada na porta 80
Server.Listen(80, SOCK_RAW);

// Aguardar uma conexão de entrada
do
  v[1] = Server.Accept(Client.NextFree());
loop until (v[1] != -1);

// Uma conexão de entrada foi aceita. Depois
// -> enviar "olá" ao cliente
Client[v[1]].SendRawData("hallo");
Client[v[1]].Close();

// Fechar o servidor novamente
Server.Close();
```

## Erros possíveis

`DESTINYERROR_NOFREESOCKET`, `DESTINYERROR_SOCKETERROR`, `DESTINYERROR_SOCKETNOTOPEN`, `DESTINYERROR_VALUE`

## Veja também

`Server`, `Server.Listen()`, `Server.Close()`, `Sockettyp`, `Client[]`, `Client[].SendRawData()`, `Client[].Close()`, `Client.NextFree()`

MÉTODO

# Método: Server.Close()

## Sintaxe

```
Server.Close()
```

## Descrição

Com o método Close é possível fechar o objeto Server. Se o objeto Server tiver sido colocado antes, por meio de `Server.Listen()`, no estado de espera por conexões de entrada, esse estado termina com a chamada deste método. As conexões já aceitas por `Server.Accept()` continuam existindo e precisam, se for o caso, ser fechadas separadamente por `Client[].Close()` ou `Client.CloseAll()`. Se um servidor já fechado for fechado novamente, nada acontece (por isso, não é problema fechar o objeto Server por segurança no início de um script).

## Exemplo

```
$
// Aceitar conexões de entrada na porta 80
Server.Listen(80, SOCK_RAW);

// Aguardar uma conexão de entrada
do
  v[1] = Server.Accept(Client.NextFree());
loop until (v[1] != -1);

// Uma conexão de entrada foi aceita. Depois
// -> enviar "olá" ao cliente
Client[v[1]].SendRawData("hallo");
Client[v[1]].Close();

// Fechar o servidor novamente
Server.Close();
```

## Veja também

`Server`, `Server.Listen()`, `Server.Accept()`, `Sockettyp`, `Client[]`, `Client[].SendRawData()`, `Client[].Close()`, `Client.NextFree()`, `Client.CloseAll()`

MÉTODO

# Método: Server.Listen()

## Sintaxe

```
Server.Listen(Port, [Tipo de socket])
```

## Descrição

Com o método Listen é possível fazer o objeto Server passar a esperar por conexões de entrada. As conexões de entrada são aceitas depois pelo método `Accept`.

> **Informação:** Em conexões de entrada pela internet, podem ocorrer problemas ao usar um firewall ou roteador. Como configurar o firewall/roteador para permitir conexões de entrada pode ser consultado na internet.

## Lista de parâmetros

### Parâmetro: `Port`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... 65535
- **Descrição:** A porta na qual se deve esperar por conexões de entrada. Aqui, para protocolos de livre escolha, deveriam ser escolhidas portas maiores que 1000.

### Parâmetro: `Tipo de socket`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 0
- **Faixa de valores:** 0 ... 1
- **Descrição:** O tipo de socket que os sockets cliente terão nas conexões aceitas. Para isso são usadas as constantes de `tipo de socket`.

`SOCK_DESTINY`: usa-se o `protocolo Destiny` (protocolo fácil de usar, projetado especificamente para o RPG Maker). `SOCK_RAW`: não se usa nenhum protocolo especial. O usuário pode enviar e receber dados brutos (geralmente Strings, mas também outros tipos de dado são possíveis). Isso é usado, por exemplo, para requisições web.

## Exemplo

```
$
// Aceitar conexões de entrada na porta 80
Server.Listen(80, SOCK_RAW);

// Aguardar uma conexão de entrada
do
  v[1] = Server.Accept(Client.NextFree());
loop until (v[1] != -1);

// Uma conexão de entrada foi aceita. Depois
// -> enviar "olá" ao cliente
Client[v[1]].SendRawData("hallo");
Client[v[1]].Close();

// Fechar o servidor novamente
Server.Close();
```

## Erros possíveis

`DESTINYERROR_CANTCREATESOCKET`, `DESTINYERROR_SOCKETSTILLOPEN`, `DESTINYERROR_SOCKETSTARTUP`, `DESTINYERROR_VALUE`

## Veja também

`Server`, `Server.Accept()`, `Server.Close()`, Sockettyp, `Client[]`, `Client[].SendRawData()`, `Client[].Close()`, `Client.NextFree()`

PROPRIEDADE

# Propriedade: Server.State

## Sintaxe

```
Server.State
```

## Descrição

Com a propriedade State é possível descobrir o status de conexão atual do servidor. Para isso são usadas as constantes de `status de socket`.

`STATE_ERROR`: ocorreu um erro. Normalmente, deve-se então chamar o método `Close`, já que a conexão do objeto não pode mais ser usada. `STATE_CLOSED`: o objeto Server não está esperando por conexões de entrada no momento. Para fazê-lo esperar por conexões, é preciso chamar o método `Listen` desse objeto. `STATE_LISTENING`: o objeto Server está esperando por conexões de entrada no momento.

As conexões de entrada podem ser aceitas pelo método `Accept`. Para que o objeto deixe de esperar por conexões de entrada, é preciso chamar o método `Close`.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** -1; 0; 2

## Exemplo

```
$
// Aceitar conexões de entrada na porta 80
Server.Listen(80, SOCK_RAW);

// Aguardar uma conexão de entrada
do
  v[1] = Server.Accept(Client.NextFree());
loop until (v[1] != -1);

// Uma conexão de entrada foi aceita. Depois
// -> enviar "olá" ao cliente
Client[v[1]].SendRawData("hallo");
Client[v[1]].Close();

// Fechar o servidor novamente
Server.Close();
```

## Veja também

`Server`, `Server.Listen()`, `Server.Accept()`, `Server.Close()`, `Server.Type`, `Client[]`, `Client[].SendRawData()`, `Client[].Close()`, `Client.NextFree()`

PROPRIEDADE

# Propriedade: Server.Type

## Sintaxe

```
Server.Type
```

## Descrição

Com a propriedade Type é possível descobrir o tipo de socket das conexões aceitas. Para isso são usadas as constantes de `tipo de socket`. O valor dessa propriedade é definido ao chamar o método `Listen` desse objeto.

`SOCK_DESTINY`: usa-se o `protocolo Destiny` (protocolo fácil de usar, projetado especificamente para o RPG Maker). `SOCK_RAW`: não se usa nenhum protocolo especial. O usuário pode enviar e receber dados brutos (geralmente Strings, mas também outros tipos de dado são possíveis). Isso costuma ser usado, por exemplo, para requisições web.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim
- **Faixa de valores:** 0 ... 1

## Exemplo

```
$
// Aceitar conexões de entrada na porta 80
Server.Listen(80, SOCK_RAW);

// Consultar o tipo de socket
if(Server.Type == SOCK_RAW)
  // A condição é sempre satisfeita
  // ...
endif;

// Fechar o servidor novamente
Server.Close();
```

## Veja também

`Server`, `Server.Listen()`, `Server.Accept()`, `Server.Close()`, `Server.State`, `Sockettyp`, `Client[]`

OBJETO

# Objeto: Skill

## Sintaxe

```
Skill
```

## Descrição

O objeto Skill serve para reunir todas as propriedades de uma técnica. As propriedades gerais ficam reunidas na categoria `Geral`. Propriedades especiais (que valem, por exemplo, só para técnicas de ataque) ficam reunidas no subobjeto correspondente. Para saber que tipo de técnica se trata, deve-se usar a propriedade Type.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Count` | Propriedade | Devolve a quantidade de técnicas. |
| `Skill[]` | Objeto | Tipo de dado: dword Opcional: não Faixa de valores: 1 ... |

PROPRIEDADE

# Propriedade: Skill.Count

## Sintaxe

```
Skill.Count
```

## Descrição

Devolve a quantidade de técnicas.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

OBJETO

# Objeto: Skill[]

## Sintaxe

```
Skill[Habilidade-Id]
```

## Descrição

## Lista de parâmetros

### Parâmetro: `Habilidade-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... Skill.Count
- **Descrição:** A Skill-Id da técnica. Precisa estar na faixa de 1 ... `Skill.Count`.

## Erros possíveis

`DESTINYERROR_ARRAYBOUND`

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Geral` | Categoria | A categoria Geral reúne todas as propriedades de uma técnica que valem para qualquer tipo de técnica. |
| `Normal` | Objeto | O objeto Normal reúne todas as propriedades do objeto Skill que valem para técnicas comuns do sistema de batalha. |
| `Switch` | Objeto | O objeto Switch reúne todas as propriedades específicas de switch do objeto Skill. |

CATEGORIA

# Categoria: Geral

## Descrição

A categoria Geral reúne todas as propriedades de uma técnica que valem para qualquer tipo de técnica.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Description` | Propriedade | Com a propriedade Description é possível descobrir o texto de descrição de uma técnica. |
| `FailureMessage` | Propriedade | Com a propriedade FailureMessage é possível descobrir a mensagem de falha de uma técnica. |
| `MpCost` | Propriedade | Com a propriedade MpCost é possível descobrir a quantidade de pontos de magia necessária para uma técnica. |
| `Name` | Propriedade | Com a propriedade Name é possível descobrir o nome de uma técnica. |
| `Sound` | Propriedade | Com a propriedade Sound é possível descobrir o efeito sonoro reproduzido ao usar uma técnica. |
| `Type` | Propriedade | Com a propriedade Type é possível descobrir o tipo de uma técnica. |
| `UseMessage` | Propriedade | Com a propriedade UseMessage é possível descobrir a primeira linha da mensagem de uso de uma técnica. |
| `UseMessage2` | Propriedade | Com a propriedade UseMessage2 é possível descobrir a segunda linha da mensagem de uso de uma técnica. |

PROPRIEDADE

# Propriedade: Skill[].Description

## Sintaxe

```
Skill[Habilidade-Id].Description
```

## Descrição

Com a propriedade Description é possível descobrir o texto de descrição de uma técnica.

## Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Skill[].FailureMessage

## Sintaxe

```
Skill[Habilidade-Id].FailureMessage
```

## Descrição

Com a propriedade FailureMessage é possível descobrir a mensagem de falha de uma técnica.

## Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Skill[].MpCost

## Sintaxe

```
Skill[Habilidade-Id].MpCost
```

## Descrição

Com a propriedade MpCost é possível descobrir a quantidade de pontos de magia necessária para uma técnica.

## Propriedades

- **Tipo de dado:** `dword`

- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Skill[].Name

### Sintaxe

```
Skill[Habilidade-Id].Name
```

### Descrição

Com a propriedade Name é possível descobrir o nome de uma técnica.

### Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Skill[].Sound

### Sintaxe

```
Skill[Habilidade-Id].Sound
```

### Descrição

Com a propriedade Sound é possível descobrir o efeito sonoro reproduzido ao usar uma técnica.

### Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Skill[].Type

### Sintaxe

```
Skill[Habilidade-Id].Type
```

### Descrição

Com a propriedade Type é possível descobrir o tipo de uma técnica. Para identificar esse valor, podem ser usadas as constantes `SkillType`:

Constante | Descrição

A técnica é uma técnica de

`SKILLTYPE_NORMAL`

combate normal

`SKILLTYPE_TELEPORT` A técnica executa uma teletransportação

A técnica executa uma fuga em

`SKILLTYPE_ESCAPE`

combate

A técnica liga um switch ao

`SKILLTYPE_SWITCH`

ser usada

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Skill[].UseMessage

## Sintaxe

```
Skill[Habilidade-Id].UseMessage
```

## Descrição

Com a propriedade UseMessage é possível descobrir a primeira linha da mensagem de uso de uma técnica.

## Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Skill[].UseMessage2

## Sintaxe

```
Skill[Habilidade-Id].UseMessage2
```

## Descrição

Com a propriedade UseMessage2 é possível descobrir a segunda linha da mensagem de uso de uma técnica.

## Propriedades

- **Tipo de dado:** `string`
- **Somente leitura:** sim

OBJETO

# Objeto: Skill[].Normal

## Sintaxe

```
Skill[Habilidade-Id].Normal
```

## Descrição

O objeto Normal reúne todas as propriedades do objeto `Skill` que valem para técnicas comuns do sistema de batalha. Os valores desse objeto só são válidos quando a técnica também é uma técnica comum do sistema de batalha.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Absorption` | Propriedade | Com a propriedade Absorption é possível descobrir se, ao usar essa técnica, os respectivos pontos de status do inimigo são absorvidos. |
| `Agility` | Propriedade | Com a propriedade Agility é possível descobrir a quantidade de pontos de agilidade que a técnica faz efeito, dependendo do tipo. |
| `Attack` | Propriedade | Com a propriedade Attack é possível descobrir a quantidade de pontos de ataque que a técnica faz efeito, dependendo do tipo. |
| `BaseEffect` | Propriedade | Com a propriedade BaseEffect é possível descobrir o valor-base para o cálculo dos pontos de dano da técnica. |
| `BaseSuccessRate` | Propriedade | Com a propriedade BaseSuccessRate é possível descobrir o valor-base para determinar o sucesso da técnica. |
| `BattleAnimation` | Propriedade | Com a propriedade BattleAnimation é possível descobrir a BattleAnimation-Id da técnica. |
| `ChangeAttributeEffect` | Propriedade | Com a propriedade ChangeAttributeEffect é possível descobrir se, ao usar essa técnica, os níveis de efeito dos respectivos atributos do inimigo/herói podem ser alterados. |
| `Defense` | Propriedade | Com a propriedade Defense é possível descobrir a quantidade de pontos de defesa que a técnica faz efeito, dependendo do tipo. |
| `HasAttribute()` | Método | Com o método HasAttribute é possível descobrir se a técnica tem um determinado atributo. |
| `HasCondition()` | Método | Com o método HasCondition é possível descobrir se a técnica pode curar ou causar uma determinada condição. |
| `HitChance` | Propriedade | Com a propriedade HitChance é possível descobrir a taxa de acerto dessa técnica. |
| `HP` | Propriedade | Com a propriedade HP é possível descobrir a quantidade de pontos de vida que a técnica cura ou retira como dano, dependendo do tipo. |
| `IgnoreDefense` | Propriedade | Com a propriedade IgnoreDefense é possível descobrir se, ao usar essa técnica, os pontos de defesa do inimigo são ignorados. |

| Nome | Tipo | Resumo |
| --- | --- | --- |
| `Mind` | Propriedade | Com a propriedade Mind é possível descobrir a quantidade de pontos de inteligência que a técnica faz efeito, dependendo do tipo. |
| `MindChance` | Propriedade | Com a propriedade MindChance é possível descobrir a taxa de acerto para ataques mágicos. |
| `MP` | Propriedade | Com a propriedade MP é possível descobrir a quantidade de pontos de magia que a técnica cura ou retira como dano, dependendo do tipo. |
| `Scope` | Propriedade | Com a propriedade Scope é possível descobrir o alcance de efeito da técnica. |
| `Variance` | Propriedade | Com a propriedade Variance é possível descobrir a variância (a margem) para o cálculo dos pontos de dano da técnica. |

PROPRIEDADE

# Propriedade: Skill[].Normal.Absorption

## Sintaxe

```
Skill[Habilidade-Id].Normal.Absorption
```

## Descrição

Com a propriedade Absorption é possível descobrir se, ao usar essa técnica, os respectivos pontos de status do inimigo são absorvidos.

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Skill[].Normal.Agility

## Sintaxe

```
Skill[Habilidade-Id].Normal.Agility
```

## Descrição

Com a propriedade Agility é possível descobrir a quantidade de pontos de agilidade que a técnica faz efeito, dependendo do tipo.

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Skill[].Normal.Attack

### Sintaxe

```
Skill[Habilidade-Id].Normal.Attack
```

### Descrição

Com a propriedade Attack é possível descobrir a quantidade de pontos de ataque que a técnica faz efeito, dependendo do tipo.

### Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Skill[].Normal.BaseEffect

### Sintaxe

```
Skill[Habilidade-Id].Normal.BaseEffect
```

### Descrição

Com a propriedade BaseEffect é possível descobrir o valor-base para o cálculo dos pontos de dano da técnica. Esse valor é informado em pontos de vida.

### Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

## Propriedade: Skill[].Normal.BaseSuccessRate

### Sintaxe

```
Skill[Habilidade-Id].Normal.BaseSuccessRate
```

### Descrição

Com a propriedade BaseSuccessRate é possível descobrir o valor-base para determinar o sucesso da técnica. Esse valor é informado em porcentagem (0 ... 100%) e indica, no fundo, qual a chance de a técnica ter sucesso.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Skill[].Normal.BattleAnimation

## Sintaxe

```
Skill[Habilidade-Id].Normal.BattleAnimation
```

## Descrição

Com a propriedade BattleAnimation é possível descobrir a BattleAnimation-Id da técnica.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Skill[].Normal.ChangeAttributeEffect

## Sintaxe

```
Skill[Habilidade-Id].Normal.ChangeAttributeEffect
```

## Descrição

Com a propriedade ChangeAttributeEffect é possível descobrir se, ao usar essa técnica, os níveis de efeito dos respectivos atributos do inimigo/herói podem ser alterados.

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Skill[].Normal.Defense

## Sintaxe

```
Skill[Habilidade-Id].Normal.Defense
```

## Descrição

Com a propriedade Defense é possível descobrir a quantidade de pontos de defesa que a técnica faz efeito, dependendo do tipo.

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

MÉTODO

# Método: Skill[].Normal.HasAttribute()

## Sintaxe

```
Skill[Habilidade-Id].Normal.HasAttribute(Attribute-
Id)
```

## Descrição

Com o método HasAttribute é possível descobrir se a técnica tem um determinado atributo.

## Lista de parâmetros

### Parâmetro: `Atributo-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... Attribute.Count
- **Descrição:** Indica a Id do atributo.

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** Indica se a técnica tem ou não o atributo.

## Erros possíveis

`DESTINYERROR_VALUE`

MÉTODO

# Método: Skill[].Normal.HasCondition()

## Sintaxe

```
Skill[Habilidade-Id].Normal.HasCondition(Condition-
Id)
```

## Descrição

Com o método HasCondition é possível descobrir se a técnica pode curar ou causar uma determinada condição.

## Lista de parâmetros

### Parâmetro: `Condição-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 1 ... Condition.Count
- **Descrição:** Indica a Id da condição.

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** Indica se a técnica tem ou não a condição.

## Erros possíveis

`DESTINYERROR_VALUE`

PROPRIEDADE

# Propriedade: Skill[].Normal.HitChance

## Sintaxe

```
Skill[Habilidade-Id].Normal.HitChance
```

## Descrição

Com a propriedade HitChance é possível descobrir a taxa de acerto dessa técnica. Esse valor normalmente indica o quanto os pontos de ataque do herói influenciam no ataque.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Skill[].Normal.HP

## Sintaxe

```
Skill[Habilidade-Id].Normal.HP
```

## Descrição

Com a propriedade HP é possível descobrir a quantidade de pontos de vida que a técnica cura ou retira como dano, dependendo do tipo.

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Skill[].Normal.IgnoreDefense

## Sintaxe

```
Skill[Habilidade-Id].Normal.IgnoreDefense
```

## Descrição

Com a propriedade IgnoreDefense é possível descobrir se, ao usar essa técnica, os pontos de defesa do inimigo são ignorados.

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Skill[].Normal.Mind

## Sintaxe

```
Skill[Habilidade-Id].Normal.Mind
```

## Descrição

Com a propriedade Mind é possível descobrir a quantidade de pontos de inteligência que a técnica faz efeito, dependendo do tipo.

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Skill[].Normal.MindChance

## Sintaxe

```
Skill[Habilidade-Id].Normal.MindChance
```

## Descrição

Com a propriedade MindChance é possível descobrir a taxa de acerto para ataques mágicos. Esse valor normalmente indica o quanto os pontos de inteligência do herói influenciam no ataque.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Skill[].Normal.MP

## Sintaxe

```
Skill[Habilidade-Id].Normal.MP
```

## Descrição

Com a propriedade MP é possível descobrir a quantidade de pontos de magia que a técnica cura ou retira como dano, dependendo do tipo.

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Skill[].Normal.Scope

## Sintaxe

```
Skill[Habilidade-Id].Normal.Scope
```

## Descrição

Com a propriedade Scope é possível descobrir o alcance de efeito da técnica.

Constante | Descrição

O alvo da técnica é um único

`SKILLSCOPE_SINGLEENEMY`

inimigo

O alvo da técnica são todos os

`SKILLSCOPE_ALLENEMIES`

inimigos

O alvo da técnica é o herói que

`SKILLSCOPE_SELF`

está usando essa técnica

O alvo da técnica é um único

`SKILLSCOPE_SINGLEALLY`

herói

O alvo da técnica são todos os

`SKILLSCOPE_ALLALLIES`

heróis

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Skill[].Normal.Variance

## Sintaxe

```
Skill[Habilidade-Id].Normal.Variance
```

## Descrição

Com a propriedade Variance é possível descobrir a variância (a margem) para o cálculo dos pontos de dano da técnica. Um valor de 0 normalmente significa que não há variância disponível. Um valor de 1 normalmente significa que ±5% de dano é aplicado. Um valor de 2 normalmente significa que ±10% de dano é aplicado.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

OBJETO

# Objeto: Skill[].Switch

## Sintaxe

```
Skill[Habilidade-Id].Switch
```

## Descrição

O objeto Switch reúne todas as propriedades específicas de switch do objeto `Skill`. Os valores desse objeto só são válidos quando a técnica também aciona um switch (`Skill[].Type` precisa ser `SKILLTYPE_SWITCH`).

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `AvailableOnBattle` | Propriedade | Com a propriedade AvailableOnBattle é possível descobrir se essa técnica também pode ser usada durante o combate. |
| `AvailableOnField` | Propriedade | Com a propriedade AvailableOnField é possível descobrir se essa técnica também pode ser usada no mapa. |
| `Switch` | Propriedade | Com a propriedade Switch é possível descobrir a Switch-Id do switch acionado ao usar essa técnica. |

PROPRIEDADE

# Propriedade: Skill[].Switch.AvailableOnBattle

## Sintaxe

```
Skill[Habilidade-Id].Switch.AvailableOnBattle
```

## Descrição

Com a propriedade AvailableOnBattle é possível descobrir se essa técnica também pode ser usada durante o combate.

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

PROPRIEDADE

# Propriedade: Skill[].Switch.AvailableOnField

## Sintaxe

```
Skill[Habilidade-Id].Switch.AvailableOnField
```

## Descrição

Com a propriedade AvailableOnField é possível descobrir se essa técnica também pode ser usada no mapa.

## Propriedades

- **Tipo de dado:** `bool`
- **Somente leitura:** sim

PROPRIEDADE

### Propriedade: Skill[].Switch.Switch

## Sintaxe

```
Skill[Habilidade-Id].Switch.Switch
```

## Descrição

Com a propriedade Switch é possível descobrir a Switch-Id do switch acionado ao usar essa técnica.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

OBJETO

# Objeto: String

## Sintaxe

```
String
```

## Descrição

O objeto String serve para reunir todos os métodos relacionados a Strings.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Chr()` | Método | Com o método Chr é possível converter um código ASCII em uma string. |
| `Compare()` | Método | Com o método Compare é possível comparar duas strings entre si. |
| `CompareLex()` | Método | Com o método CompareLex é possível comparar duas strings lexicograficamente. |
| `Fill()` | Método | Com o método Fill é possível concatenar uma string repetidamente. |
| `Format()` | Método | Com o método Format é possível formatar um número através de uma string. |
| `Length()` | Método | Com o método Length é possível saber o comprimento de uma string. |

| Nome | Tipo | Resumo |
|---|---|---|
| `LTrim()` | Método | Com o método LTrim é possível remover espaços da borda esquerda de uma string. |
| `MonthName()` | Método | Com o método MonthName é possível converter um número no nome de um mês. |
| `Ord()` | Método | Com o método Ord é possível saber o código ASCII de um caractere dentro de uma string. |
| `Pos()` | Método | Com o método Pos é possível saber a posição de uma substring dentro de outra string. |
| `Replace()` | Método | Com o método Replace é possível substituir todas as ocorrências de uma substring dentro de uma expressão por outra substring. |
| `Reverse()` | Método | Com o método Reverse é possível inverter o conteúdo de uma string. |
| `RTrim()` | Método | Com o método RTrim é possível remover espaços da borda direita de uma string. |
| `SubStr()` | Método | Com o método SubStr é possível devolver uma substring de uma string. |
| `ToLower()` | Método | Com o método ToLower é possível converter todas as letras (A - Z) de uma string em minúsculas. |
| `ToUpper()` | Método | Com o método ToUpper é possível converter todas as letras (a - z) de uma string em maiúsculas. |
| `Trim()` | Método | Com o método Trim é possível remover espaços das bordas esquerda e direita de uma string. |
| `WeekdayName()` | Método | Com o método WeekdayName é possível converter um número no nome de um dia da semana. |

MÉTODO

# Método: String.Chr()

## Sintaxe

```
String.Chr(ASCII-Code, [Quantidade])
```

## Descrição

Com o método Chr é possível converter um código ASCII em uma string. Se o parâmetro opcional "Quantidade" não for informado, é gerada uma string de exatamente um caractere.

## Lista de parâmetros

### Parâmetro: `Código-ASCII`

- **Tipo de dado:** `byte`
- **Opcional:** não
- **Faixa de valores:** 0 ... 255
- **Descrição:** O código ASCII do caractere (0 ... 255)

### Parâmetro: `Quantidade`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 1
- **Faixa de valores:** >= 0
- **Descrição:** Indica quantas vezes o caractere deve ser repetido

### Valor de retorno

- **Tipo de dado:** `string`
- **Descrição:** A string com o caractere ASCII desejado

## Exemplo

```
$
// Montar uma string
a[1] = String.Chr(0x48) .. String.Chr(0x61) .. St
```

## Erros possíveis

`DESTINYERROR_MEMORY`, `DESTINYERROR_VALUE`

## Veja também

`String.Ord()`

MÉTODO

# Método: String.Compare()

## Sintaxe

```
String.Compare(String1, String2,
[Diferenciar maiúsculas/minúsculas])
```

## Descrição

Com o método Compare é possível comparar duas strings entre si.

## Lista de parâmetros

### Parâmetro: `String1`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** A primeira string a comparar

### Parâmetro: `String2`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** A segunda string a comparar

### Parâmetro: `Diferenciar maiúsculas/minúsculas`

- **Tipo de dado:** `bool`
- **Opcional:** sim
- **Valor padrão:** True

- **Descrição:** Indica se deve ser feita distinção entre maiúsculas e minúsculas (True) ou não (False).

### Valor de retorno

- **Tipo de dado:** `bool`
- **Descrição:** Devolve True se as duas strings forem iguais; caso contrário, False

## Exemplo

```
$
// Verificar se "Katze" e "Katze" são iguais e
s[1] = String.Compare("Katze", "Katze");        /
// Verificar se "Hund" e "Katze" são iguais e
s[2] = String.Compare("Hund", "Katze");         /
// Verificar se "Hund" e "hund" são iguais, distinguindo
s[3] = String.Compare("Hund", "hund");          /
// Verificar se "Hund" e "hund" são iguais (sem
s[4] = String.Compare("Hund", "hund", False);   /
```

## Observações adicionais

O uso desse método também é possível (como em todos os métodos que devolvem valores) nas condições de ramificações If, mas em comparações com distinção de maiúsculas/minúsculas recomenda-se, de modo geral, usar o operador de comparação.

```
$
// Comparação com este método
if(String.Compare("Wort", "Zahl"))
  // …
endif;

// Comparação totalmente equivalente com o operador de
if("Wort" == "Zahl")
  // …
endif;
```

## Veja também

`String.CompareLex()`

MÉTODO

# Método: String.CompareLex()

## Sintaxe

```
String.CompareLex(String1, String2,
[Diferenciar maiúsculas/minúsculas])
```

## Descrição

Com o método CompareLex é possível comparar duas strings lexicograficamente.

## Lista de parâmetros

### Parâmetro: `String1`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** A primeira string a comparar

### Parâmetro: `String2`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** A segunda string a comparar

### Parâmetro: `Diferenciar maiúsculas/minúsculas`

- **Tipo de dado:** `bool`
- **Opcional:** sim
- **Valor padrão:** True
- **Descrição:** Indica se deve ser feita distinção entre maiúsculas e minúsculas (True) ou não (False).

### Valor de retorno

- **Tipo de dado:** `dword`
- **Descrição:** Devolve um valor < 0 se a primeira string for lexicograficamente menor que a segunda. Devolve um valor = 0 se as duas strings forem lexicograficamente iguais. Devolve um valor > 0 se a primeira string for lexicograficamente maior que a segunda.

## Exemplo

```
$
// Verificar se "abc" e "abc" são lexicograficamente iguais
v[1] = String.CompareLex("abc", "abc");         /
// Verificar se "abc" e "def" são lexicograficamente iguais
v[2] = String.CompareLex("abc", "def");         /
// Verificar se "abc" e "123" são lexicograficamente iguais
v[3] = String.CompareLex("abc", "123");         /
// Verificar se "abc" e "ABC" são lexicograficamente iguais
v[4] = String.CompareLex("abc", "ABC");         /
// Verificar se "abc" e "ABC" são lexicograficamente iguais
v[5] = String.CompareLex("abc", "ABC", False);  /
```

## Observações adicionais

O uso desse método também é possível (como em todos os métodos que devolvem valores) nas condições de ramificações If, mas em comparações lexicográficas com distinção de maiúsculas/minúsculas recomenda-se, de modo geral, usar os operadores de comparação.

```
$
// Comparação com este método
if(String.CompareLex("Wort", "Zahl") == 0)
  // …
elseif(String.CompareLex("Wort", "Zahl") < 0)
  // …
elseif(String.CompareLex("Wort", "Zahl") > 0)
```

```
  // …
endif;

// Comparações totalmente equivalentes com os operadores
if("Wort" == "Zahl")
  // …
elseif("Wort" < "Zahl")
  // …
elseif("Wort" > "Zahl")
  // …
endif;
```

## Veja também

`String.Compare()`

MÉTODO

# Método: String.Fill()

## Sintaxe

```
String.Fill(Ausdruck, Anzahl, [Valor intermediário])
```

## Descrição

Com o método Fill é possível concatenar uma string repetidamente.

## Lista de parâmetros

### Parâmetro: `Expressão`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** A expressão que deve ser repetida

### Parâmetro: `Quantidade`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 0
- **Descrição:** A quantidade de vezes que a string deve ser repetida

### Parâmetro: `Valor intermediário`

- **Tipo de dado:** `string`
- **Opcional:** sim
- **Valor padrão:** Leerstring
- **Descrição:** Indica um valor a ser copiado entre cada dois valores

### Valor de retorno

- **Tipo de dado:** `string`

- **Descrição:** A expressão concatenada e repetida

## Exemplo

```
$
// Repetir a string "Repetição" três vezes
a[1] = String.Fill("Wiederholung", 3);       // "RepetiçãoRepetiçãoRepetição"
// Repetir a string "Repetição" três vezes, com espaço
a[2] = String.Fill("Wiederholung", 3, " ");  // "Repetição Repetição Repetição"
```

## Erros possíveis

`DESTINYERROR_MEMORY`, `DESTINYERROR_VALUE`

MÉTODO

# Método: String.Format()

## Sintaxe

```
String.Format(Formatierungsstring, Zahl, [Dezimal-
Komma])
```

## Descrição

Com o método Format é possível formatar um número através de uma string. A notação do formato é parecida com a do MessageLink.

## Propriedades

- **Depende do idioma:** sim

## Lista de parâmetros

### Parâmetro: `String de formatação`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** Indica o formato da conversão. Para converter um número inteiro, usa-se no início da string de formatação o caractere "D" (de Dword). Depois desse caractere, informa-se a quantidade mínima de dígitos. Já para números de ponto flutuante, usa-se um "F" (de Float) como primeiro caractere. Depois dele também pode vir a quantidade mínima de dígitos da parte inteira. Além disso, depois dessa indicação pode-se colocar um ponto seguido da quantidade exata de dígitos da parte decimal.

### Parâmetro: `Número`

- **Tipo de dado:** `dword-float`
- **Opcional:** não
- **Descrição:** Indica o número que deve ser formatado.

### Parâmetro: `Vírgula decimal`

- **Tipo de dado:** `bool`
- **Opcional:** sim
- **Valor padrão:** Convert.DecimalComma
- **Descrição:** Indica se a vírgula é usada como separador decimal (True) ou o ponto (False). Se esse valor for informado, o padrão de `Convert.DecimalComma` é ignorado, e esse método deixa de depender do idioma.

### Valor de retorno

- **Tipo de dado:** `string`
- **Descrição:** O número como string formatada

## Exemplo

No exemplo a seguir, parte-se do princípio de que o idioma atual está configurado como alemão.

```
$
// Formatar o número 1 como inteiro e exibir o resultado
a[1] = String.Format("D", 1);              // "1"
// Formatar o número 5, com no mínimo 4 dígitos, como inteiro
a[2] = String.Format("D4", 5);             // "0005"
// Formatar o número -11, com no mínimo 0 dígitos, como inteiro
a[3] = String.Format("D0", -11);           // "-11"
// Formatar o número 7 como ponto flutuante e
a[4] = String.Format("F", 7);              // "7,00"
// Formatar o número 9 como ponto flutuante, com no mínimo 3
a[5] = String.Format("F3", 9);             // "009,00"
// Formatar o número 10 como ponto flutuante, com no mínimo
a[6] = String.Format("F1.2", 10);          // "10,00"
// Formatar o número Pi como ponto flutuante, com no mínimo
a[7] = String.Format("F0.3", Math.Pi);     // "3,142"
// Formatar o número 1,25 explicitamente no formato em inglês
a[8] = String.Format("F", 1.25, False);    // "1.25"
// Formatar o número 1,25 explicitamente no formato em alemão
a[9] = String.Format("F", 1.25, True);     // "1,25"
```

## Erros possíveis

`DESTINYERROR_MEMORY`, `DESTINYERROR_VALUE`

## Veja também

`Convert.DecimalComma`, `Math.Pi`

MÉTODO

# Método: String.Length()

## Sintaxe

```
String.Length(Texto)
```

## Descrição

Com o método Length é possível saber o comprimento de uma string.

## Lista de parâmetros

### Parâmetro: `Texto`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** O texto cujo comprimento deve ser determinado.

### Valor de retorno

- **Tipo de dado:** `dword`
- **Descrição:** O comprimento da string

## Exemplo

```
$
// Atribuir a v[1] o comprimento de uma string fixa
v[1] = String.Length("12345");        // 5
// Atribuir a v[2] o comprimento de uma string guardada
a[1] = "Dies ist'n langer Text!";
v[2] = String.Length(a[1]);           // 23
```

## Veja também

`String.Pos()`, `String.SubStr()`

MÉTODO

# Método: String.LTrim()

## Sintaxe

```
String.LTrim(Text, [Caractere])
```

## Descrição

Com o método LTrim é possível remover espaços da borda esquerda de uma string. Se o parâmetro opcional Caracteres for informado, também é possível remover vários caracteres diferentes da borda esquerda.

## Lista de parâmetros

### Parâmetro: `Texto`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** O texto cujos espaços à esquerda devem ser removidos

### Parâmetro: `Caractere`

- **Tipo de dado:** `string`
- **Opcional:** sim
- **Valor padrão:** Leerzeichen
- **Descrição:** Indica os caracteres que devem ser removidos da borda esquerda da string

### Valor de retorno

- **Tipo de dado:** `string`
- **Descrição:** O texto reduzido

## Exemplo

```
$
// Remover os espaços da borda esquerda de um texto
a[1] = String.LTrim("   Text   ");
// Remover sinais de pontuação da borda esquerda de um texto
a[2] = String.LTrim("!.Satzzeichen;?", ".?!;");
```

## Erros possíveis

`DESTINYERROR_MEMORY`

## Veja também

`String.RTrim()`, `String.Trim()`

MÉTODO

# Método: String.MonthName()

## Sintaxe

```
String.MonthName(Monat, Kurzform, [Idioma])
```

## Descrição

Com o método MonthName é possível converter um número no nome de um mês. A saída pode ser tanto por extenso (por exemplo, "Fevereiro") quanto abreviada (por exemplo, "Fev").

## Propriedades

- **Depende do idioma:** sim

## Lista de parâmetros

### Parâmetro: `Mês`

- **Tipo de dado:** `dword`
- **Opcional:** não

- **Faixa de valores:** 1 ... 12
- **Descrição:** O número do mês (1 = janeiro, 2 = fevereiro, 3 = março, 4 = abril, 5 = maio, 6 = junho, 7 = julho, 8 = agosto, 9 = setembro, 10 = outubro, 11 = novembro, 12 = dezembro). Para isso podem ser usadas as `constantes de mês`

### Parâmetro: `Forma abreviada`

- **Tipo de dado:** `bool`
- **Opcional:** não
- **Descrição:** Indica se a saída deve ser abreviada (True) ou não (False).

### Parâmetro: `Idioma`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** Destiny.Language
- **Faixa de valores:** 0 ... 1
- **Descrição:** Indica o índice do idioma. Para isso podem ser usadas as `constantes de idioma`.

### Valor de retorno

- **Tipo de dado:** `string`
- **Descrição:** O nome do mês conforme a formatação indicada

## Exemplo

No exemplo a seguir, parte-se do princípio de que o idioma atual está configurado como alemão.

```
$
// Obter o nome do mês de março e exibir o resulta
a[1] = String.MonthName(MONTH_MARCH);
// Obter o nome do mês de março por extenso
a[2] = String.MonthName(MONTH_MARCH, True);
// Obter o nome do mês de março em inglês
a[3] = String.MonthName(MONTH_MARCH, False, LANGU
// Obter o nome do mês de março abreviado em inglês
a[4] = String.MonthName(MONTH_MARCH, True, LANGUA
// Obter o nome do mês da data atual
a[5] = String.MonthName(Time.Month);
```

## Observações adicionais

As formas abreviadas e por extenso em inglês dos nomes de mês devolvidas por essa função são compatíveis com requisições HTTP (RFC-2616).

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Time.Month`, `String.WeekdayName()`, `Monatskonstanten`, `Sprachkonstanten`

MÉTODO

# Método: String.Ord()

## Sintaxe

```
String.Ord(Text, [Posição])
```

## Descrição

Com o método Ord é possível saber o código ASCII de um caractere dentro de uma string. Se o parâmetro opcional "Posição" não for informado, é obtido sempre o código ASCII do primeiro caractere.

## Lista de parâmetros

### Parâmetro: `Texto`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** A string que contém o caractere

### Parâmetro: `Posição`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 0
- **Faixa de valores:** 0 ... (String-Länge - 1)
- **Descrição:** A posição dentro da string

### Valor de retorno

- **Tipo de dado:** `byte`
- **Descrição:** O código ASCII do caractere

## Exemplo

```
$
// Guardar em v[1] o código ASCII do primeiro caractere
v[1] = String.Ord("test");  // 0x74 = 116 (corresp
// Guardar em v[2] o código ASCII do terceiro caractere
v[2] = String.Ord("test");  // 0x73 = 115 (corresp
```

## Erros possíveis

`DESTINYERROR_STRINGRANGE`

## Veja também

`String.Chr()`

MÉTODO

# Método: String.Pos()

## Sintaxe

```
String.Pos(Ausdruck, Suchwert, [Startposition,
[Groß-/Kleinbuchstaben, [Para trás]]])
```

## Descrição

Com o método Pos é possível saber a posição de uma substring dentro de outra string. A busca começa na posição inicial indicada. Se a posição inicial opcional não for informada, a busca começa no início da string pesquisada.

## Lista de parâmetros

### Parâmetro: `Expressão`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** A expressão em que se vai pesquisar

### Parâmetro: `Valor buscado`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** O valor buscado dentro da expressão

### Parâmetro: `Posição inicial`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** 0
- **Faixa de valores:** -(String-Länge) ... 0 ... (String-Länge - 1)
- **Descrição:** A posição inicial dentro da expressão, a partir da qual a busca é feita. Se for informado um valor negativo, ele se refere ao fim da string.

### Parâmetro: `Maiúsculas/minúsculas`

- **Tipo de dado:** `bool`
- **Opcional:** sim
- **Valor padrão:** True
- **Descrição:** Indica se a busca deve distinguir maiúsculas de minúsculas (True) ou não (False).

### Parâmetro: `Para trás`

- **Tipo de dado:** `bool`
- **Opcional:** sim
- **Valor padrão:** False
- **Descrição:** Indica se a busca é feita de trás para frente (True) ou da frente para trás (False).

### Valor de retorno

- **Tipo de dado:** `dword`
- **Descrição:** A posição encontrada dentro da string pesquisada, ou -1 caso o valor buscado não seja encontrado

## Exemplo

```
$
// A posição da palavra "Banane" dentro do
v[1] = String.Pos("Apfel Banane Orange", "Banane"
// A posição da palavra "Erdbeere" dentro d
v[2] = String.Pos("Apfel Banane Orange", "Erdbeer
// A posição dos caracteres "an" dentro da
v[3] = String.Pos("Apfel Banane Orange", "an")
// A posição dos caracteres "an" dentro da
v[4] = String.Pos("Apfel Banane Orange", "an", 8)
// A posição do caractere "a" dentro da
v[5] = String.Pos("Apfel Banane Orange", "a")
// A posição do caractere "a" (sem distinguir
v[6] = String.Pos("Apfel Banane Orange", "a", 0,
// A posição do caractere "e" dentro da
v[7] = String.Pos("Apfel Banane Orange", "e")
// A posição do caractere "e" dentro da
v[8] = String.Pos("Apfel Banane Orange", "e", 0,
```

## Erros possíveis

`DESTINYERROR_STRINGRANGE`

## Veja também

`String.SubStr()`, `String.Replace()`

MÉTODO

# Método: String.Replace()

## Sintaxe

```
String.Replace(Ausdruck, Suchwert, Ersetzungswert,
[Diferenciar maiúsculas/minúsculas])
```

## Descrição

Com o método Replace é possível substituir todas as ocorrências de uma substring dentro de uma expressão por outra substring.

## Lista de parâmetros

### Parâmetro: `Expressão`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** A expressão com a substring a buscar

### Parâmetro: `Valor buscado`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** A substring a buscar

### Parâmetro: `Valor de substituição`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** O valor pelo qual as ocorrências encontradas serão substituídas

### Parâmetro: `Diferenciar maiúsculas/minúsculas`

- **Tipo de dado:** `bool`
- **Opcional:** sim
- **Valor padrão:** True
- **Descrição:** Indica se a busca deve distinguir maiúsculas de minúsculas (True) ou não (False).

### Valor de retorno

- **Tipo de dado:** `string`
- **Descrição:** A expressão em que todas as ocorrências do valor buscado foram substituídas pelo valor de substituição

## Exemplo

```
$
// Em um texto, substituir a palavra "Orange" por "Banane"
a[1] = String.Replace("Orangensaft presst man am
// Em um texto, substituir a palavra "Orange" por "Banane"
a[2] = String.Replace("Orangensaft presst man am
```

## Erros possíveis

`DESTINYERROR_MEMORY`

## Veja também

`String.SubStr()`, `String.Pos()`

MÉTODO

# Método: String.Reverse()

## Sintaxe

```
String.Reverse(Expressão)
```

## Descrição

Com o método Reverse é possível inverter o conteúdo de uma string.

## Lista de parâmetros

### Parâmetro: `Expressão`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** A expressão cujo conteúdo deve ser invertido

### Valor de retorno

- **Tipo de dado:** `string`
- **Descrição:** A expressão com o conteúdo invertido

## Exemplo

```
$
// Inverter o texto "sträwkcüR" e guardar em a[1]
a[1] = String.Reverse("strawkcüR");  // "Rückwärts"
```

## Erros possíveis

`DESTINYERROR_MEMORY`

MÉTODO

# Método: String.RTrim()

## Sintaxe

```
String.RTrim(Text, [Caractere])
```

## Descrição

Com o método RTrim é possível remover espaços da borda direita de uma string. Se o parâmetro opcional Caracteres for informado, também é possível remover vários caracteres diferentes da borda direita.

## Lista de parâmetros

### Parâmetro: `Texto`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** O texto cujos espaços à direita devem ser removidos

### Parâmetro: `Caractere`

- **Tipo de dado:** `string`
- **Opcional:** sim
- **Valor padrão:** Leerzeichen
- **Descrição:** Indica os caracteres que devem ser removidos da borda direita da string

### Valor de retorno

- **Tipo de dado:** `string`
- **Descrição:** O texto reduzido

## Exemplo

```
$
// Remover os espaços da borda direita de um texto
a[1] = String.RTrim("   Text   ");
// Remover sinais de pontuação da borda direita de um texto
a[2] = String.RTrim("!.Satzzeichen;?", ".?!;");
```

## Erros possíveis

`DESTINYERROR_MEMORY`

## Veja também

`String.LTrim()`, `String.Trim()`

MÉTODO

# Método: String.SubStr()

## Sintaxe

```
String.SubStr(Ausdruck, Startposition, [Comprimento])
```

## Descrição

Com o método SubStr é possível devolver uma substring de uma string. A substring começa na posição inicial indicada e tem o comprimento indicado. Se nenhum comprimento for informado, ou for informado um valor negativo, é devolvido todo o restante da string a partir da posição indicada. Se o comprimento informado for maior que o restante da string, é devolvido todo o restante da string. Se a posição inicial for um valor negativo, o comprimento da string é somado a ela, de modo que, nesse caso, a posição inicial se refere à borda direita da string.

## Lista de parâmetros

### Parâmetro: `Expressão`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** A expressão com a substring

### Parâmetro: `Posição inicial`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** -(String-Länge) ... 0 ... (String-Länge - 1)

- **Descrição:** A posição inicial dentro da expressão. Se for informado um valor negativo, ele é subtraído do comprimento da string.

### Parâmetro: `Comprimento`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** -1
- **Descrição:** O comprimento dos caracteres a copiar (-1 para o restante da string)

### Valor de retorno

- **Tipo de dado:** `string`
- **Descrição:** A substring correspondente

## Exemplo

```
$
// Os primeiros 3 caracteres (aqui, a parte esquerda) d
a[1] = String.SubStr("0123456789", 0, 3);   // "012"
// Os últimos 3 caracteres (aqui, a parte direita) d
a[2] = String.SubStr("0123456789", -3);     // "789"
// Os 4 caracteres a partir do quarto caractere (aqui
a[3] = String.SubStr("0123456789", 3, 4);   // "3456"
```

## Erros possíveis

`DESTINYERROR_MEMORY`, `DESTINYERROR_STRINGRANGE`

## Veja também

`String.Length()`, `String.Pos()`, `String.Replace()`

MÉTODO

# Método: String.ToLower()

## Sintaxe

```
String.ToLower(Ausdruck, [Caractere especial])
```

## Descrição

Com o método ToLower é possível converter todas as letras (A - Z) de uma string em minúsculas. Se o parâmetro opcional de caracteres especiais estiver definido como True, os caracteres especiais da tabela ASCII estendida (iso-8859-1) também são convertidos em minúsculas.

## Lista de parâmetros

### Parâmetro: `Expressão`

- **Tipo de dado:** `string`

- **Opcional:** não
- **Descrição:** A expressão cujas letras devem ser convertidas em minúsculas

### Parâmetro: `Caractere especial`

- **Tipo de dado:** `bool`
- **Opcional:** sim
- **Valor padrão:** false
- **Descrição:** Indica se caracteres especiais como Ä, Ö, Ü, Ç... também devem ser convertidos em minúsculas

### Valor de retorno

- **Tipo de dado:** `string`
- **Descrição:** A expressão com todas as letras convertidas em minúsculas

## Exemplo

```
$
// Converter todas as letras em minúsculas e
a[1] = String.ToLower("HINWEIS: SÄFTE UND BRÖTCHE
// Converter todas as letras e caracteres especiais em minúsculas
a[2] = String.ToLower("HINWEIS: SÄFTE UND BRÖTCHE
```

## Erros possíveis

`DESTINYERROR_MEMORY`

## Veja também

`String.ToUpper()`

MÉTODO

# Método: String.ToUpper()

## Sintaxe

```
String.ToUpper(Ausdruck, [Caractere especial])
```

## Descrição

Com o método ToUpper é possível converter todas as letras (a - z) de uma string em maiúsculas. Se o parâmetro opcional de caracteres especiais estiver definido como True, os caracteres especiais da tabela ASCII estendida (iso-8859-1) também são convertidos em maiúsculas.

## Lista de parâmetros

### Parâmetro: `Expressão`

- **Tipo de dado:** `string`
- **Opcional:** não

- **Descrição:** A expressão cujas letras devem ser convertidas em maiúsculas

### Parâmetro: `Caractere especial`

- **Tipo de dado:** `bool`
- **Opcional:** sim
- **Valor padrão:** false
- **Descrição:** Indica se caracteres especiais como ä, ö, ü, ç... também devem ser convertidos em maiúsculas

### Valor de retorno

- **Tipo de dado:** `string`
- **Descrição:** A expressão com todas as letras convertidas em maiúsculas

## Exemplo

```
$
// Converter todas as letras em maiúsculas e
a[1] = String.ToUpper("Amperé ist die Einheit der
// Converter todas as letras e caracteres especiais em maiúsculas
a[2] = String.ToUpper("Amperé ist die Einheit der
// Caso especial para a Alemanha*: exibir o texto corretamente
a[3] = String.Replace(String.ToUpper("Amperé ist
```

(* Para que suíços/austríacos também entendam o trocadilho: a palavra alemã "Extrawurst" — literalmente "salsicha extra" — é uma metáfora usada na Alemanha para "sair da linha e querer algo só seu, diferente dos demais".)

## Erros possíveis

`DESTINYERROR_MEMORY`

## Veja também

`String.ToLower()`, `String.Replace()`

MÉTODO

# Método: String.Trim()

## Sintaxe

```
String.Trim(Text, [Caractere])
```

## Descrição

Com o método Trim é possível remover espaços das bordas esquerda e direita de uma string. Se o parâmetro opcional Caracteres for informado, também é possível remover vários caracteres diferentes das bordas esquerda e direita.

## Lista de parâmetros

### Parâmetro: `Texto`

- **Tipo de dado:** `string`
- **Opcional:** não
- **Descrição:** O texto cujos espaços no início e no fim devem ser removidos

### Parâmetro: `Caractere`

- **Tipo de dado:** `string`
- **Opcional:** sim
- **Valor padrão:** Leerzeichen
- **Descrição:** Indica os caracteres que devem ser removidos das bordas esquerda e direita da string

### Valor de retorno

- **Tipo de dado:** `string`
- **Descrição:** O texto reduzido

## Exemplo

```
$
// Remover os espaços das bordas esquerda e direita de um
a[1] = String.Trim("   Text   ");
// Remover sinais de pontuação das bordas esquerda e direita de um
a[2] = String.Trim("!.Satzzeichen;?", ".?!;");
```

## Erros possíveis

`DESTINYERROR_MEMORY`

## Veja também

`String.LTrim()`, `String.RTrim()`

MÉTODO

# Método: String.WeekdayName()

## Sintaxe

```
String.WeekdayName(Wochentag, [Kurzform, [Idioma]])
```

## Descrição

Com o método WeekdayName é possível converter um número no nome de um dia da semana. A saída pode ser tanto por extenso (por exemplo, "Sábado") quanto abreviada (por exemplo, "Sáb").

## Propriedades

- **Depende do idioma:** sim

## Lista de parâmetros

### Parâmetro: `Dia da semana`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** 0 ... 6
- **Descrição:** O número do dia da semana (0 = domingo, 1 = segunda-feira, 2 = terça-feira, 3 = quarta-feira, 4 = quinta-feira, 5 = sexta-feira, 6 = sábado). Para isso podem ser usadas as `constantes de dia da semana`.

### Parâmetro: `Forma abreviada`

- **Tipo de dado:** `bool`
- **Opcional:** sim
- **Valor padrão:** False
- **Descrição:** Indica se a saída deve ser abreviada (True) ou não (False).

### Parâmetro: `Idioma`

- **Tipo de dado:** `dword`
- **Opcional:** sim
- **Valor padrão:** Destiny.Language
- **Faixa de valores:** 0 ... 1
- **Descrição:** Indica o índice do idioma. Para isso podem ser usadas as `constantes de idioma`.

### Valor de retorno

- **Tipo de dado:** `string`
- **Descrição:** O nome do dia da semana conforme a formatação indicada

## Exemplo

No exemplo a seguir, parte-se do princípio de que o idioma atual está configurado como alemão.

```
$
// Obter o nome do dia da semana para terça-feira e
a[1] = String.WeekdayName(WEEKDAY_TUESDAY);
// Obter o nome do dia da semana para terça-feira abreviado
a[2] = String.WeekdayName(WEEKDAY_TUESDAY, True);
// Obter o nome do dia da semana para terça-feira em inglês
a[3] = String.WeekdayName(WEEKDAY_TUESDAY, False,
// Obter o nome do dia da semana para terça-feira abreviado em
a[4] = String.WeekdayName(WEEKDAY_TUESDAY, True,
// Obter o nome do dia da semana da data atual
a[5] = String.WeekdayName(Time.Weekday);
```

## Observações adicionais

As formas abreviadas em inglês dos nomes de dia devolvidas por essa função são compatíveis com requisições HTTP (RFC-2616).

## Erros possíveis

`DESTINYERROR_VALUE`

## Veja também

`Time.Weekday`, `String.MonthName()`, `Wochentagskonstanten`, `Sprachkonstanten`

OBJETO

# Objeto: Time

## Sintaxe

```
Time
```

## Descrição

O objeto Time serve para descobrir a hora/data atual do computador.

## Exemplo

```
$
// Montar um texto com a data e hora atuais do
// Resulta, por exemplo, em: "Hoje é quinta-feira, dia
a[1] = "Heute ist .. " .. String.WeekdayName(Time
                          String.Format("D2", Tim
                          String.Format("D2", Tim
                          String.Format("D4", Tim
                          String.Format("D2", Tim
                          String.Format("D2", Tim
                          String.Format("D2", Tim
                          String.Format("D3", Tim
```

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Day` | Propriedade | Com a propriedade Day é possível ler o dia atual do mês. |
| `Hour` | Propriedade | Com a propriedade Hour é possível ler a hora atual no formato de 24 horas. |
| `Millisecond` | Propriedade | Com a propriedade Millisecond é possível ler o milissegundo atual. |
| `Minute` | Propriedade | Com a propriedade Minute é possível ler o minuto atual. |
| `Month` | Propriedade | Com a propriedade Month é possível ler o mês atual. |
| `Second` | Propriedade | Com a propriedade Second é possível ler o segundo atual. |
| `Tick` | Propriedade | Com a propriedade Tick é possível ler a quantidade de milissegundos desde que o computador foi iniciado. |
| `Weekday` | Propriedade | Com a propriedade Weekday é possível ler o dia da semana atual. |
| `Year` | Propriedade | Com a propriedade Year é possível ler o ano atual. |

## Veja também

`String.WeekdayName()`, `String.Format()`

PROPRIEDADE

# Propriedade: Time.Day

## Sintaxe

```
Time.Day
```

## Descrição

Com a propriedade Day é possível ler o dia atual do mês.

## Propriedades

- **Tipo de dado:** `word`
- **Somente leitura:** sim

## Exemplo

```
$
// Descobrir o dia atual do mês e guardar em v[1]
v[1] = Time.Day;
// Obter o dia atual no formato de dois dígitos, como
a[1] = String.Format("D2", Time.Day);      // Resulta
```

## Veja também

`Time`, `Time.Month`, `Time.Year`, `String.Format()`

PROPRIEDADE

# Propriedade: Time.Hour

## Sintaxe

```
Time.Hour
```

## Descrição

Com a propriedade Hour é possível ler a hora atual no formato de 24 horas.

## Propriedades

- **Tipo de dado:** `word`
- **Somente leitura:** sim

## Exemplo

```
$
// Descobrir a hora atual e guardar em v[1]
v[1] = Time.Hour;
// Obter a hora atual no formato de dois dígitos, como
a[1] = String.Format("D2", Time.Hour);     // Resulta
```

## Veja também

`Time`, `Time.Minute`, `Time.Second`, `Time.Millisecond`, `String.Format()`

PROPRIEDADE

# Propriedade: Time.Millisecond

## Sintaxe

```
Time.Millisecond
```

## Descrição

Com a propriedade Millisecond é possível ler o milissegundo atual.

## Propriedades

- **Tipo de dado:** `word`
- **Somente leitura:** sim

## Exemplo

```
$
// Descobrir o milissegundo atual e guardar em v[1
v[1] = Time.Millisecond;
// Obter o milissegundo atual no formato de três dígitos
a[1] = String.Format("D3", Time.Millisecond);   /
```

## Veja também

`Time`, `Time.Hour`, `Time.Minute`, `Time.Second`, `Time.Tick`, `String.Format()`

PROPRIEDADE

# Propriedade: Time.Minute

## Sintaxe

```
Time.Minute
```

## Descrição

Com a propriedade Minute é possível ler o minuto atual.

## Propriedades

- **Tipo de dado:** `word`
- **Somente leitura:** sim

## Exemplo

```
$
// Descobrir o minuto atual e guardar em v[1]
v[1] = Time.Minute;
// Obter o minuto atual no formato de dois dígitos, como
a[1] = String.Format("D2", Time.Minute);   // Resulta
```

## Veja também

`Time`, `Time.Hour`, `Time.Second`, `Time.Millisecond`, `String.Format()`

PROPRIEDADE

# Propriedade: Time.Month

## Sintaxe

```
Time.Month
```

## Descrição

Com a propriedade Month é possível ler o mês atual. 1 = janeiro, 2 = fevereiro, 3 = março, 4 = abril, 5 = maio, 6 = junho, 7 = julho, 8 = agosto, 9 = setembro, 10 = outubro, 11 = novembro, 12 = dezembro. Para isso também podem ser usadas as `constantes de mês`.

## Propriedades

- **Tipo de dado:** `word`
- **Somente leitura:** sim

## Exemplo

```
$
// Descobrir o mês atual e guardar em v[1]
v[1] = Time.Month;
// Obter o mês atual no formato de dois dígitos, como
a[1] = String.Format("D2", Time.Month);    // Resulta
// Converter o mês atual no nome do mês
a[2] = String.MonthName(Time.Month);       // Resulta
```

## Veja também

`Time`, `Time.Day`, `Time.Year`, `String.Format()`, `String.MonthName()`

PROPRIEDADE

# Propriedade: Time.Second

## Sintaxe

```
Time.Second
```

## Descrição

Com a propriedade Second é possível ler o segundo atual.

## Propriedades

- **Tipo de dado:** `word`
- **Somente leitura:** sim

## Exemplo

```
$
// Descobrir o segundo atual e guardar em v[1]
v[1] = Time.Second;
// Obter o segundo atual no formato de dois dígitos, como
a[1] = String.Format("D2", Time.Second);   // Resulta
```

## Veja também

`Time`, `Time.Hour`, `Time.Minute`, `Time.Millisecond`, `String.Format()`

PROPRIEDADE

# Propriedade: Time.Tick

## Sintaxe

```
Time.Tick
```

## Descrição

Com a propriedade Tick é possível ler a quantidade de milissegundos desde que o computador foi iniciado. Essa propriedade é útil para fazer medições de tempo.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

- **Volátil:** sim

## Exemplo

```
$
// Descobrir o tick atual e guardar em v[1]
v[1] = Time.Tick;
// Executar alguma ação qualquer que consuma tempo
for(v[3] = 0; v[3] < 1000000; v[3]++)
  Math.Sin(123, DEG);
next;
// Descobrir o tick atual e guardar em v[2]
v[2] = Time.Tick;
// Calcular a diferença entre os dois momentos
// Resulta, no notebook de desenvolvimento de Bananen-Joe, em: "Isso
a[1] = "Eine million Mal Sinus von 123 berechnen
```

## Veja também

`Time`, `Time.Hour`, `Time.Minute`, `Time.Second`, `Time.Tick`, `Math.Sin()`

PROPRIEDADE

# Propriedade: Time.Weekday

## Sintaxe

```
Time.Weekday
```

## Descrição

Com a propriedade Weekday é possível ler o dia da semana atual. São usados os seguintes valores: 0 = domingo, 1 = segunda-feira, 2 = terça-feira, 3 = quarta-feira, 4 = quinta-feira, 5 = sexta-feira, 6 = sábado. Para isso também podem ser usadas as `constantes de dia da semana`.

## Propriedades

- **Tipo de dado:** `word`
- **Somente leitura:** sim

## Exemplo

```
$
// Ler o dia da semana atual e convertê-lo em um
a[1] = String.WeekdayName(Time.Weekday);   // Resulta
```

## Veja também

`Time`, `Time.Day`, `String.WeekdayName()`

PROPRIEDADE

# Propriedade: Time.Year

## Sintaxe

```
Time.Year
```

## Descrição

Com a propriedade Year é possível ler o ano atual.

## Propriedades

- **Tipo de dado:** `word`
- **Somente leitura:** sim

## Exemplo

```
$
// Descobrir o ano atual e guardar em v[1]
v[1] = Time.Year;
// Obter o ano atual no formato de quatro dígitos,
a[1] = String.Format("D4", Time.Year);     // Resulta
// Verificar se o ano atual é bissexto*
s[1] = Logic.If(Time.Year % 4 == 0, Logic.If(Time
```

- O ano bissexto é definido da seguinte forma: um ano é bissexto quando o ano é divisível por 4 sem resto, mas não por 100. Se for divisível por 100, então também precisa ser divisível por 400 sem resto.

## Veja também

`Time`, `Time.Day`, `Time.Month`, `String.Format()`, `Logic.If()`

OBJETO

# Objeto: v

## Sintaxe

```
v
```

## Descrição

O objeto v serve para tornar acessíveis as variáveis (números inteiros) do RPG_RT. A diferença em relação ao objeto d é que os Dwords do objeto d são gerenciados pela Destiny.dll, enquanto as variáveis (Dwords) do objeto v são gerenciadas pelo RPG_RT.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `Count` | Propriedade | Devolve a quantidade de variáveis no objeto v. |
| `v[]` | Propriedade | Permite o acesso a uma determinada variável do objeto v. |

## Veja também

`a`, `d`, `f`, `s`

PROPRIEDADE

# Propriedade: v.Count

## Sintaxe

```
v.Count
```

## Descrição

Devolve a quantidade de variáveis no objeto `v`. Isso pode ser usado para fazer um laço sobre todas as variáveis.

> **Informação:** A quantidade de variáveis disponível só é definida em tempo de execução do jogo. Somente quando uma variável (ou uma com Variable-Id maior) é acessada pela primeira vez, ela é criada (com o valor 0). Esse comportamento vem do RPG_RT e foi mantido assim no DestinyPatch. Por esse motivo, o objeto `v` também não tem um método ReAlloc.

Exemplo:

Se, depois de iniciado o jogo, a terceira variável for acessada, então a partir desse momento existem três variáveis ( `v.Count` = 3). Se depois disso a quinta variável for acessada, então a partir desse momento existem cinco variáveis (`v.Count` = 5). Se depois disso a terceira variável for acessada de novo, continuam existindo cinco variáveis ( `v.Count` = 5).

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** sim

## Exemplo

```
$
// Definir todas as variáveis como 777
for(d[1] = 1; d[1] <= v.Count; d[1]++)
  v[d[1]] = 777;
next;
```

## Veja também

`v`, `v[]`

PROPRIEDADE

# Propriedade: v[]

## Sintaxe

```
v[Variável-Id]
```

## Descrição

Permite o acesso a uma determinada variável do objeto `v`.

## Propriedades

- **Tipo de dado:** `dword`
- **Somente leitura:** não

## Lista de parâmetros

### Parâmetro: `Variável-Id`

- **Tipo de dado:** `dword`
- **Opcional:** não
- **Faixa de valores:** >= 1 && <= 100000000
- **Descrição:** A Variable-Id da variável. Esse valor precisa ser maior/igual a 1. Se for informada uma Variable-Id maior que `v.Count`, a quantidade de variáveis aumenta automaticamente.

## Exemplo

```
$
// Definir o valor da primeira variável como 5
v[1] = 5;
```

## Erros possíveis

`DESTINYERROR_ARRAYBOUND`

## Veja também

`v`, `v.Count`

# Lista de grupos de constantes

## Descrição

As constantes são usadas para dar nomes simbólicos a valores usados com frequência. Por exemplo, em vez de `Command.SetWeather(1, 2)` pode-se escrever `Command.SetWeather(WEATHER_RAIN, WEATHERSTRENGTH_NORMAL)`. Assim não é preciso saber explicitamente que o 1 do primeiro parâmetro significa o clima "chuva", e assim por diante.

De modo geral, é altamente recomendável usar constantes em vez de números escritos diretamente, para que o significado do valor se mantenha.

## Itens contidos

| Nome | Tipo | Resumo |
|---|---|---|
| `(Des)ativação` | Constantes | O grupo de constantes (De-)aktivierung contém as constantes para indicar se uma propriedade deve ser ativada ou desativada. |
| `Página exibida no EnterHeroName` | Constantes | O grupo de constantes "Página exibida no EnterHeroName" contém as constantes para indicar qual página é exibida primeiro ao mostrar a tela de EnterHeroName. |
| `Exibição dos gráficos do sistema` | Constantes | O grupo de constantes "Exibição dos gráficos do sistema" contém as constantes para indicar os modos de exibição do gráfico do sistema (fundo das caixas de mensagem, menus etc.). |
| `Tipos de atributo` | Constantes | O grupo de constantes Attributtyp contém as constantes para indicar os tipos de atributo. |
| `Modos de chamada de CommonEvents` | Constantes | O grupo de constantes "Modos de chamada de CommonEvents" contém as constantes para indicar os modos de chamada de um CommonEvent. |
| `Peças de equipamento` | Constantes | O grupo de constantes Ausrüstungsteile contém as constantes para indicar as peças de equipamento em comandos de equipamento. |
| `Comandos de movimento` | Constantes | O grupo de constantes Bewegungsbefehle contém as constantes para indicar comandos de movimento. |
| `Transições de imagem` | Constantes | O grupo de constantes Bildübergang contém as constantes para indicar as transições de imagem em determinados eventos. |
| `Direções de olhar dos eventos` | Constantes | O grupo de constantes "Direções de olhar dos eventos" contém as constantes para indicar as direções em que um evento olha ou se move. |
| `Valores booleanos` | Constantes | O grupo de constantes "valores booleanos" contém as constantes para indicar os dois valores booleanos. |
| `Modos de arquivo` | Constantes | O grupo de constantes Dateimodi contém as constantes para indicar os modos de arquivo. |
| `Posições em arquivo` | Constantes | O grupo de constantes Dateipositionen contém as constantes para indicar os pontos de referência das posições em arquivo. |
| `Atributos do sistema de arquivos` | Constantes | O grupo de constantes Dateisystemattribute contém as constantes para indicar atributos do sistema de arquivos (atributos de arquivos/pastas). |
| `Tipos de dados` | Constantes | O grupo de constantes Datentyp contém as constantes para indicar tipos de dado. |
| `Erros do Destiny` | Constantes | O grupo de constantes Destiny-Fehler contém as constantes para identificar os erros do Destiny, usadas por exemplo para comparar com Error[].Code. |
| `Níveis de efeito` | Constantes | O grupo de constantes Effektstufen contém as constantes para indicar níveis de efeito. |

| Nome | Tipo | Resumo |
|---|---|---|
| `Valores de evento` | Constantes | O grupo de constantes Event-Werte contém as constantes para indicar valores de evento. |
| `Posições do face` | Constantes | O grupo de constantes Face-Positionen contém as constantes para indicar posições do face. |
| `Frames de eventos` | Constantes | O grupo de constantes "Frames de eventos" contém as constantes para indicar os frames de movimento de um evento. |
| `Grupo inteiro em comandos de status` | Constantes | O grupo de constantes "Grupo inteiro em comandos de status" contém a constante para indicar o grupo inteiro em comandos de status. |
| `Valores de herói` | Constantes | O grupo de constantes Helden-Werte contém as constantes para indicar valores de herói. |
| `Tipos de música de fundo` | Constantes | O grupo de constantes Hintergrundmusik-Arten contém as constantes para indicar tipos de música de fundo em comandos de música de fundo. |
| `Tipos de item` | Constantes | O grupo de constantes Item-Typen contém as constantes para indicar os diferentes tipos de item. |
| `Usos de item` | Constantes | O grupo de constantes Item-Verwendungen contém a constante para indicar a quantidade de usos de um item. |
| `Valores de item` | Constantes | O grupo de constantes Item-Werte contém as constantes para indicar valores de item. |
| `Áreas de efeito de item` | Constantes | O grupo de constantes Item-Wirkungsbereiche contém as constantes para indicar os alcances de efeito de itens. |
| `Tipos de condição` | Constantes | O grupo de constantes Konditionsarten contém as constantes para indicar os tipos de condição. |
| `Restrições de condição` | Constantes | O grupo de constantes Konditionsbeschränkungen contém as constantes para indicar as restrições impostas por certas condições em combate. |
| `Meses` | Constantes | O grupo de constantes Monate contém as constantes para indicar os meses. |
| `Formatos de mensagem` | Constantes | O grupo de constantes Nachrichtenformat contém as constantes para indicar formatos de mensagem. |
| `Posições de mensagem` | Constantes | O grupo de constantes Nachrichtenpositionen contém as constantes para indicar posições de mensagem. |
| `Ações de imagem (Picture)` | Constantes | O grupo de constantes Picture-Aktionen contém as constantes para indicar as ações de uma Picture. |
| `Fontes` | Constantes | O grupo de constantes Schriftarten contém as constantes para indicar as fontes do RPG_RT. |
| `Tipos de scroll` | Constantes | O grupo de constantes Scroll-Arten contém as constantes para indicar os tipos de scroll do mapa. |
| `Socket: seleção do servidor` | Constantes | O grupo de constantes "Socket: seleção do servidor" contém a constante para indicar a seleção de socket do servidor. |
| `Socket: flags` | Constantes | O grupo de constantes Socket-Flags contém as constantes para indicar flags de socket. |
| `Status do socket` | Constantes | O grupo de constantes Socketstatus contém as constantes para indicar o status de um socket. |
| `Tipos de socket` | Constantes | O grupo de constantes Sockettypen contém as constantes para indicar tipos de socket. |
| `Caracteres especiais` | Constantes | O grupo de constantes Sonderzeichen contém as constantes para caracteres especiais que não podem ser digitados diretamente no código-fonte. |

| Nome | Tipo | Resumo |
| --- | --- | --- |
| `Outros valores` | Constantes | O grupo de constantes Sonstige-Werte contém as constantes para indicar outros valores diversos. |
| `Tipos de som` | Constantes | O grupo de constantes Sound-Arten contém as constantes para indicar tipos de som em comandos de som. |
| `Eventos especiais` | Constantes | O grupo de constantes "Eventos especiais" contém as constantes para indicar eventos especiais do RPG Maker. |
| `Idiomas` | Constantes | O grupo de constantes Sprachen contém as constantes para indicar os idiomas suportados pela Destiny.dll. |
| `Valores de status` | Constantes | O grupo de constantes Status-Werte contém as constantes para indicar valores de status em comandos de valor de status. |
| `Operações com switches` | Constantes | O grupo de constantes Switch-Operationen contém as constantes para indicar operações com switches. |
| `Eventos de tecla` | Constantes | O grupo de constantes Tastenereignisse contém as constantes para indicar os eventos ao pressionar uma tecla. |
| `Áreas de efeito de técnicas` | Constantes | O grupo de constantes Technik-Wirkungsbereiche contém as constantes para indicar os alcances de efeito de técnicas. |
| `Tipos de técnica` | Constantes | O grupo de constantes Technikarten contém as constantes para indicar os diferentes tipos de técnica. |
| `Cores de texto` | Constantes | O grupo de constantes Textfarben contém as constantes para indicar as cores de texto. |
| `Operações com variáveis` | Constantes | O grupo de constantes Variablen-Operationen contém as constantes para indicar operações com variáveis. |
| `Variantes de exibição da tela` | Constantes | O grupo de constantes "Variantes de exibição da tela" contém as constantes para indicar as variantes de exibição da tela. |
| `Variantes de limpeza da tela` | Constantes | O grupo de constantes "Variantes de limpeza da tela" contém as constantes para indicar as variantes de apagar a tela. |
| `Códigos virtuais de tecla` | Constantes | O grupo de constantes "Códigos virtuais de tecla" contém as constantes para identificação das teclas do teclado. |
| `Tipos de valor` | Constantes | O grupo de constantes Wertetyp contém as constantes para indicar tipos de valor. |
| `Tipos de clima` | Constantes | O grupo de constantes Wetterarten contém as constantes para indicar os diferentes tipos de clima. |
| `Intensidades de clima` | Constantes | O grupo de constantes Wetterstärken contém as constantes para indicar as intensidades dos efeitos de clima. |
| `Formato de ângulo` | Constantes | O grupo de constantes Winkelformat contém as constantes para indicar os diferentes formatos de ângulo. |
| `Dias da semana` | Constantes | O grupo de constantes Wochentage contém as constantes para indicar os dias da semana. |

CONSTANTES

# Grupo de constantes: (Des)ativação

## Descrição

O grupo de constantes (De-)aktivierung contém as constantes para indicar se uma propriedade deve ser ativada ou desativada. Essas constantes são usadas junto com o objeto `Command`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `DISABLE` | `dword` | 0 | Desativar |
| `ENABLE` | `dword` | 1 | Ativar |

## Veja também

`Command.EnableTeleport()`, `Command.EnableEscape()`, `Command.EnableSaveMenu()`, `Command.EnableMenu()`

CONSTANTES

# Grupo de constantes: Página exibida no EnterHeroName

## Descrição

O grupo de constantes "Página exibida no EnterHeroName" contém as constantes para indicar qual página é exibida primeiro ao mostrar a tela de EnterHeroName. Essas constantes são usadas junto com o objeto `Command`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `ENTRYPAGE_HIRAGANA` | `dword` | 0 | A página Hiragana é exibida primeiro (em conjuntos de caracteres ocidentais, costuma ser a página de letras) |
| `ENTRYPAGE_KATAKANA` | `dword` | 1 | A página Katakana é exibida primeiro (em conjuntos de caracteres ocidentais, costuma ser a página de símbolos) |

## Veja também

`Command.EnterHeroName()`

CONSTANTES

# Grupo de constantes: Exibição dos gráficos do sistema

## Descrição

O grupo de constantes "Exibição dos gráficos do sistema" contém as constantes para indicar os modos de exibição do gráfico do sistema (fundo das caixas de mensagem, menus etc.). Essas constantes são usadas junto com o objeto `Command`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `DISPLAY_STRETCH` | `dword` | 0 | O gráfico do sistema é exibido esticado |
| `DISPLAY_REPEAT` | `dword` | 1 | O gráfico do sistema é exibido repetido (em mosaico) |

## Veja também

`Command.SetSystemGraphic()`

CONSTANTES

# Grupo de constantes: Tipos de atributo

## Descrição

O grupo de constantes Attributtyp contém as constantes para indicar os tipos de atributo. Essas constantes são usadas junto com o objeto `Attribute`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `ATTRIBUTETYPE_WEAPON` | `dword` | 0 | O atributo é um atributo de arma |
| `ATTRIBUTETYPE_MAGIC` | `dword` | 1 | O atributo é um atributo de magia |

## Veja também

`Attribute`

CONSTANTES

# Grupo de constantes: Modos de chamada de CommonEvents

## Descrição

O grupo de constantes "Modos de chamada de CommonEvents" contém as constantes para indicar os modos de chamada de um CommonEvent. Essas constantes são usadas junto com o objeto `CommonEvent[]`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `CALLMODE_CALL` | `dword` | 5 | O CommonEvent só é iniciado quando é chamado |
| `CALLMODE_PARALLELPROCESS` | `dword` | 4 | O CommonEvent roda como processo paralelo |
| `CALLMODE_AUTOSTART` | `dword` | 3 | O CommonEvent roda como processo de início automático |

## Veja também

`CommonEvent[].CallMode`

CONSTANTES

# Grupo de constantes: Peças de equipamento

## Descrição

O grupo de constantes Ausrüstungsteile contém as constantes para indicar as peças de equipamento em comandos de equipamento. Essas constantes são usadas junto com o objeto `Command`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `EQUIPMENT_ALL` | `dword` | 0 | Todo o equipamento de um herói |
| `EQUIPMENT_WEAPON` | `dword` | 1 | A arma de um herói |
| `EQUIPMENT_SHIELD` | `dword` | 2 | O escudo de um herói |
| `EQUIPMENT_ARMOR` | `dword` | 3 | A armadura de um herói |
| `EQUIPMENT_HELMET` | `dword` | 4 | O elmo de um herói |
| `EQUIPMENT_ACCESSORY` | `dword` | 5 | O acessório de um herói |

## Veja também

`Command.EquipItem()`, `Command.DropEquipment()`

CONSTANTES

# Grupo de constantes: Comandos de movimento

## Descrição

O grupo de constantes Bewegungsbefehle contém as constantes para indicar comandos de movimento. Essas constantes são usadas junto com os objetos `Command` e `Event`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `MC_STEPUP` | `dword` | 0 | Um passo para cima |
| `MC_STEPRIGHT` | `dword` | 1 | Um passo para a direita |
| `MC_STEPDOWN` | `dword` | 2 | Um passo para baixo |
| `MC_STEPLEFT` | `dword` | 3 | Um passo para a esquerda |
| `MC_STEPRIGHTUP` | `dword` | 4 | Um passo na diagonal para a direita e para cima |
| `MC_STEPRIGHTDOWN` | `dword` | 5 | Um passo na diagonal para a direita e para baixo |
| `MC_STEPLEFTDOWN` | `dword` | 6 | Um passo na diagonal para a esquerda e para baixo |
| `MC_STEPLEFTUP` | `dword` | 7 | Um passo na diagonal para a esquerda e para cima |
| `MC_STEPRANDOM` | `dword` | 8 | Um passo em uma direção aleatória (cima, esquerda, direita ou baixo) |
| `MC_STEPTOWARDHERO` | `dword` | 9 | Um passo em direção ao herói (cima, esquerda, direita ou baixo) |
| `MC_STEPAWAYFROMHERO` | `dword` | 10 | Um passo para longe do herói (cima, esquerda, direita ou baixo) |
| `MC_STEPFORWARD` | `dword` | 11 | Um passo para frente (na direção atual em que o evento olha) |
| `MC_LOOKUP` | `dword` | 12 | Olhar para cima |
| `MC_LOOKRIGHT` | `dword` | 13 | Olhar para a direita |
| `MC_LOOKDOWN` | `dword` | 14 | Olhar para baixo |
| `MC_LOOKLEFT` | `dword` | 15 | Olhar para a esquerda |
| `MC_TURNRIGHT` | `dword` | 16 | Girar 90° para a direita |
| `MC_TURNLEFT` | `dword` | 17 | Girar 90° para a esquerda |
| `MC_TURN180` | `dword` | 18 | Girar 180° |
| `MC_RIGHTLEFTTURN` | `dword` | 19 | Girar 90° aleatoriamente para a esquerda ou para a direita |
| `MC_LOOKRANDOM` | `dword` | 20 | Olhar em uma direção aleatória (cima, esquerda, direita ou baixo) |
| `MC_LOOKTOWARDHERO` | `dword` | 21 | Olhar em direção ao herói |
| `MC_LOOKAWAYFROMHERO` | `dword` | 22 | Olhar para longe do herói |
| `MC_WAIT` | `dword` | 23 | Aguardar um instante |
| `MC_STARTJUMPING` | `dword` | 24 | Começar a pular |
| `MC_STOPJUMPING` | `dword` | 25 | Parar de pular |
| `MC_FIXDIRECTION` | `dword` | 26 | Fixar a direção em que o evento olha |
| `MC_RELEASEDIRECTION` | `dword` | 27 | Liberar a direção em que o evento olha |
| `MC_INCREASESPEED` | `dword` | 28 | Aumentar a velocidade de movimento |
| `MC_DECREASESPEED` | `dword` | 29 | Diminuir a velocidade de movimento |

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `MC_INCREASEFREQUENCY` | `dword` | 30 | Aumentar a frequência de movimento |
| `MC_DECREASEFREQUENCY` | `dword` | 31 | Diminuir a frequência de movimento |
| `MC_SETSWITCHON` | `dword` | 32 | Ligar um switch |
| `MC_SETSWITCHOFF` | `dword` | 33 | Desligar um switch |
| `MC_SETCHAR` | `dword` | 34 | Definir o character gráfico (char) do evento movido |
| `MC_PLAYSOUND` | `dword` | 35 | Reproduzir um som |
| `MC_STARTPHASING` | `dword` | 36 | Ativar o movimento sem colisão |
| `MC_STOPPHASING` | `dword` | 37 | Desativar o movimento sem colisão |
| `MC_FIXANIMATION` | `dword` | 38 | Desativar a animação de movimento |
| `MC_RELEASEANIMATION` | `dword` | 39 | Ativar a animação de movimento |
| `MC_INCREASETRANSPARENCY` | `dword` | 40 | Aumentar a transparência |
| `MC_DECREASETRANSPARENCY` | `dword` | 41 | Diminuir a transparência |

## Veja também

`Command.MoveEvent()`, `Event[].MoveRoute[].Command`

CONSTANTES

# Grupo de constantes: Transições de imagem

## Descrição

O grupo de constantes Bildübergang contém as constantes para indicar as transições de imagem em determinados eventos. Essas constantes são usadas junto com o objeto `Command`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `TRANSITION_TELEPORTERASE` | `dword` | 0 | A transição ao apagar a tela antes de uma teletransportação |
| `TRANSITION_TELEPORTSHOW` | `dword` | 1 | A transição ao exibir a tela depois de uma teletransportação |
| `TRANSITION_BATTLESTARTERASE` | `dword` | 2 | A transição ao apagar a tela antes de um combate |
| `TRANSITION_BATTLESTARTSHOW` | `dword` | 3 | A transição ao exibir a tela no início de um combate |
| `TRANSITION_BATTLEENDERASE` | `dword` | 4 | A transição ao apagar a tela no fim de um combate |
| `TRANSITION_BATTLEENDSHOW` | `dword` | 5 | A transição ao exibir a tela depois de um combate |

## Veja também

`Command.SetScreenTransition()`

CONSTANTES

# Grupo de constantes: Direções de olhar dos eventos

## Descrição

O grupo de constantes "Direções de olhar dos eventos" contém as constantes para indicar as direções em que um evento olha ou se move. Essas constantes são usadas junto com o objeto `Event`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `DIR_UP` | `dword` | 0 | Cima |
| `DIR_RIGHT` | `dword` | 1 | Direita |
| `DIR_DOWN` | `dword` | 2 | Baixo |
| `DIR_LEFT` | `dword` | 3 | Esquerda |
| `DIR_RIGHT_UP` | `dword` | 4 | Direita-cima (somente em `Event[].DirMove` e `Event[].CanMoveTo()`) |
| `DIR_UP_RIGHT` | `dword` | 4 | Direita-cima (somente em `Event[].DirMove` e `Event[].CanMoveTo()`) |
| `DIR_RIGHT_DOWN` | `dword` | 5 | Direita-baixo (somente em `Event[].DirMove` e `Event[].CanMoveTo()`) |
| `DIR_DOWN_RIGHT` | `dword` | 5 | Direita-baixo (somente em `Event[].DirMove` e `Event[].CanMoveTo()`) |
| `DIR_LEFT_DOWN` | `dword` | 6 | Esquerda-baixo (somente em `Event[].DirMove` e `Event[].CanMoveTo()`) |
| `DIR_DOWN_LEFT` | `dword` | 6 | Esquerda-baixo (somente em `Event[].DirMove` e `Event[].CanMoveTo()`) |
| `DIR_LEFT_UP` | `dword` | 7 | Esquerda-cima (somente em `Event[].DirMove` e `Event[].CanMoveTo()`) |
| `DIR_UP_LEFT` | `dword` | 7 | Esquerda-cima (somente em `Event[].DirMove` e `Event[].CanMoveTo()`) |

## Veja também

`Event[].DirMove`, `Event[].DirLook`, `Event[].CanMoveTo()`

CONSTANTES

# Grupo de constantes: Valores booleanos

## Descrição

O grupo de constantes "valores booleanos" contém as constantes para indicar os dois valores booleanos. Essas constantes pertencem à linguagem de script e podem ser usadas com qualquer objeto; são usadas sobretudo com `s[]` e com o objeto `Logic`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `false` | `bool` | 0 | Falso / Desligado / 0 |
| `true` | `bool` | 1 | Verdadeiro / Ligado / 1 |

## Veja também

`s[]`, `Logic`

CONSTANTES

# Grupo de constantes: Modos de arquivo

## Descrição

O grupo de constantes Dateimodi contém as constantes para indicar os modos de arquivo. Essas constantes são usadas junto com o objeto `File[]`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `FILE_READ` | `dword` | 1 | O arquivo é aberto para leitura |
| `FILE_WRITE` | `dword` | 2 | O arquivo é aberto para escrita |
| `FILE_APPEND` | `dword` | 6 | O arquivo é aberto para escrita, com o ponteiro de arquivo posicionado no fim do arquivo ao abrir |

## Veja também

`File[].Open()`

CONSTANTES

# Grupo de constantes: Posições em arquivo

## Descrição

O grupo de constantes Dateipositionen contém as constantes para indicar os pontos de referência das posições em arquivo. Essas constantes são usadas junto com o objeto `File[]`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `FILEPOS_BEGIN` | `dword` | 0 | A posição indicada se refere ao início do arquivo |
| `FILEPOS_CURRENT` | `dword` | 1 | A posição indicada se refere à posição atual do ponteiro de arquivo |
| `FILEPOS_END` | `dword` | 2 | A posição indicada se refere ao fim do arquivo |

## Veja também

`File[].SetFilePointer()`

CONSTANTES

# Grupo de constantes: Atributos do sistema de arquivos

## Descrição

O grupo de constantes Dateisystemattribute contém as constantes para indicar atributos do sistema de arquivos (atributos de arquivos/pastas). Essas constantes são usadas junto com o objeto `Directory`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `FILE_ATTRIBUTE_NORMAL` | `dword` | 128 | Um arquivo normal |
| `FILE_ATTRIBUTE_SYSTEM` | `dword` | 4 | Um arquivo/uma pasta que pertence ao sistema |
| `FILE_ATTRIBUTE_HIDDEN` | `dword` | 2 | Um arquivo/uma pasta oculto(a) |
| `FILE_ATTRIBUTE_ARCHIVE` | `dword` | 32 | Um arquivo/uma pasta com o bit de arquivamento definido |
| `FILE_ATTRIBUTE_DIRECTORY` | `dword` | 16 | Uma pasta |
| `FILE_ATTRIBUTE_READONLY` | `dword` | 1 | Um arquivo/uma pasta somente leitura |

## Veja também

`Directory.GetAttributes()`, `Directory.SetAttributes()`

CONSTANTES

# Grupo de constantes: Tipos de dados

## Descrição

O grupo de constantes Datentyp contém as constantes para indicar tipos de dado. Essas constantes são usadas junto com os objetos `Client` e `File`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `TYPE_VARIABLE` | `dword` | 1 | O tipo de dado é uma variável (como não se distingue entre variáveis e Dwords, isso é tratado como TYPE_DWORD) |
| `TYPE_SWITCH` | `dword` | 2 | O tipo de dado é um switch |
| `TYPE_BOOL` | `dword` | 2 | O tipo de dado é um valor booleano |
| `TYPE_DWORD` | `dword` | 3 | O tipo de dado é um Dword |
| `TYPE_DOUBLE` | `dword` | 4 | O tipo de dado é um Float |
| `TYPE_FLOAT` | `dword` | 4 | O tipo de dado é um Float |
| `TYPE_STRING` | `dword` | 5 | O tipo de dado é uma String |
| `TYPE_BYTE` | `dword` | 6 | O tipo de dado é um Byte |
| `TYPE_WORD` | `dword` | 7 | O tipo de dado é um Word |

## Veja também

`Client[].Raw.Send()`, `Client[].Raw.Recv()`, `Client[].Destiny.GetRecvType()`, `File[].WriteRawData()`, `File[].ReadRawData()`

CONSTANTES

# Grupo de constantes: Erros do Destiny

## Descrição

O grupo de constantes Destiny-Fehler contém as constantes para identificar os erros do Destiny, usadas por exemplo para comparar com `Error[].Code`. Cada constante corresponde a um dos erros listados na lista de erros.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `DESTINYERROR_UNKNOWN` | `dword` | 1 | Constante numérica do erro “Erro desconhecido”. |
| `DESTINYERROR_SYNTAX` | `dword` | 1 | Constante numérica do erro “Erro de sintaxe”. |
| `DESTINYERROR_NOVALUE` | `dword` | 2 | Constante numérica do erro “Nenhum valor informado”. |
| `DESTINYERROR_UNKNOWNNAME` | `dword` | 3 | Constante numérica do erro “Identificador desconhecido”. |
| `DESTINYERROR_CONVERT` | `dword` | 4 | Constante numérica do erro “Erro de conversão”. |
| `DESTINYERROR_READONLY` | `dword` | 5 | Constante numérica do erro “Escrita em valor somente leitura”. |
| `DESTINYERROR_ARRAYBOUND` | `dword` | 6 | Constante numérica do erro “Limites do array excedidos”. |
| `DESTINYERROR_RANGE` | `dword` | 7 | Constante numérica do erro “Valor fora da faixa permitida”. |
| `DESTINYERROR_MEMORY` | `dword` | 8 | Constante numérica do erro “Erro no gerenciamento de memória”. |
| `DESTINYERROR_VALUE` | `dword` | 9 | Constante numérica do erro “Valor inválido”. |
| `DESTINYERROR_BINARYFLOAT` | `dword` | 10 | Constante numérica do erro “Operação binária com valor de ponto flutuante”. |
| `DESTINYERROR_CALCSWITCH` | `dword` | 11 | Constante numérica do erro “Operação matemática com Switch”. |
| `DESTINYERROR_CALCSTRING` | `dword` | 12 | Constante numérica do erro “Operação matemática com String”. |
| `DESTINYERROR_FLOATERROR` | `dword` | 13 | Constante numérica do erro “Erro de ponto flutuante”. |
| `DESTINYERROR_FLOATLENGTH` | `dword` | 14 | Constante numérica do erro “Número decimal longo demais”. |
| `DESTINYERROR_DIVISIONBYZERO` | `dword` | 15 | Constante numérica do erro “Divisão por zero”. |
| `DESTINYERROR_STRINGFORMAT` | `dword` | 16 | Constante numérica do erro “Formato de string inválido”. |
| `DESTINYERROR_STRINGRANGE` | `dword` | 17 | Constante numérica do erro “Posição de texto inválida”. |
| `DESTINYERROR_PICTURE` | `dword` | 18 | Constante numérica do erro “Imagem não inicializada”. |
| `DESTINYERROR_PIXELRANGE` | `dword` | 19 | Constante numérica do erro “Pixel fora da área da imagem”. |
| `DESTINYERROR_SAMEPICTURE` | `dword` | 20 | Constante numérica do erro “Imagem de origem e destino iguais com sobreposição”. |

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| DESTINYERROR_PALETTERANGE | dword | 21 | Constante numérica do erro “Entrada fora da paleta de 256 cores”. |
| DESTINYERROR_SOCKETSTARTUP | dword | 22 | Constante numérica do erro “Falha na inicialização do socket”. |
| DESTINYERROR_NOFREESOCKET | dword | 23 | Constante numérica do erro “Nenhum socket livre disponível”. |
| DESTINYERROR_CANTCREATESOCKET | dword | 24 | Constante numérica do erro “Não foi possível criar o socket”. |
| DESTINYERROR_SOCKETSTILLOPEN | dword | 25 | Constante numérica do erro “Socket não fechado”. |
| DESTINYERROR_SOCKETNOTOPEN | dword | 26 | Constante numérica do erro “Socket não aberto”. |
| DESTINYERROR_CANTCONNECT | dword | 27 | Constante numérica do erro “Falha na conexão”. |
| DESTINYERROR_SOCKETTYPE | dword | 28 | Constante numérica do erro “Operação inválida para este tipo de socket”. |
| DESTINYERROR_SOCKETERROR | dword | 29 | Constante numérica do erro “Erro em operação de socket”. |
| DESTINYERROR_OOB | dword | 30 | Constante numérica do erro “Out-of-Band”. |
| DESTINYERROR_STRINGTOOLONG | dword | 31 | Constante numérica do erro “String longa demais”. |
| DESTINYERROR_NOFREEFILEHANDLE | dword | 32 | Constante numérica do erro “O handle de arquivo não está livre”. |
| DESTINYERROR_CANTRESOLVEPATH | dword | 33 | Constante numérica do erro “Não foi possível resolver o caminho”. |
| DESTINYERROR_NOPERMISSION | dword | 34 | Constante numérica do erro “Sem permissão de acesso”. |
| DESTINYERROR_CANTOPENFILE | dword | 35 | Constante numérica do erro “Não foi possível abrir o arquivo”. |
| DESTINYERROR_FILENOTOPEN | dword | 36 | Constante numérica do erro “Arquivo não aberto”. |
| DESTINYERROR_CANTACCESSFILE | dword | 37 | Constante numérica do erro “Acesso a arquivo impossível”. |
| DESTINYERROR_CANTCREATEDIR | dword | 38 | Constante numérica do erro “Não foi possível criar a pasta”. |
| DESTINYERROR_CANTREMOVEDIR | dword | 39 | Constante numérica do erro “Não foi possível excluir a pasta”. |
| DESTINYERROR_CANTRENAMEFILE | dword | 40 | Constante numérica do erro “Não foi possível renomear o arquivo/pasta”. |
| DESTINYERROR_CANTCOPYFILE | dword | 41 | Constante numérica do erro “Não foi possível copiar o arquivo”. |
| DESTINYERROR_CANTDELETEFILE | dword | 42 | Constante numérica do erro “Não foi possível excluir o arquivo”. |
| DESTINYERROR_CANTREADATTRIBUTES | dword | 43 | Constante numérica do erro “Não foi possível ler os atributos”. |
| DESTINYERROR_CANTWRITEATTRIBUTES | dword | 44 | Constante numérica do erro “Não foi possível gravar os atributos”. |
| DESTINYERROR_SEARCHSTILLOPEN | dword | 45 | Constante numérica do erro “Busca já aberta”. |
| DESTINYERROR_CANTSTARTSEARCH | dword | 46 | Constante numérica do erro “Não foi possível iniciar a busca”. |
| DESTINYERROR_NOSEARCHSTARTED | dword | 47 | Constante numérica do erro “Busca não iniciada”. |
| DESTINYERROR_PROTECTED | dword | 48 | Constante numérica do erro “Função do Destiny bloqueada”. |

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `DESTINYERROR_CANTEXECUTEFILE` | `dword` | 49 | Constante numérica do erro “Execução de arquivo impossível”. |
| `DESTINYERROR_NOTENOUGHPARAMETERS` | `dword` | 50 | Constante numérica do erro “Parâmetros insuficientes”. |
| `DESTINYERROR_NOTCOMPARABLE` | `dword` | 51 | Constante numérica do erro “Não comparável”. |
| `DESTINYERROR_INVALIDSIGN` | `dword` | 52 | Constante numérica do erro “Sinal (prefixo) inválido”. |
| `DESTINYERROR_SETNOTALLOWEDHERE` | `dword` | 53 | Constante numérica do erro “O valor não pode ser gravado aqui”. |
| `DESTINYERROR_MISSINGENDIF` | `dword` | 54 | Constante numérica do erro “EndIf ausente”. |
| `DESTINYERROR_MISSINGLOOP` | `dword` | 55 | Constante numérica do erro “Loop ausente”. |
| `DESTINYERROR_MISSINGCONDITION` | `dword` | 56 | Constante numérica do erro “Condição ausente”. |
| `DESTINYERROR_MISSINGNEXT` | `dword` | 57 | Constante numérica do erro “Next ausente”. |
| `DESTINYERROR_MISSINGCASE` | `dword` | 58 | Constante numérica do erro “Case ausente”. |
| `DESTINYERROR_MISSINGENDSWITCH` | `dword` | 59 | Constante numérica do erro “EndSwitch ausente”. |
| `DESTINYERROR_BREAKNOTALLOWEDHERE` | `dword` | 60 | Constante numérica do erro “Break não é permitido aqui”. |
| `DESTINYERROR_CONTINUENOTALLOWEDHERE` | `dword` | 61 | Constante numérica do erro “Continue não é permitido aqui”. |
| `DESTINYERROR_INVALIDHANDLE` | `dword` | 62 | Constante numérica do erro “Handle inválido”. |
| `DESTINYERROR_OPERATOREXPECTED` | `dword` | 63 | Constante numérica do erro “Operador esperado”. |
| `DESTINYERROR_CANTACCESSSAVESLOT` | `dword` | 64 | Constante numérica do erro “Acesso ao slot de memória impossível”. |
| `DESTINYERROR_EXCEPTION` | `dword` | 65 | Constante numérica do erro “Exceção ocorrida”. |
| `DESTINYERROR_INVALIDLOOPOPERATIONLEVEL` | `dword` | 66 | Constante numérica do erro “Nível de operador de laço inválido”. |
| `DESTINYERROR_CONTINUEDOESNTPOINTTOLOOP` | `dword` | 67 | Constante numérica do erro “Continue não se refere a um laço”. |

## Veja também

`Error`, `Errors`, `Error[].Code`

CONSTANTES

# Grupo de constantes: Níveis de efeito

## Descrição

O grupo de constantes Effektstufen contém as constantes para indicar níveis de efeito. Essas constantes são usadas junto com o objeto `Attribute`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `EFFECT_A` | `dword` | 0 | Nível de efeito "A" |
| `EFFECT_B` | `dword` | 1 | Nível de efeito "B" |

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `EFFECT_C` | `dword` | 2 | Nível de efeito "C" |
| `EFFECT_D` | `dword` | 3 | Nível de efeito "D" |
| `EFFECT_E` | `dword` | 4 | Nível de efeito "E" |

## Veja também

`Attribute`

CONSTANTES

# Grupo de constantes: Valores de evento

## Descrição

O grupo de constantes Event-Werte contém as constantes para indicar valores de evento. Essas constantes são usadas junto com o objeto `Command`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `EVENT_MAPID` | `dword` | 0 | A Map-Id do evento. |
| `EVENT_X` | `dword` | 1 | A coordenada X do evento. |
| `EVENT_Y` | `dword` | 2 | A coordenada Y do evento. |
| `EVENT_DIRECTION` | `dword` | 3 | A direção para a qual o evento está olhando. |
| `EVENT_SCREENX` | `dword` | 4 | A coordenada X em pixels do evento, relativa à posição atual da tela. |
| `EVENT_SCREENY` | `dword` | 5 | A coordenada Y em pixels do evento, relativa à posição atual da tela. |

## Veja também

`Command.ChangeVariable()`, `Command.ChangeVariableRange()`

CONSTANTES

# Grupo de constantes: Posições do face

## Descrição

O grupo de constantes Face-Positionen contém as constantes para indicar posições do face. Essas constantes são usadas junto com o objeto `Command`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `FACEPOSITION_LEFT` | `dword` | 0 | O face é exibido na borda esquerda da caixa de mensagem. |
| `FACEPOSITION_RIGHT` | `dword` | 1 | O face é exibido na borda direita da caixa de mensagem. |

## Veja também

`Command.ShowFace()`

CONSTANTES

# Grupo de constantes: Frames de eventos

## Descrição

O grupo de constantes "Frames de eventos" contém as constantes para indicar os frames de movimento de um evento. Essas constantes são usadas junto com o objeto `Event`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `EVENTFRAME_LEFT` | `dword` | 0 | A perna do lado esquerdo está para baixo (frame da esquerda) |
| `EVENTFRAME_NORMAL` | `dword` | 1 | As duas pernas para baixo (frame do meio) |
| `EVENTFRAME_RIGHT` | `dword` | 2 | A perna do lado direito está para baixo (frame da direita) |
| `EVENTFRAME_NORMAL2` | `dword` | 3 | As duas pernas para baixo (frame do meio) |

## Veja também

`Event[].Frame`, `Event[].FrameSimple`

CONSTANTES

# Grupo de constantes: Grupo inteiro em comandos de status

## Descrição

O grupo de constantes "Grupo inteiro em comandos de status" contém a constante para indicar o grupo inteiro em comandos de status. Essa constante é usada junto com o objeto `Command`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `STATE_WHOLEPARTY` | `dword` | 0 | Informada no lugar de uma Herói-Id para indicar que a alteração de status deve valer para o grupo inteiro. |

## Veja também

`Command.AddExp()`, `Command.SubtractExp()`, `Command.AddLevel()`, `Command.SubtractLevel()`, `Command.AddAbility()`, `Command.SubtractAbility()`, `Command.AddSkill()`, `Command.RemoveSkill()`, `Command.EquipItem()`, `Command.DropEquipment()`, `Command.AddHp()`, `Command.SubtractHp()`, `Command.AddMp()`, `Command.SubtractMp()`, `Command.AddCondition()`, `Command.RemoveCondition()`, `Command.FullRecovery()`, `Command.DealDamage()`

CONSTANTES

# Grupo de constantes: Valores de herói

## Descrição

O grupo de constantes Helden-Werte contém as constantes para indicar valores de herói. Essas constantes são usadas junto com o objeto `Command`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `ACTOR_LEVEL` | `dword` | 0 | O nível atual do herói. |
| `ACTOR_EXP` | `dword` | 1 | Os pontos de experiência atuais do herói. |
| `ACTOR_HP` | `dword` | 2 | Os pontos de vida atuais do herói. |
| `ACTOR_MP` | `dword` | 3 | Os pontos de magia atuais do herói. |
| `ACTOR_MAXHP` | `dword` | 4 | Os pontos de vida máximos atuais do herói (já considerando os efeitos de equipamentos, condições etc.). |
| `ACTOR_MAXMP` | `dword` | 5 | Os pontos de magia máximos atuais do herói (já considerando os efeitos de equipamentos, condições etc.). |
| `ACTOR_ATTACK` | `dword` | 6 | Os pontos de ataque atuais do herói (já considerando os efeitos de equipamentos, condições etc.). |
| `ACTOR_DEFENSE` | `dword` | 7 | Os pontos de defesa atuais do herói (já considerando os efeitos de equipamentos, condições etc.). |
| `ACTOR_MIND` | `dword` | 8 | Os pontos de inteligência atuais do herói (já considerando os efeitos de equipamentos, condições etc.). |
| `ACTOR_AGILITY` | `dword` | 9 | Os pontos de agilidade atuais do herói (já considerando os efeitos de equipamentos, condições etc.). |
| `ACTOR_WEAPON` | `dword` | 10 | A Item-Id da arma atualmente equipada pelo herói. |
| `ACTOR_SHIELD` | `dword` | 11 | A Item-Id do escudo atualmente equipado pelo herói. |
| `ACTOR_ARMOR` | `dword` | 12 | A Item-Id da armadura atualmente equipada pelo herói. |
| `ACTOR_HELMET` | `dword` | 13 | A Item-Id do elmo atualmente equipado pelo herói. |
| `ACTOR_ACCESSORY` | `dword` | 14 | A Item-Id do acessório atualmente equipado pelo herói. |

## Veja também

`Command.ChangeVariable()`, `Command.ChangeVariableRange()`

CONSTANTES

# Grupo de constantes: Tipos de música de fundo

## Descrição

O grupo de constantes Hintergrundmusik-Arten contém as constantes para indicar tipos de música de fundo em comandos de música de fundo. Essas constantes são usadas junto com o objeto `Command`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `MUSIC_BATTLE` | `dword` | 0 | A música de fundo durante o combate |
| `MUSIC_VICTORY` | `dword` | 1 | A música de fundo durante a vitória |
| `MUSIC_INN` | `dword` | 2 | A música de fundo durante uma estadia em pousada |
| `MUSIC_BOAT` | `dword` | 3 | A música de fundo durante a viagem de barco |
| `MUSIC_SHIP` | `dword` | 4 | A música de fundo durante a viagem de navio |
| `MUSIC_AIRSHIP` | `dword` | 5 | A música de fundo durante a viagem de aeronave |
| `MUSIC_GAMEOVER` | `dword` | 6 | A música de fundo durante a tela de Game Over |

## Veja também

`Command.SetSystemMusic()`

CONSTANTES

# Grupo de constantes: Tipos de item

## Descrição

O grupo de constantes Item-Typen contém as constantes para indicar os diferentes tipos de item. Essas constantes são usadas junto com o objeto `Item[]`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `ITEMTYPE_COMMON` | `dword` | 0 | O item não pode ser usado |
| `ITEMTYPE_WEAPON` | `dword` | 1 | O item é uma arma |
| `ITEMTYPE_SHIELD` | `dword` | 2 | O item é um escudo |
| `ITEMTYPE_ARMOR` | `dword` | 3 | O item é uma armadura |
| `ITEMTYPE_HELMET` | `dword` | 4 | O item é um elmo |
| `ITEMTYPE_ACCESSORY` | `dword` | 5 | O item é um acessório |
| `ITEMTYPE_MEDICINE` | `dword` | 6 | O item é um remédio |
| `ITEMTYPE_BOOK` | `dword` | 7 | O item é um livro de ensino (ensina técnicas) |
| `ITEMTYPE_MATERIAL` | `dword` | 8 | O item é um material (altera valores de status) |

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `ITEMTYPE_UNIQUE` | `dword` | 9 | O item é um grimório (usa técnicas) |
| `ITEMTYPE_SWITCH` | `dword` | 10 | O item liga um switch ao ser usado |

## Veja também

`Item[].Type`

CONSTANTES

# Grupo de constantes: Usos de item

## Descrição

O grupo de constantes Item-Verwendungen contém a constante para indicar a quantidade de usos de um item. Essa constante é usada junto com o objeto `Item[]`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `INFINITE_USES` | `dword` | 0 | O item pode ser usado infinitas vezes |

## Veja também

`Item[].NumberOfUses`

CONSTANTES

# Grupo de constantes: Valores de item

## Descrição

O grupo de constantes Item-Werte contém as constantes para indicar valores de item. Essas constantes são usadas junto com o objeto `Command`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `ITEM_OWNED` | `dword` | 0 | Indica quantas unidades do item o grupo possui |
| `ITEM_EQUIPPED` | `dword` | 1 | Indica quantas unidades do item o grupo tem equipadas |

## Veja também

`Command.ChangeVariable()`, `Command.ChangeVariableRange()`

CONSTANTES

# Grupo de constantes: Áreas de efeito de item

## Descrição

O grupo de constantes Item-Wirkungsbereiche contém as constantes para indicar os alcances de efeito de itens. Essas constantes são usadas junto com o objeto `Item[]`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `ITEMSCOPE_SINGLEALLY` | `dword` | 0 | O item atua apenas em um único herói |
| `ITEMSCOPE_ALLALLIES` | `dword` | 1 | O item atua em todo o grupo (todos os heróis) |

## Veja também

`Item[].Medicine.Scope`

CONSTANTES

# Grupo de constantes: Tipos de condição

## Descrição

O grupo de constantes Konditionsarten contém as constantes para indicar os tipos de condição. Essas constantes são usadas junto com o objeto `Condition[]`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `CONDITIONTYPE_COMBAT` | `dword` | 0 | A condição só atua em combate |
| `CONDITIONTYPE_MOVEMENT` | `dword` | 1 | A condição atua durante o movimento no mapa, e também em combate, se for o caso |

## Veja também

`Condition[].Type`

CONSTANTES

# Grupo de constantes: Restrições de condição

## Descrição

O grupo de constantes Konditionsbeschränkungen contém as constantes para indicar as restrições impostas por certas condições em combate. Essas constantes são usadas junto com o objeto `Condition[]`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `CONDITIONLIMITATION_NONE` | `dword` | 0 | Nenhuma restrição |
| `CONDITIONLIMITATION_NOACTION` | `dword` | 1 | Nenhuma ação pode ser executada (ex.: ao dormir) |
| `CONDITIONLIMITATION_ATTACKENEMY` | `dword` | 2 | Um inimigo aleatório é atacado (ex.: "modo berserker") |
| `CONDITIONLIMITATION_ATTACKALLY` | `dword` | 3 | Um aliado aleatório é atacado (ex.: "confusão") |

## Veja também

`Condition[].ActionLimitation`

CONSTANTES

# Grupo de constantes: Meses

## Descrição

O grupo de constantes Monate contém as constantes para indicar os meses. Essas constantes são usadas junto com os objetos `Time` e `String`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `JANUARY` | `dword` | 1 | Janeiro |
| `FEBRUARY` | `dword` | 2 | Fevereiro |
| `MARCH` | `dword` | 3 | Março |
| `APRIL` | `dword` | 4 | Abril |
| `MAY` | `dword` | 5 | Maio |
| `JUNE` | `dword` | 6 | Junho |
| `JULY` | `dword` | 7 | Julho |
| `AUGUST` | `dword` | 8 | Agosto |
| `SEPTEMBER` | `dword` | 9 | Setembro |
| `OCTOBER` | `dword` | 10 | Outubro |
| `NOVEMBER` | `dword` | 11 | Novembro |
| `DECEMBER` | `dword` | 12 | Dezembro |

## Veja também

`Time.Month`, `String.MonthName()`

CONSTANTES

# Grupo de constantes: Formatos de mensagem

## Descrição

O grupo de constantes Nachrichtenformat contém as constantes para indicar formatos de mensagem. Essas constantes são usadas junto com o objeto `Command`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `MESSAGEFORMAT_NORMAL` | `dword` | 0 | As caixas de mensagem são exibidas normalmente. |
| `MESSAGEFORMAT_TRANSPARENT` | `dword` | 1 | As caixas de mensagem são exibidas de forma transparente. |

## Veja também

`Command.SetMessageStyle()`

CONSTANTES

# Grupo de constantes: Posições de mensagem

## Descrição

O grupo de constantes Nachrichtenpositionen contém as constantes para indicar posições de mensagem. Essas constantes são usadas junto com o objeto `Command`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `MESSAGEPOSITION_TOP` | `dword` | 0 | As caixas de mensagem são exibidas na borda superior da tela. |
| `MESSAGEPOSITION_MIDDLE` | `dword` | 1 | As caixas de mensagem são exibidas no meio da tela. |
| `MESSAGEPOSITION_BOTTOM` | `dword` | 2 | As caixas de mensagem são exibidas na borda inferior da tela. |

## Veja também

`Command.SetMessageStyle()`

CONSTANTES

# Grupo de constantes: Ações de imagem (Picture)

## Descrição

O grupo de constantes Picture-Aktionen contém as constantes para indicar as ações de uma Picture. Essas constantes são usadas junto com o objeto `Picture`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `ACTION_NONE` | `dword` | 0 | Nenhuma ação |
| `ACTION_ROTATION` | `dword` | 1 | A Picture gira |
| `ACTION_RIPPLE` | `dword` | 2 | A Picture ondula através de um efeito de onda |

## Veja também

`Picture[].Action`

CONSTANTES

# Grupo de constantes: Fontes

## Descrição

O grupo de constantes Schriftarten contém as constantes para indicar as fontes do RPG_RT. Essas constantes são usadas junto com o objeto `Command`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `FONT_RMG2000` | `dword` | 0 | A fonte "RMG2000" é exibida esticada |
| `FONT_RM2000` | `dword` | 1 | A fonte "RM2000" é exibida esticada |

## Veja também

`Command.SetSystemGraphic()`

CONSTANTES

# Grupo de constantes: Tipos de scroll

## Descrição

O grupo de constantes Scroll-Arten contém as constantes para indicar os tipos de scroll do mapa. Essas constantes são usadas junto com o objeto `Map`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `SCROLLTYPE_NORMAL` | `dword` | 0 | O mapa não continua rolando na borda |
| `SCROLLTYPE_VERTICAL` | `dword` | 1 | O mapa continua rolando na borda superior e inferior (vertical) |
| `SCROLLTYPE_HORIZONTAL` | `dword` | 2 | O mapa continua rolando na borda esquerda e direita (horizontal) |
| `SCROLLTYPE_BOTH` | `dword` | 3 | O mapa continua rolando em todas as bordas (horizontal e vertical) |

## Veja também

`Map.ScrollType`

CONSTANTES

# Grupo de constantes: Socket: seleção do servidor

## Descrição

O grupo de constantes "Socket: seleção do servidor" contém a constante para indicar a seleção de socket do servidor. Essa constante é usada junto com o objeto `Server`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `NEXT_FREE_SOCKET` | `dword` | -1 | Deve ser usado o próximo socket livre (se houver). Se não existir nenhum socket livre, é lançado um erro. Recomenda-se usar `Client.NextFree()` em vez dessa constante, já que esse método cria um novo socket quando necessário. |

## Veja também

`Server.Accept()`

CONSTANTES

# Grupo de constantes: Socket: flags

## Descrição

O grupo de constantes Socket-Flags contém as constantes para indicar flags de socket. Essas constantes são usadas junto com o objeto `Client`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `SOCKETFLAG_NOTBLOCKING` | `dword` | 1 | O socket não bloqueia o jogo ao estabelecer a conexão. |

## Veja também

`Client[].Connect()`

CONSTANTES

# Grupo de constantes: Status do socket

## Descrição

O grupo de constantes Socketstatus contém as constantes para indicar o status de um socket. Essas constantes são usadas junto com os objetos `Client` e `Server`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `STATE_ERROR` | `dword` | -1 | Ocorreu um erro com o socket |
| `STATE_CLOSED` | `dword` | 0 | O socket está fechado |
| `STATE_CONNECTED` | `dword` | 1 | O socket está conectado (dados podem ser recebidos/enviados) |
| `STATE_LISTENING` | `dword` | 2 | O socket aguarda conexões de entrada (modo servidor) |
| `STATE_CLOSING` | `dword` | 3 | O socket já não tem conexão, mas ainda há dados pendentes que podem ser recebidos |
| `STATE_CONNECTING` | `dword` | 4 | O socket está no processo de estabelecer uma conexão |

## Veja também

`Client[].State`, `Server.State`

CONSTANTES

# Grupo de constantes: Tipos de socket

## Descrição

O grupo de constantes Sockettypen contém as constantes para indicar tipos de socket. Essas constantes são usadas junto com os objetos `Client` e `Server`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `SOCK_DESTINY` | `dword` | 0 | O socket é um socket Destiny |
| `SOCK_RAW` | `dword` | 1 | O socket é um socket Raw |

## Veja também

`Client`, `Server`

CONSTANTES

# Grupo de constantes: Caracteres especiais

## Descrição

O grupo de constantes Sonderzeichen contém as constantes para caracteres especiais que não podem ser digitados diretamente no código-fonte. Essas constantes pertencem à linguagem de script e podem ser usadas com qualquer objeto que trabalhe com Strings.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `QUOTE` | `string` | " | O caractere de aspas duplas |
| `CRLF` | `string` | \r\n | Quebra de linha no estilo Windows (Carriage Return + Line Feed, 0x0D 0x0A) |
| `LF` | `string` | \n | Quebra de linha no estilo Linux (Line Feed, 0x0A) |
| `CR` | `string` | \r | Quebra de linha no estilo Macintosh (Carriage Return, 0x0D) |

## Veja também

`String`, `a[]`

CONSTANTES

# Grupo de constantes: Outros valores

## Descrição

O grupo de constantes Sonstige-Werte contém as constantes para indicar outros valores diversos. Essas constantes são usadas junto com o objeto `Command`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `MISC_MONEY` | `dword` | 0 | O dinheiro atual do grupo. |
| `MISC_TIMERSECONDSLEFT` | `dword` | 1 | Os segundos restantes do timer. |
| `MISC_PARTYSIZE` | `dword` | 2 | O tamanho do grupo. |
| `MISC_NUMBEROFSAVES` | `dword` | 3 | A quantidade de vezes que o jogo atual foi salvo. |
| `MISC_NUMBEROFBATTLES` | `dword` | 4 | A quantidade de combates no sistema de batalha padrão do jogo atual. |
| `MISC_NUMBEROFVICTORIES` | `dword` | 5 | A quantidade de vitórias no sistema de batalha padrão do jogo atual. |
| `MISC_NUMBEROFDEFEATS` | `dword` | 6 | A quantidade de derrotas no sistema de batalha padrão do jogo atual. |
| `MISC_NUMBEROFESCAPES` | `dword` | 7 | A quantidade de fugas no sistema de batalha padrão do jogo atual. |
| `MISC_MIDITICK` | `dword` | 8 | O tick MIDI da música de fundo atual. |

## Veja também

`Command.ChangeVariable()`, `Command.ChangeVariableRange()`

CONSTANTES

# Grupo de constantes: Tipos de som

## Descrição

O grupo de constantes Sound-Arten contém as constantes para indicar tipos de som em comandos de som. Essas constantes são usadas junto com o objeto `Command`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `SOUND_CURSOR` | `dword` | 0 | O som ao mover o cursor de seleção |
| `SOUND_DECISION` | `dword` | 1 | O som ao confirmar uma seleção |
| `SOUND_CANCEL` | `dword` | 2 | O som ao cancelar uma seleção |
| `SOUND_BUZZER` | `dword` | 3 | O som ao tentar fazer uma seleção inválida |
| `SOUND_STARTBATTLE` | `dword` | 4 | O som ao iniciar um combate |
| `SOUND_ESCAPE` | `dword` | 5 | O som ao fugir de um combate |
| `SOUND_ENEMYATTACK` | `dword` | 6 | O som ao ataque de um inimigo |

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| SOUND_ENEMYDAMAGE | dword | 7 | O som quando um inimigo sofre dano |
| SOUND_ALLYDAMAGE | dword | 8 | O som quando um membro do grupo sofre dano |
| SOUND_EVASION | dword | 9 | O som ao esquivar de um ataque |
| SOUND_ENEMYDIE | dword | 10 | O som quando um inimigo morre |
| SOUND_ITEMUSE | dword | 11 | O som ao usar um item |

## Veja também

Command.SetSystemSound()

CONSTANTES

# Grupo de constantes: Eventos especiais

## Descrição

O grupo de constantes "Eventos especiais" contém as constantes para indicar eventos especiais do RPG Maker. Essas constantes são usadas junto com os objetos Event, MapEvent e Command.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| HERO | dword | 10001 | O evento do herói |
| BOAT | dword | 10002 | O evento do barco |
| SHIP | dword | 10003 | O evento do navio |
| AIRSHIP | dword | 10004 | O evento da aeronave |
| THIS | dword | 10005 | O evento atual |

## Veja também

Event[], MapEvent, Command.FlashEvent(), Command.MoveEvent(), Command.SetVehicleChar(), Command.SetVehiclePlace(), Command.SetEventPlace(), Command.SwapTwoEvents(), Command.ShowBattleAnimation()

CONSTANTES

# Grupo de constantes: Idiomas

## Descrição

O grupo de constantes Sprachen contém as constantes para indicar os idiomas suportados pela Destiny.dll. Essas constantes são usadas junto com os objetos Destiny, Convert e String.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `LANGUAGE_GERMAN` | `dword` | 0 | As mensagens de erro, os nomes de dias da semana e os nomes de meses ficam em alemão, e o separador decimal usado é a vírgula |
| `LANGUAGE_ENGLISH` | `dword` | 1 | As mensagens de erro, os nomes de dias da semana e os nomes de meses ficam em inglês, e o separador decimal usado é o ponto |

## Veja também

`Destiny.Language`, `String.MonthName()`, `String.WeekdayName()`, `Convert.DecimalComma`

CONSTANTES

# Grupo de constantes: Valores de status

## Descrição

O grupo de constantes Status-Werte contém as constantes para indicar valores de status em comandos de valor de status. Essas constantes são usadas junto com o objeto `Command`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `ABILITY_MAXHP` | `dword` | 0 | Os pontos de vida máximos de um herói |
| `ABILITY_MAXMP` | `dword` | 1 | Os pontos de magia máximos de um herói |
| `ABILITY_ATTACK` | `dword` | 2 | Os pontos de ataque máximos de um herói |
| `ABILITY_DEFENSE` | `dword` | 3 | Os pontos de defesa máximos de um herói |
| `ABILITY_MIND` | `dword` | 4 | Os pontos de inteligência máximos de um herói |
| `ABILITY_AGILITY` | `dword` | 5 | Os pontos de agilidade máximos de um herói |

## Veja também

`Command.AddAbility()`, `Command.SubtractAbility()`

CONSTANTES

# Grupo de constantes: Operações com switches

## Descrição

O grupo de constantes Switch-Operationen contém as constantes para indicar operações com switches. Essas constantes são usadas junto com o objeto `Command`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `OPERATION_ON` | `dword` | 0 | O switch é ligado. |

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `OPERATION_OFF` | `dword` | 1 | O switch é desligado. |
| `OPERATION_TOGGLE` | `dword` | 2 | O switch é alternado (de ligado para desligado, e vice-versa). |

## Veja também

`Command.ChangeSwitch()`, `Command.ChangeSwitchRange()`

CONSTANTES

# Grupo de constantes: Eventos de tecla

## Descrição

O grupo de constantes Tastenereignisse contém as constantes para indicar os eventos ao pressionar uma tecla. Essas constantes são usadas junto com o objeto `Keyboard`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `KEYEVENTF_KEYDOWN` | `dword` | 0 | A tecla foi pressionada |
| `KEYEVENTF_KEYUP` | `dword` | 1 | A tecla foi solta |

## Veja também

`Keyboard.SetKeyState()`

CONSTANTES

# Grupo de constantes: Áreas de efeito de técnicas

## Descrição

O grupo de constantes Technik-Wirkungsbereiche contém as constantes para indicar os alcances de efeito de técnicas. Essas constantes são usadas junto com o objeto `Skill[]`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `SKILLSCOPE_SINGLEENEMY` | `dword` | 0 | A técnica atua apenas em um único inimigo |
| `SKILLSCOPE_ALLENEMIES` | `dword` | 1 | A técnica atua em todos os inimigos |
| `SKILLSCOPE_SELF` | `dword` | 2 | A técnica atua no próprio herói que a está usando |
| `SKILLSCOPE_SINGLEALLY` | `dword` | 3 | A técnica atua em um único herói |
| `SKILLSCOPE_ALLALLIES` | `dword` | 4 | A técnica atua em todo o grupo (todos os heróis) |

## Veja também

`Skill[].Normal.Scope`

CONSTANTES

# Grupo de constantes: Tipos de técnica

## Descrição

O grupo de constantes Technikarten contém as constantes para indicar os diferentes tipos de técnica. Essas constantes são usadas junto com o objeto `Skill[]`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `SKILLTYPE_NORMAL` | `dword` | 0 | A técnica é uma técnica de combate normal |
| `SKILLTYPE_TELEPORT` | `dword` | 1 | A técnica executa uma teletransportação |
| `SKILLTYPE_ESCAPE` | `dword` | 2 | A técnica executa uma fuga em combate |
| `SKILLTYPE_SWITCH` | `dword` | 3 | A técnica liga um switch ao ser usada |

## Veja também

`Skill[].Type`

CONSTANTES

# Grupo de constantes: Cores de texto

## Descrição

O grupo de constantes Textfarben contém as constantes para indicar as cores de texto. Essas constantes são usadas junto com o objeto `Condition[]`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `TEXTCOLOR_DEFAULT` | `dword` | 0 | Cor padrão |
| `TEXTCOLOR_BLUE` | `dword` | 1 | Azul |
| `TEXTCOLOR_LIGHTORANGE` | `dword` | 2 | Laranja claro |
| `TEXTCOLOR_BLACK` | `dword` | 3 | Preto |
| `TEXTCOLOR_YELLOW` | `dword` | 4 | Amarelo |
| `TEXTCOLOR_RED` | `dword` | 5 | Vermelho |
| `TEXTCOLOR_PURPLE` | `dword` | 6 | Lilás |
| `TEXTCOLOR_MAGENTA` | `dword` | 7 | Magenta |
| `TEXTCOLOR_ORANGE` | `dword` | 8 | Laranja |
| `TEXTCOLOR_GREEN` | `dword` | 9 | Verde |
| `TEXTCOLOR_LIGHTBLUE` | `dword` | 10 | Azul claro |
| `TEXTCOLOR_LIGHTRED` | `dword` | 11 | Vermelho claro |

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `TEXTCOLOR_OCHER` | `dword` | 12 | Ocre (amarelo escuro) |
| `TEXTCOLOR_VIOLET` | `dword` | 13 | Violeta |
| `TEXTCOLOR_GOLD` | `dword` | 14 | Dourado |
| `TEXTCOLOR_SEAGREEN` | `dword` | 15 | Verde-mar |
| `TEXTCOLOR_DARKPURPLE` | `dword` | 16 | Lilás escuro |
| `TEXTCOLOR_OCEANBLUE` | `dword` | 17 | Azul-oceano |
| `TEXTCOLOR_TEAL` | `dword` | 18 | Turquesa |
| `TEXTCOLOR_BROWN` | `dword` | 19 | Marrom |

## Veja também

`Condition[].Color`

CONSTANTES

# Grupo de constantes: Operações com variáveis

## Descrição

O grupo de constantes Variablen-Operationen contém as constantes para indicar operações com variáveis. Essas constantes são usadas junto com o objeto `Command`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `OPERATION_SET` | `dword` | 0 | A variável é definida com um valor |
| `OPERATION_ADD` | `dword` | 1 | Um valor é somado à variável |
| `OPERATION_SUBTRACT` | `dword` | 2 | Um valor é subtraído da variável |
| `OPERATION_MULTIPLY` | `dword` | 3 | A variável é multiplicada por um valor |
| `OPERATION_DIVIDE` | `dword` | 4 | A variável é dividida por um valor |
| `OPERATION_MODULO` | `dword` | 5 | A variável é dividida por um valor, e o resto não divisível dessa divisão é guardado nessa variável |

## Veja também

`Command.ChangeVariable()`, `Command.ChangeVariableRange()`

CONSTANTES

# Grupo de constantes: Variantes de exibição da tela

## Descrição

O grupo de constantes "Variantes de exibição da tela" contém as constantes para indicar as variantes de exibição da tela. Essas constantes são usadas junto com o objeto `Command`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `SHOW_FADEIN` | `dword` | 0 | A tela é exibida com fade-in |
| `SHOW_RANDOMBLOCKS` | `dword` | 1 | A tela se preenche aleatoriamente com blocos do novo conteúdo |
| `SHOW_RANDOMBLOCKSUP` | `dword` | 2 | A tela se preenche aleatoriamente com blocos do novo conteúdo, de baixo para cima |
| `SHOW_RANDOMBLOCKSDOWN` | `dword` | 3 | A tela se preenche aleatoriamente com blocos do novo conteúdo, de cima para baixo |
| `SHOW_BLINDOPEN` | `dword` | 4 | A tela se preenche com listras horizontais do novo conteúdo |
| `SHOW_STRIPEUPDOWN` | `dword` | 5 | A tela é preenchida simultaneamente de cima e de baixo, cada uma com uma faixa do novo conteúdo |
| `SHOW_STRIPELEFTRIGHT` | `dword` | 6 | A tela é preenchida simultaneamente da esquerda e da direita, cada uma com uma faixa do novo conteúdo |
| `SHOW_OUTSIDEINSIDE` | `dword` | 7 | A tela se preenche em forma de retângulo, de fora para dentro |
| `SHOW_INSIDEOUTSIDE` | `dword` | 8 | A tela se preenche em forma de retângulo, de dentro para fora |
| `SHOW_SCROLLDOWN` | `dword` | 9 | A tela é deslocada para dentro vindo de cima |
| `SHOW_SCROLLUP` | `dword` | 10 | A tela é deslocada para dentro vindo de baixo |
| `SHOW_SCROLLRIGHT` | `dword` | 11 | A tela é deslocada para dentro vindo da esquerda |
| `SHOW_SCROLLLEFT` | `dword` | 12 | A tela é deslocada para dentro vindo da direita |
| `SHOW_COMBINEUPDOWN` | `dword` | 13 | A tela se forma a partir de duas partes horizontais do novo conteúdo, vindas simultaneamente de cima e de baixo |
| `SHOW_COMBINELEFTRIGHT` | `dword` | 14 | A tela se forma a partir de duas partes verticais do novo conteúdo, vindas simultaneamente da esquerda e da direita |
| `SHOW_COMBINE` | `dword` | 15 | A tela se forma a partir de quatro partes do novo conteúdo, vindas simultaneamente dos cantos da tela |
| `SHOW_ZOOMOUT` | `dword` | 16 | A tela é reduzida em direção ao herói, de um grande aumento até o tamanho normal |
| `SHOW_MOSAIC` | `dword` | 17 | A tela é exibida através de um mosaico cada vez mais fino |
| `SHOW_ROSTERSCROLL` | `dword` | 18 | A tela está borrada de forma ondulada e vai ficando cada vez mais nítida |
| `SHOW_INSTANT` | `dword` | 19 | O novo conteúdo da tela fica visível imediatamente, sem transição |

## Veja também

`Command.SetScreenTransition()`, `Command.ShowScreen()`

CONSTANTES

# Grupo de constantes: Variantes de limpeza da tela

## Descrição

O grupo de constantes "Variantes de limpeza da tela" contém as constantes para indicar as variantes de apagar a tela. Essas constantes são usadas junto com o objeto `Command`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `ERASE_FADEOUT` | `dword` | 0 | A tela desaparece com fade-out |
| `ERASE_RANDOMBLOCKS` | `dword` | 1 | A tela se preenche aleatoriamente com blocos |
| `ERASE_RANDOMBLOCKSUP` | `dword` | 2 | A tela se preenche aleatoriamente com blocos, de baixo para cima |
| `ERASE_RANDOMBLOCKSDOWN` | `dword` | 3 | A tela se preenche aleatoriamente com blocos, de cima para baixo |
| `ERASE_BLINDCLOSE` | `dword` | 4 | A tela se preenche com listras horizontais |
| `ERASE_STRIPEUPDOWN` | `dword` | 5 | A tela é preenchida simultaneamente de cima e de baixo, cada uma com uma faixa preta |
| `ERASE_STRIPELEFTRIGHT` | `dword` | 6 | A tela é preenchida simultaneamente da esquerda e da direita, cada uma com uma faixa preta |
| `ERASE_OUTSIDEINSIDE` | `dword` | 7 | A tela se preenche em forma de retângulo, de fora para dentro |
| `ERASE_INSIDEOUTSIDE` | `dword` | 8 | A tela se preenche em forma de retângulo, de dentro para fora |
| `ERASE_SCROLLUP` | `dword` | 9 | A tela é deslocada para fora, para cima |
| `ERASE_SCROLLDOWN` | `dword` | 10 | A tela é deslocada para fora, para baixo |
| `ERASE_SCROLLLEFT` | `dword` | 11 | A tela é deslocada para fora, para a esquerda |
| `ERASE_SCROLLRIGHT` | `dword` | 12 | A tela é deslocada para fora, para a direita |
| `ERASE_DIVISIONUPDOWN` | `dword` | 13 | A tela se divide horizontalmente ao meio, e as metades de cima e de baixo se afastam simultaneamente |
| `ERASE_DIVISIONLEFTRIGHT` | `dword` | 14 | A tela se divide verticalmente ao meio, e as metades da esquerda e da direita se afastam simultaneamente |
| `ERASE_DIVISION` | `dword` | 15 | A tela se divide horizontal e verticalmente ao meio, e os quatro retângulos se afastam simultaneamente em direção ao seu canto da tela |
| `ERASE_ZOOMIN` | `dword` | 16 | A tela é fortemente ampliada em direção ao herói |
| `ERASE_MOSAIC` | `dword` | 17 | A tela desaparece através de um mosaico cada vez mais grosso |
| `ERASE_ROSTERSCROLL` | `dword` | 18 | A tela borra de forma ondulada |
| `ERASE_INSTANT` | `dword` | 19 | A tela desaparece imediatamente, sem transição |
| `ERASE_DONTERASE` | `dword` | 20 | A tela não é apagada. Essa constante é inválida em `Command.EraseScreen()`. |

## Veja também

`Command.SetScreenTransition()`, `Command.EraseScreen()`

CONSTANTES

# Grupo de constantes: Códigos virtuais de tecla

## Descrição

O grupo de constantes "Códigos virtuais de tecla" contém as constantes para identificação das teclas do teclado. Essas constantes são usadas junto com o objeto `Keyboard`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `VK_A` | `dword` | 65 | A tecla "A" |
| `VK_B` | `dword` | 66 | A tecla "B" |
| `VK_C` | `dword` | 67 | A tecla "C" |
| `VK_D` | `dword` | 68 | A tecla "D" |
| `VK_E` | `dword` | 69 | A tecla "E" |
| `VK_F` | `dword` | 70 | A tecla "F" |
| `VK_G` | `dword` | 71 | A tecla "G" |
| `VK_H` | `dword` | 72 | A tecla "H" |
| `VK_I` | `dword` | 73 | A tecla "I" |
| `VK_J` | `dword` | 74 | A tecla "J" |
| `VK_K` | `dword` | 75 | A tecla "K" |
| `VK_L` | `dword` | 76 | A tecla "L" |
| `VK_M` | `dword` | 77 | A tecla "M" |
| `VK_N` | `dword` | 78 | A tecla "N" |
| `VK_O` | `dword` | 79 | A tecla "O" |
| `VK_P` | `dword` | 80 | A tecla "P" |
| `VK_Q` | `dword` | 81 | A tecla "Q" |
| `VK_R` | `dword` | 82 | A tecla "R" |
| `VK_S` | `dword` | 83 | A tecla "S" |
| `VK_T` | `dword` | 84 | A tecla "T" |
| `VK_U` | `dword` | 85 | A tecla "U" |
| `VK_V` | `dword` | 86 | A tecla "V" |
| `VK_W` | `dword` | 87 | A tecla "W" |
| `VK_X` | `dword` | 88 | A tecla "X" |
| `VK_Y` | `dword` | 89 | A tecla "Y" |
| `VK_Z` | `dword` | 90 | A tecla "Z" |
| `VK_0` | `dword` | 48 | A tecla "0" |
| `VK_1` | `dword` | 49 | A tecla "1" |
| `VK_2` | `dword` | 50 | A tecla "2" |
| `VK_3` | `dword` | 51 | A tecla "3" |

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| VK_4 | dword | 52 | A tecla "4" |
| VK_5 | dword | 53 | A tecla "5" |
| VK_6 | dword | 54 | A tecla "6" |
| VK_7 | dword | 55 | A tecla "7" |
| VK_8 | dword | 56 | A tecla "8" |
| VK_9 | dword | 57 | A tecla "9" |
| VK_F1 | dword | 112 | A tecla de função 1 |
| VK_F2 | dword | 113 | A tecla de função 2 |
| VK_F3 | dword | 114 | A tecla de função 3 |
| VK_F4 | dword | 115 | A tecla de função 4 |
| VK_F5 | dword | 116 | A tecla de função 5 |
| VK_F6 | dword | 117 | A tecla de função 6 |
| VK_F7 | dword | 118 | A tecla de função 7 |
| VK_F8 | dword | 119 | A tecla de função 8 |
| VK_F9 | dword | 120 | A tecla de função 9 |
| VK_F10 | dword | 121 | A tecla de função 10 |
| VK_F11 | dword | 122 | A tecla de função 11 |
| VK_F12 | dword | 123 | A tecla de função 12 |
| VK_UP | dword | 38 | Seta para cima |
| VK_LEFT | dword | 37 | Seta para a esquerda |
| VK_DOWN | dword | 40 | Seta para baixo |
| VK_RIGHT | dword | 39 | Seta para a direita |
| VK_TAB | dword | 9 | Tab (tabulação horizontal) |
| VK_MENU | dword | 18 | A tecla "Alt" (Alternate) ou "Alt Gr" (Alternate Graphic) |
| VK_SHIFT | dword | 16 | A tecla Shift (esquerda ou direita) |
| VK_SPACE | dword | 32 | A barra de espaço |
| VK_CONTROL | dword | 17 | A tecla "Ctrl" (Control; esquerda ou direita) |
| VK_RETURN | dword | 13 | A tecla Enter |
| VK_ESCAPE | dword | 27 | Escape (Esc) |
| VK_PAUSE | dword | 19 | A tecla "Pause" |
| VK_PRINT | dword | 42 | A tecla "Print Screen" |
| VK_SCROLL | dword | 145 | A tecla "Scroll Lock" |
| VK_CAPITAL | dword | 20 | A tecla "Caps Lock" |
| VK_BACK | dword | 8 | A tecla Backspace |
| VK_HOME | dword | 36 | A tecla "Home" |
| VK_END | dword | 35 | A tecla "End" |
| VK_PGUP | dword | 33 | A tecla "Page Up" |
| VK_PGDN | dword | 34 | A tecla "Page Down" |

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `VK_INSERT` | `dword` | 45 | A tecla "Insert" |
| `VK_DELETE` | `dword` | 46 | A tecla "Delete" |
| `VK_LBUTTON` | `dword` | 1 | O botão esquerdo do mouse |
| `VK_MBUTTON` | `dword` | 4 | O botão do meio do mouse |
| `VK_RBUTTON` | `dword` | 2 | O botão direito do mouse |
| `VK_DECIMAL` | `dword` | 110 | O separador decimal do teclado numérico |
| `VK_NUMLOCK` | `dword` | 144 | A tecla "Num Lock" (alterna o teclado numérico) |
| `VK_NUMPAD0` | `dword` | 96 | A tecla "0" do teclado numérico |
| `VK_NUMPAD1` | `dword` | 97 | A tecla "1" do teclado numérico |
| `VK_NUMPAD2` | `dword` | 98 | A tecla "2" do teclado numérico |
| `VK_NUMPAD3` | `dword` | 99 | A tecla "3" do teclado numérico |
| `VK_NUMPAD4` | `dword` | 100 | A tecla "4" do teclado numérico |
| `VK_NUMPAD5` | `dword` | 101 | A tecla "5" do teclado numérico |
| `VK_NUMPAD6` | `dword` | 102 | A tecla "6" do teclado numérico |
| `VK_NUMPAD7` | `dword` | 103 | A tecla "7" do teclado numérico |
| `VK_NUMPAD8` | `dword` | 104 | A tecla "8" do teclado numérico |
| `VK_NUMPAD9` | `dword` | 105 | A tecla "9" do teclado numérico |
| `VK_ADD` | `dword` | 107 | A tecla "+" (mais) |
| `VK_SUBTRACT` | `dword` | 109 | A tecla "-" (menos) |
| `VK_MULTIPLY` | `dword` | 106 | A tecla "*" (vezes) |
| `VK_DIVIDE` | `dword` | 111 | A tecla "/" (dividido) |

## Veja também

`Keyboard.GetKeyState()`, `Keyboard.GetKey()`, `Keyboard.GetKeyText()`, `Keyboard.SetKeyState()`

CONSTANTES

# Grupo de constantes: Tipos de valor

## Descrição

O grupo de constantes Wertetyp contém as constantes para indicar tipos de valor. Essas constantes são usadas junto com o objeto `Command`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `VALUETYPE_NUMBER` | `dword` | 0 | Como valor é usado um número |
| `VALUETYPE_VARIABLE` | `dword` | 1 | Como valor é usado o valor de outra variável |
| `VALUETYPE_VARIABLEINDIRECT` | `dword` | 2 | Como valor é usado o valor de outra variável, que por sua vez é referenciada por outra variável |

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `VALUETYPE_RANDOM` | `dword` | 3 | Como valor é usado um número aleatório |
| `VALUETYPE_ITEM` | `dword` | 4 | Como valor é usada uma propriedade de item |
| `VALUETYPE_ACTOR` | `dword` | 5 | Como valor é usada uma propriedade de herói |
| `VALUETYPE_EVENT` | `dword` | 6 | Como valor é usada uma propriedade de evento |
| `VALUETYPE_MISC` | `dword` | 7 | Como valor é usada outra propriedade qualquer do jogo |

## Veja também

`Command.ChangeVariable()`, `Command.ChangeVariableRange()`

CONSTANTES

# Grupo de constantes: Tipos de clima

## Descrição

O grupo de constantes Wetterarten contém as constantes para indicar os diferentes tipos de clima. Essas constantes são usadas junto com o objeto `Command`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `WEATHER_NONE` | `dword` | 0 | Nenhum efeito de clima (céu limpo) |
| `WEATHER_RAIN` | `dword` | 1 | Está chovendo |
| `WEATHER_SNOW` | `dword` | 2 | Está nevando |

## Veja também

`Command.SetWeather()`

CONSTANTES

# Grupo de constantes: Intensidades de clima

## Descrição

O grupo de constantes Wetterstärken contém as constantes para indicar as intensidades dos efeitos de clima. Essas constantes são usadas junto com o objeto `Command`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `WEATHERSTRENGTH_WEAK` | `dword` | 0 | O efeito de clima é fraco (ex.: uma leve garoa) |
| `WEATHERSTRENGTH_NORMAL` | `dword` | 1 | O efeito de clima é normal (ex.: uma chuva normal) |
| `WEATHERSTRENGTH_STRONG` | `dword` | 2 | O efeito de clima é forte (ex.: um temporal) |

## Veja também

`Command.SetWeather()`

CONSTANTES

# Grupo de constantes: Formato de ângulo

## Descrição

O grupo de constantes Winkelformat contém as constantes para indicar os diferentes formatos de ângulo. Essas constantes são usadas junto com os objetos `Convert` e `Math`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `DEG` | `dword` | 1 | Os ângulos são informados em graus (1 círculo completo = 360) |
| `RAD` | `dword` | 2 | Os ângulos são informados em radianos (1 círculo completo = 2 × π = 6,283...) |
| `GRAD` | `dword` | 3 | Os ângulos são informados em grados (1 círculo completo = 400) |
| `RPG` | `dword` | 4 | Os ângulos são informados no formato do RPG Maker (1 círculo completo = 256) |

## Veja também

`Convert.Angle()`, `Math.Sin()`, `Math.Cos()`, `Math.Tan()`, `Math.Cot()`, `Math.Sec()`, `Math.Csc()`, `Math.ASin()`, `Math.ACos()`, `Math.ATan()`, `Math.ACot()`, `Math.ASec()`, `Math.ACsc()`

CONSTANTES

# Grupo de constantes: Dias da semana

## Descrição

O grupo de constantes Wochentage contém as constantes para indicar os dias da semana. Essas constantes são usadas junto com os objetos `Time` e `String`.

## Constantes

| Nome | Tipo | Valor | Descrição |
|---|---|---|---|
| `SUNDAY` | `dword` | 0 | Domingo |
| `MONDAY` | `dword` | 1 | Segunda-feira |
| `TUESDAY` | `dword` | 2 | Terça-feira |
| `WEDNESDAY` | `dword` | 3 | Quarta-feira |
| `THURSDAY` | `dword` | 4 | Quinta-feira |
| `FRIDAY` | `dword` | 5 | Sexta-feira |
| `SATURDAY` | `dword` | 6 | Sábado |

## Veja também

`Time.Weekday`, `String.WeekdayName()`

# Lista de erros

## Descrição

Durante a execução de um DestinyScript, podem ocorrer erros em vários pontos. A lista de todos os erros possíveis e suas descrições está reunida aqui. Para uma descrição mais detalhada, clique no nome do erro.

## Erros

| Nº | Nome | Descrição |
|---|---|---|
| 0 | `DESTINYERROR_UNKNOWN` | Ocorreu um erro desconhecido |
| 1 | `DESTINYERROR_SYNTAX` | Ocorreu um erro de sintaxe |
| 2 | `DESTINYERROR_NOVALUE` | Nenhum valor foi informado onde se esperava um valor |
| 3 | `DESTINYERROR_UNKNOWNNAME` | Foi informado um identificador desconhecido |
| 4 | `DESTINYERROR_CONVERT` | Ocorreu um erro de conversão |
| 5 | `DESTINYERROR_READONLY` | Tentou-se escrever em um valor somente leitura |
| 6 | `DESTINYERROR_ARRAYBOUND` | Os limites de um array foram excedidos |
| 7 | `DESTINYERROR_RANGE` | O valor está fora da faixa permitida para uma conversão |
| 8 | `DESTINYERROR_MEMORY` | Ocorreu um erro de memória |
| 9 | `DESTINYERROR_VALUE` | Foi atribuído ou usado como parâmetro um valor inválido |
| 10 | `DESTINYERROR_BINARYFLOAT` | Foi executada uma operação binária com valor de ponto flutuante |
| 11 | `DESTINYERROR_CALCSWITCH` | Foi executada uma operação aritmética/binária com um switch |
| 12 | `DESTINYERROR_CALCSTRING` | Foi executada uma operação matemática com uma string |
| 13 | `DESTINYERROR_FLOATERROR` | Ocorreu um erro de ponto flutuante |
| 14 | `DESTINYERROR_FLOATLENGTH` | Foi informado um número decimal longo demais |
| 15 | `DESTINYERROR_DIVISIONBYZERO` | Foi executada uma divisão por zero |
| 16 | `DESTINYERROR_STRINGFORMAT` | Foi informado um formato de string inválido |
| 17 | `DESTINYERROR_STRINGRANGE` | Foi informada uma posição de texto inválida |
| 18 | `DESTINYERROR_PICTURE` | Foi acessado um AuroraSheet não inicializado |
| 19 | `DESTINYERROR_PIXELRANGE` | Foi acessado um pixel fora da área da imagem |
| 20 | `DESTINYERROR_SAMEPICTURE` | Foi executada uma cópia com sobreposição entre a mesma imagem de origem e de destino |
| 21 | `DESTINYERROR_PALETTERANGE` | Tentou-se acessar uma entrada de paleta inválida |
| 22 | `DESTINYERROR_SOCKETSTARTUP` | A inicialização do sistema de sockets falhou |
| 23 | `DESTINYERROR_NOFREESOCKET` | Não há nenhum socket livre |
| 24 | `DESTINYERROR_CANTCREATESOCKET` | Não foi possível criar um novo socket |
| 25 | `DESTINYERROR_SOCKETSTILLOPEN` | Foi executada uma operação que exige um socket fechado |
| 26 | `DESTINYERROR_SOCKETNOTOPEN` | Foi executada uma operação que exige um socket aberto |
| 27 | `DESTINYERROR_CANTCONNECT` | Não foi possível estabelecer conexão com o socket |
| 28 | `DESTINYERROR_SOCKETTYPE` | Foi executada uma operação inválida para o tipo de socket |

| Nº | Nome | Descrição |
|---|---|---|
| 29 | `DESTINYERROR_SOCKETERROR` | Ocorreu um erro de socket |
| 30 | `DESTINYERROR_OOB` | Chegou lixo a um socket (Out-of-Band) |
| 31 | `DESTINYERROR_STRINGTOOLONG` | Uma string informada é longa demais para ser enviada por um socket Destiny |
| 32 | `DESTINYERROR_NOFREEFILEHANDLE` | Foi executada uma operação que exige um handle de arquivo fechado |
| 33 | `DESTINYERROR_CANTRESOLVEPATH` | Um caminho informado não pôde ser resolvido |
| 34 | `DESTINYERROR_NOPERMISSION` | Tentou-se executar uma operação para a qual não há permissão |
| 35 | `DESTINYERROR_CANTOPENFILE` | Não foi possível abrir um arquivo |
| 36 | `DESTINYERROR_FILENOTOPEN` | Foi executada uma operação que exige um handle de arquivo aberto |
| 37 | `DESTINYERROR_CANTACCESSFILE` | Não é possível acessar os dados de um arquivo |
| 38 | `DESTINYERROR_CANTCREATEDIR` | Não foi possível criar uma pasta |
| 39 | `DESTINYERROR_CANTREMOVEDIR` | Não foi possível excluir uma pasta |
| 40 | `DESTINYERROR_CANTRENAMEFILE` | Não foi possível renomear um arquivo/uma pasta |
| 41 | `DESTINYERROR_CANTCOPYFILE` | Não foi possível copiar um arquivo |
| 42 | `DESTINYERROR_CANTDELETEFILE` | Não foi possível excluir um arquivo |
| 43 | `DESTINYERROR_CANTREADATTRIBUTES` | Não foi possível ler os atributos de um arquivo/uma pasta |
| 44 | `DESTINYERROR_CANTWRITEATTRIBUTES` | Não foi possível gravar os atributos de um arquivo/uma pasta |
| 45 | `DESTINYERROR_SEARCHSTILLOPEN` | Uma busca em pasta já foi iniciada |
| 46 | `DESTINYERROR_CANTSTARTSEARCH` | Não foi possível iniciar uma busca em pasta |
| 47 | `DESTINYERROR_NOSEARCHSTARTED` | Nenhuma busca em pasta foi iniciada ainda |
| 48 | `DESTINYERROR_PROTECTED` | Tentou-se chamar uma função bloqueada do Destiny |
| 49 | `DESTINYERROR_CANTEXECUTEFILE` | A tentativa de executar um arquivo falhou |
| 50 | `DESTINYERROR_NOTENOUGHPARAMETERS` | Não foram informados parâmetros suficientes para executar um comando |
| 51 | `DESTINYERROR_NOTCOMPARABLE` | Foram comparados dois valores que não são comparáveis |
| 52 | `DESTINYERROR_INVALIDSIGN` | Foi informado um sinal (prefixo/sufixo) inválido |
| 53 | `DESTINYERROR_SETNOTALLOWEDHERE` | Um valor foi escrito em um lugar onde isso não é permitido |
| 54 | `DESTINYERROR_MISSINGENDIF` | Falta a indicação "EndIf" |
| 55 | `DESTINYERROR_MISSINGLOOP` | Falta a indicação "Loop" |
| 56 | `DESTINYERROR_MISSINGCONDITION` | Nenhuma condição foi informada |
| 57 | `DESTINYERROR_MISSINGNEXT` | Falta a indicação "Next" |
| 58 | `DESTINYERROR_MISSINGCASE` | Falta a indicação "Case" |
| 59 | `DESTINYERROR_MISSINGENDSWITCH` | Falta a indicação "EndSwitch" |
| 60 | `DESTINYERROR_BREAKNOTALLOWEDHERE` | Foi executado um Break em um lugar onde isso não é permitido |
| 61 | `DESTINYERROR_CONTINUENOTALLOWEDHERE` | Foi executado um Continue em um lugar onde isso não é permitido |
| 62 | `DESTINYERROR_INVALIDHANDLE` | Foi usado um handle inválido |
| 63 | `DESTINYERROR_OPERATOREXPECTED` | Esperava-se um operador em um ponto, mas nenhum foi informado |
| 64 | `DESTINYERROR_CANTACCESSSAVESLOT` | Não é possível acessar um slot de save |
| 65 | `DESTINYERROR_EXCEPTION` | Ocorreu uma exceção |

| Nº | Nome | Descrição |
|---|---|---|
| 66 | `DESTINYERROR_INVALIDLOOPOPERATIONLEVEL` | Foi informado um nível inválido de operador de laço |
| 67 | `DESTINYERROR_CONTINUEDOESNTPOINTTOLOOP` | Foi executado um Continue que não se refere a um laço |

ERRO

# Erro desconhecido

## Descrição

Ocorreu um erro desconhecido. Na verdade, este erro nunca deveria ocorrer; ele foi incluído apenas para o caso de um erro de programação. Se algum dia ele aparecer, isso indica ou uma memória defeituosa ou um erro de programa no DestinyPatch.

## Veja também

`DESTINYERROR_UNKNOWN`

ERRO

# Erro 1: Sintaxe

## Descrição

Este erro ocorre sempre que um caractere inesperado aparece no DestinyScript. Em geral acontece quando houve erro de digitação.

## Exemplo

```
$
v[1) = 7;  // Em vez de `]` aparece `)`
```

## Veja também

`DESTINYERROR_SYNTAX`

ERRO

# Erro 2: Nenhum valor informado

## Descrição

Este erro ocorre sempre que o interpretador espera um valor, mas não recebe nenhum. Isso acontece, por exemplo, quando se escreve um parêntese vazio em um cálculo, quando um parâmetro não opcional não é informado (embora, por causa de uma vírgula, ele devesse vir) ou quando um método não devolve valor de retorno e mesmo assim se trabalha com ele.

## Exemplo

Parênteses vazios em um cálculo:

```
$
v[1] = 3 + ()
```

Parâmetro não opcional ausente:

```
$
v[1] = Logic.And3(true, true, );
```

Acesso ao valor de retorno inexistente de um método:

```
$
v[1] = Keyboard.SetKeyState(VK_LEFT, KEYEVENTF_KE
```

## Veja também

`DESTINYERROR_NOVALUE`

ERRO

# Erro 3: Identificador desconhecido

## Descrição

Este erro ocorre sempre que um identificador desconhecido (nome de uma variável, constante, objeto, método, propriedade…) é digitado incorretamente. Na maioria das vezes acontece por erro de digitação.

## Exemplo

```
$
v[1] = Picture[1].Toastbrot;    // O objeto Picture…
```

## Veja também

`DESTINYERROR_UNKNOWNNAME`

ERRO

# Erro 4: Erro de conversão

## Descrição

Ocorreu um erro na conversão de tipos de dados. Na verdade, este erro nunca deveria ocorrer; ele foi incluído apenas para o caso de um erro de programação. Se algum dia ele aparecer, isso indica ou uma memória defeituosa ou um erro de programa no DestinyPatch.

## Veja também

`DESTINYERROR_CONVERT`

ERRO

# Erro 5: Escrita em valor somente leitura

## Descrição

Este erro ocorre sempre que se tenta escrever em um valor somente leitura. Se um valor é somente leitura ou não, isso sempre consta na documentação do valor (por exemplo, nas propriedades de um objeto).

## Exemplo

```
$
Destiny.DllVersionMajor = 200;  // A propriedade…
```

## Veja também

`DESTINYERROR_READONLY`

ERRO

# Erro 6: Limites do array excedidos

## Descrição

Este erro ocorre sempre que um valor inválido (= um índice inválido) é informado entre colchetes. A faixa de valores permitida para um objeto/uma propriedade consta na documentação do objeto/da propriedade.

> **Informação:** Em geral, objetos com colchetes possuem a propriedade `Count`, com a qual se descobre o limite superior do índice. O limite inferior costuma ser 0 (objetos da `Destiny.dll`) ou 1 (objetos do `RPG_RT`).

## Exemplo

```
$
Picture[-1].UseMaskColor = false;   // O índice…
```

## Veja também

`DESTINYERROR_ARRAYBOUND`

ERRO

# Erro 7: Valor fora da faixa permitida

## Descrição

Este erro ocorre sempre que um tipo de dado é convertido em outro, mas o valor é grande demais para o tipo de destino.

## Exemplo

```
$
v[1] = Math.Exp(11);  // O maior valor possível…
                      // 100.000.000.000 (= Math.…
```

## Veja também

`DESTINYERROR_RANGE`

ERRO

# Erro 8: Erro no gerenciamento de memória

## Descrição

Este erro ocorre sempre que não há memória suficiente para executar um comando/uma operação. Na maioria das vezes isso acontece quando uma string fica grande demais.

## Exemplo

```
$
a[1] = "Blutwurst";
for(;;)  // Laço infinito
  a[1] ..= String.Fill(a[1], 1000);
next;
```

## Veja também

`DESTINYERROR_MEMORY`

ERRO

# Erro 9: Valor inválido

## Descrição

Este erro ocorre sempre que se usa um valor que não é permitido para a sua finalidade (por exemplo, o parâmetro de um método ou o valor de uma propriedade). A faixa de valores dos parâmetros de um método ou das propriedades consta na documentação do respectivo método/da respectiva propriedade.

> **Informação:** Na maioria dos parâmetros/propriedades em que há faixas de valores prescritas, existem constantes disponíveis. Usando constantes, é possível garantir que apenas valores válidos sejam usados.

## Exemplo

```
$
Server.Listen(1000000);    // A porta precisa estar no intervalo…
```

## Veja também

`DESTINYERROR_VALUE`

ERRO

# Erro 10: Operação binária com valor de ponto flutuante

## Descrição

Este erro ocorre sempre que uma operação binária (E binário ( & ), OU binário ( | ), NÃO binário ( ~ ) ou OU exclusivo binário ( ^ )) é executada com um valor de ponto flutuante. Operações binárias só são permitidas com números inteiros (Byte, Word ou Dword).

> **Informação:** Para executar uma operação binária com um número de ponto flutuante, um switch ou uma string, pode-se convertê-lo antes explicitamente para um inteiro (por exemplo, com `Convert.Dword()`).

## Exemplo

```
$
v[1] = 1.5 & 1;  // Operações binárias só são…
```

## Veja também

`DESTINYERROR_BINARYFLOAT`

ERRO

# Erro 11: Operação matemática com Switch

## Descrição

Este erro ocorre sempre que uma operação aritmética/binária (mais ( + ), menos ( - ), vezes ( * ), dividido ( / ), módulo ( % ), E binário ( & ), OU binário ( | ), NÃO binário ( ~ ) ou OU exclusivo binário ( ^ )) é executada com um switch. Com switches só são permitidas operações lógicas (E lógico ( && ), OU lógico ( || ) e NÃO lógico ( ! )).

> **Informação:** Para executar uma operação aritmética/binária com um switch, pode-se convertê-lo antes explicitamente para outro tipo de dado (por exemplo, Dword ou Float), por exemplo com `Convert.Dword()`.

## Exemplo

```
$
s[1] = true + false;  // Operações aritméticas/binárias…
```

## Veja também

`DESTINYERROR_CALCSWITCH`

ERRO

# Erro 12: Operação matemática com String

## Descrição

Este erro ocorre sempre que uma operação aritmética/binária (menos ( - ), vezes ( * ), dividido ( / ), módulo ( % ), E binário ( & ), OU binário ( | ), NÃO binário ( ~ ) ou OU exclusivo binário ( ^ )) é executada com uma string. Com strings só são permitidas as operações de concatenação (mais ( + ) — com duas strings isso é uma concatenação — e concatenação ( .. )), bem como as operações lógicas (E lógico ( && ), OU lógico ( || ) e NÃO lógico ( ! )).

> **Informação:** Para executar uma operação aritmética/binária com uma string, pode-se convertê-la antes explicitamente para outro tipo de dado (por exemplo, Dword ou Float), por exemplo com `Convert.Dword()`.

## Exemplo

```
$
a[1] = "3" * 5;  // Multiplicações com uma string…
```

## Veja também

`DESTINYERROR_CALCSTRING`

ERRO

# Erro 13: Erro de ponto flutuante

## Descrição

Este erro ocorre sempre que ocorre um erro não especificado com mais detalhes em um cálculo com número de ponto flutuante. Em geral isso só acontece quando houve um erro de dados (número de ponto flutuante inválido), quando o resultado de um cálculo de ponto flutuante não é um número real (por exemplo, um número complexo) ou quando o resultado de um cálculo é indefinido. Um erro de dados pode ocorrer quando o número de ponto flutuante foi lido de um arquivo que continha apenas lixo.

## Exemplo

```
$
f[1] = Math.Power(-2, 2.5);  // Tentativa de calcular um número complexo…
```

## Veja também

`DESTINYERROR_FLOATERROR`

ERRO

# Erro 14: Número decimal longo demais

## Descrição

Este erro ocorre sempre que um número (em forma de texto) com mais de 13 dígitos é convertido em número de ponto flutuante. Zeros à esquerda e à direita não são contados.

## Exemplo

```
$
f[1] = 1234567890.0987654321;    // O número tem…
```

## Veja também

`DESTINYERROR_FLOATLENGTH`

ERRO

# Erro 15: Divisão por zero

## Descrição

Este erro ocorre sempre que um número é dividido por 0 (isso vale também para o módulo). Segundo algumas regras básicas da matemática, o resultado da divisão por 0 é "indefinido" e, por isso, leva a erros.

## Exemplo

```
$
v[1] = 5 / 0;  // Divisão por 0 não é possível…
```

## Veja também

`DESTINYERROR_DIVISIONBYZERO`

ERRO

# Erro 16: Formato de string inválido

## Descrição

Este erro ocorre sempre que uma string é convertida em número, mas a string não contém um número formatado corretamente.

## Exemplo

```
$
v[1] = "Zahl: 12345";  // A string contém, além…
```

## Veja também

`DESTINYERROR_STRINGFORMAT`

ERRO

# Erro 17: Posição de texto inválida

## Descrição

Este erro ocorre sempre que se acessa uma posição inválida dentro de uma string.

> **Informação:** O primeiro caractere de uma string tem sempre a posição 0. O último caractere da string é ( `String.Length()` - 1).

## Exemplo

```
$
v[1] = String.Ord("Hallo", 5);  // Acessa-se o…
```

## Veja também

`DESTINYERROR_STRINGRANGE`

ERRO

# Erro 18: Imagem não inicializada

## Descrição

Este erro ocorre sempre que se acessa uma propriedade/um método de um AuroraSheet que exige um AuroraSheet carregado (por exemplo, `Picture[].AuroraSheet.Pixel[]`), mas ele ainda não está carregado.

> **Informação:** Se um AuroraSheet está carregado pode ser verificado com o método `IsEmpty` do objeto `AuroraSheet`.

## Exemplo

```
$
Picture[1].AuroraSheet.Free();                // A…
v[1] = Picture[1].AuroraSheet.Pixel[1, 1];    // A…
```

## Veja também

`DESTINYERROR_PICTURE`

ERRO

# Erro 19: Pixel fora da área da imagem

## Descrição

Este erro ocorre sempre que, por meio de uma propriedade/um método, se acessa uma área de imagem fora de um AuroraSheet.

> **Informação:** A largura de um AuroraSheet pode ser obtida pela propriedade `Width`, e a altura pela propriedade `Height`.

As coordenadas X e Y se referem sempre a valores na faixa de 0 … (largura - 1) e 0 … (altura - 1). As coordenadas Esquerda e Cima são equivalentes a X e Y. As coordenadas Direita e Baixo são, respectivamente, Esquerda + largura desejada e Cima + altura desejada. Assim, elas podem estar na faixa de 0 … largura e 0 … altura.

## Exemplo

```
$
Picture[1].AuroraSheet.Create(20, 20);
Picture[1].AuroraSheet.FillRect(0, 0, 100, 100, 0
```

## Veja também

`DESTINYERROR_PIXELRANGE`

ERRO

# Erro 20: Imagem de origem e destino iguais com sobreposição

## Descrição

Este erro ocorre sempre que, por meio de um método de cópia, se tenta copiar uma área de um AuroraSheet para o mesmo AuroraSheet e as áreas se sobrepõem. Como as áreas se sobrescreveriam em um comportamento indefinido, é disparado um erro.

> **Informação:** As coordenadas de destino (X e Y) precisam ser maiores ou iguais a Direita/Baixo ou menores que Esquerda - (Direita - Esquerda) e Cima - (Baixo - Cima); isso vale para X e Y separadamente.

Alteração em relação ao DestinyPatch V1.0: agora é permitido informar a mesma imagem de origem e de destino, desde que as áreas da imagem não se sobreponham.

## Exemplo

```
$
// O AuroraSheet da primeira Picture, no tamanho…
Picture[1].AuroraSheet.Create(100, 100);
// Copia a área da imagem de (50, 50) a (60, 60) para (…
Picture[1].AuroraSheet.CopyRect(45, 45, Picture[1
```

## Veja também

`DESTINYERROR_SAMEPICTURE`

ERRO

# Erro 21: Entrada fora da paleta de 256 cores

## Descrição

Este erro ocorre sempre que se faz referência a uma entrada de paleta fora da paleta de 256 cores.

> **Informação:** As entradas da paleta ficam sempre na faixa de 0 a 255.

## Exemplo

```
$
// Acessa-se a 257ª entrada (índice = 256) de uma paleta…
v[1] = Picture[1].AuroraSheet.Palette[256];
```

## Veja também

`DESTINYERROR_PALETTERANGE`

ERRO

# Erro 22: Falha na inicialização do socket

## Descrição

Ocorreu um erro na inicialização do sistema de sockets do Windows. Trata-se de um erro do sistema de destino, que pode ser causado por uma versão errada do Winsock instalada (o DestinyPatch V2.0 usa o Winsock versão 2.2) ou pela falta de dispositivos correspondentes (placa de rede, modem…).

## Veja também

`DESTINYERROR_SOCKETSTARTUP`

ERRO

# Erro 23: Nenhum socket livre disponível

## Descrição

Este erro ocorre sempre que ao método `Server.Accept()` é passada a constante `NEXT_FREE_SOCKET` e não há mais nenhum socket de cliente livre disponível. Este erro não pode ocorrer se, em vez de `NEXT_FREE_SOCKET`, for usado o valor de retorno de `Client.NextFree()`.

## Exemplo

```
$
// Excluir todos os sockets
Client.ReAlloc(0);
// Aceitar uma conexão na porta 12345
Server.Listen(12345);
// Aguardar uma conexão de entrada
do while((v[1] = Server.Accept(NEXT_FREE_SOCKET))
  pause;  // Aliviar o RPG_RT
loop;

// Fechar o servidor
Server.Close();
```

## Veja também

`DESTINYERROR_NOFREESOCKET`

ERRO

# Erro 24: Não foi possível criar o socket

## Descrição

Este erro ocorre sempre que não foi possível criar mais um socket. Isso não tem a ver com a quantidade de sockets no objeto `Client[]`, mas com os sockets do Windows: se já existem sockets demais (ou se algum processo bloqueia a criação de novos sockets), não é possível estabelecer conexões com outros computadores.

## Exemplo

O código a seguir pode (dependendo do sistema operacional(!)) provocar este erro. Também podem ocorrer outros erros (por exemplo, `DESTINYERROR_MEMORY`). Nos testes no Windows XP SP3 ocorreu sempre o erro `DESTINYERROR_CANTCONNECT`.

```
$
// Criar muitíssimos sockets
for(v[1] = 0; v[1] < 2147483647; v[1]++)
  // Abrir espaço para um novo socket
  Client.ReAlloc(v[1] + 1);
  // Uma conexão (não bloqueante) com…
  // (aqui o socket é finalmente criado!…
  Client[v[1]].Connect("127.0.0.1", v[1] % 65535
next;
```

## Veja também

`DESTINYERROR_CANTCREATESOCKET`

ERRO

# Erro 25: Socket não fechado

## Descrição

Este erro ocorre sempre que uma operação que exige um socket fechado é executada com um socket aberto. Isso pode acontecer, por exemplo, quando se chama o método `Connect` de um socket `Client` que já está conectado.

> **Informação:** Para verificar se um socket está conectado no momento, pode-se consultar a propriedade `State` do socket. Para estabelecer uma conexão com um socket já conectado, ele precisa primeiro ser fechado com o método `Close`.

> **Informação:** Com `Client.NextFree()` é possível descobrir o índice do próximo socket livre. Se não existir nenhum socket livre, um é criado automaticamente.

## Exemplo

```
$
// Estabelecer uma conexão com o Google
Client[0].Connect("www.google.de", 80, SOCK_RAW);
// Estabelecer mais uma conexão com o Google (com…
Client[0].Connect("www.google.de", 80, SOCK_RAW);
```

## Veja também

`DESTINYERROR_SOCKETSTILLOPEN`

ERRO

# Erro 26: Socket não aberto

## Descrição

Este erro ocorre sempre que uma operação que exige um socket aberto é executada com um socket fechado. Isso pode acontecer, por exemplo, quando se chama o método `SendVariable` de um socket `Destiny-Client` fechado.

> **Informação:** Para verificar se um socket está conectado no momento, pode-se consultar a propriedade `State` do socket. Se um socket estiver no estado `STATE_CLOSING`, ainda é possível obter dados pelos métodos de recebimento (sem que ocorra erro).

## Exemplo

```
$
// Fechar o primeiro socket
Client[0].Close();
// Enviar dados pelo primeiro socket
```

```
if(Client[0].Type == SOCK_DESTINY)
  Client[0].Destiny.SendVariable(0, 0);
else
  Client[0].Raw.Send(0);
endif;
```

### Veja também

`DESTINYERROR_SOCKETNOTOPEN`

ERRO

## Erro 27: Falha na conexão

### Descrição

Este erro ocorre sempre que uma tentativa de conexão com `Client[].Connect()` falha. Se ao método `Connect` foi passado o flag `SOCKETFLAG_NOTBLOCKING`, este erro só ocorre quando, no momento do pedido de conexão, já está claro que a conexão vai falhar (por exemplo, quando foi informado um endereço de destino inválido).

> **Informação:** Se o flag `SOCKETFLAG_NOTBLOCKING` foi informado em `Client[].Connect()`, então `Client[].State` devolve o valor `STATE_ERROR` quando a tentativa de conexão falhou.

### Exemplo

```
$
// Estabelecer uma conexão com um endereço inválido…
Client[0].Connect("adresse.die.ganz.sicher.nicht.
```

### Veja também

`DESTINYERROR_CANTCONNECT`

ERRO

## Erro 28: Operação inválida para este tipo de socket

### Descrição

Este erro ocorre sempre que se chama um método de socket destinado a outro tipo de socket. O tipo do socket é informado em `Client[].Connect()` ou em `Server.Listen()` e pode ser consultado com `Client[].Type`.

> **Informação:** Desde o DestinyPatch V2.0 foram introduzidos os subobjetos `Client[].Destiny` e `Client[].Raw`, para separar de forma mais limpa as propriedades/métodos usados de maneiras diferentes.

### Exemplo

```
$
// Estabelecer uma conexão com o Google por um socket cliente Raw…
Client[0].Connect("www.google.de", 80, SOCK_RAW);

// Enviar dados ao Google pelos métodos que…
```

```
// são previstos para sockets cliente Destiny.
Client[0].Destiny.SendVariable(0, 0);
```

## Veja também

`DESTINYERROR_SOCKETTYPE`

ERRO

# Erro 29: Erro em operação de socket

## Descrição

Ocorreu um erro em uma operação de socket. Este erro não é especificado com mais detalhes e pode ser causado por falta de recursos ou por uma conexão de rede interrompida.

## Veja também

`DESTINYERROR_SOCKETERROR`

ERRO

# Erro 30: Out-of-Band

## Descrição

Este erro ocorre sempre que chegou lixo (dados inválidos) a um socket. Isso pode acontecer, por exemplo, quando se conecta por um socket Destiny-Client a um servidor que não suporta o `DestinyProtokoll`, ou quando a conexão com o computador de destino é especialmente ruim. A única coisa que ainda se pode fazer é fechar o socket cliente com `Client[].Close()`.

> **Informação:** O `DestinyProtokoll` não possui correção de erros própria; por isso, ao chegarem dados inválidos, a conexão precisa ser fechada e reestabelecida. Normalmente, porém, só devem chegar dados válidos, pois o protocolo TCP/IP utilizado possui medidas próprias de correção de erros.

## Veja também

`DESTINYERROR_OOB`

ERRO

# Erro 31: String longa demais

## Descrição

Este erro ocorre sempre que, em um socket Destiny-Client, se envia uma string com mais de 255 caracteres.

> **Informação:** Para enviar strings com mais de 255 caracteres, pode-se proceder da seguinte forma:

1. A string é primeiro dividida em pedaços de 255 caracteres (`String.SubStr()`).
2. Depois, o primeiro pedaço da string é enviado com o ID da string.

3. Em seguida, todas as outras partes da string são enviadas com o ID 0.
4. O receptor anexa cada pedaço recebido com o ID 0 à última string recebida.

## Exemplo

```
$
// Estabelecer uma conexão com o próprio computador (…
Client[0].Connect("127.0.0.1", 12345, SOCK_DESTIN
Client[0].Destiny.SendString(1, String.Fill("a",
```

## Veja também

`DESTINYERROR_STRINGTOOLONG`

ERRO

# Erro 32: O handle de arquivo não está livre

## Descrição

Este erro ocorre sempre que uma operação que exige um handle de arquivo fechado é executada com um handle de arquivo aberto. Normalmente isso acontece em `File[].Open()`, quando se tenta abrir outro arquivo com um handle já aberto.

> **Informação:** Com `File.NextFree()` é possível descobrir o índice de um handle de arquivo livre. Se não existir nenhum handle de arquivo livre, um é criado automaticamente.

## Exemplo

```
$
// Abrir um arquivo
File[0].Open("Test.txt", FILE_WRITE);
// Abrir outro arquivo (com o mesmo handle…
File[0].Open("Test2.txt", FILE_WRITE);
```

## Veja também

`DESTINYERROR_NOFREEFILEHANDLE`

ERRO

# Erro 33: Não foi possível resolver o caminho

## Descrição

Ocorreu um erro ao resolver um caminho (do sistema de arquivos). Pode ser um erro do sistema de destino, mas, em teoria, também pode ser causado por um nome de arquivo inválido.

## Veja também

`DESTINYERROR_CANTRESOLVEPATH`

ERRO

# Erro 34: Sem permissão de acesso

## Descrição

Este erro ocorre sempre que uma operação (do sistema de arquivos) é executada sem que haja permissão para isso. Em geral, ele ocorre quando se tenta acessar um caminho fora da pasta do jogo.

## Exemplo

```
$
// Acessar um arquivo fora da pasta do jogo…
File[0].Open("..\Test.txt", FILE_WRITE);
```

## Veja também

`DESTINYERROR_NOPERMISSION`

ERRO

# Erro 35: Não foi possível abrir o arquivo

## Descrição

Este erro ocorre sempre que se abre um arquivo e a abertura falha. Isso pode ser porque um arquivo inexistente foi aberto com direitos de leitura, porque não há espaço livre suficiente ou porque o nome do arquivo contém caracteres inválidos.

## Exemplo

```
$
// Abrir um arquivo com nome inválido
File[0].Open("?", FILE_READ);
```

## Veja também

`DESTINYERROR_CANTOPENFILE`

ERRO

# Erro 36: Arquivo não aberto

## Descrição

Este erro ocorre sempre que uma operação que exige um handle de arquivo aberto é executada com um handle de arquivo fechado.

## Exemplo

```
$
// Fechar o primeiro handle de arquivo
File[0].Close();
// Tentar enviar dados pelo handle de arquivo fechado…
File[0].WriteRawData(12345);
```

## Veja também

`DESTINYERROR_FILENOTOPEN`

ERRO

# Erro 37: Acesso a arquivo impossível

## Descrição

Ocorreu um erro ao acessar (ler/gravar) os dados de um arquivo. Pode ser um erro do sistema de destino (por exemplo, quando não há mais espaço disponível).

## Veja também

`DESTINYERROR_CANTACCESSFILE`

ERRO

# Erro 38: Não foi possível criar a pasta

## Descrição

Ocorreu um erro ao criar uma pasta. Na maioria das vezes é um erro do sistema de destino (por exemplo, quando não há mais espaço disponível ou já existe uma pasta ou um arquivo com esse nome). O erro também pode estar no nome da pasta, se ele contiver caracteres inválidos.

## Veja também

`DESTINYERROR_CANTCREATEDIR`

ERRO

# Erro 39: Não foi possível excluir a pasta

## Descrição

Ocorreu um erro ao excluir uma pasta. Na maioria das vezes é um erro do sistema de destino (por exemplo, quando a pasta não está vazia ou não existe pasta com esse nome). O erro também pode estar no nome da pasta, se ele contiver caracteres inválidos.

## Veja também

`DESTINYERROR_CANTREMOVEDIR`

ERRO

# Erro 40: Não foi possível renomear o arquivo/pasta

## Descrição

Ocorreu um erro ao renomear um arquivo/uma pasta. Na maioria das vezes é um erro do sistema de destino (por exemplo, quando não há mais espaço disponível ou já existe um arquivo/uma pasta com esse nome). O erro também pode estar no nome do arquivo/da pasta, se ele contiver caracteres inválidos.

## Veja também

`DESTINYERROR_CANTRENAMEFILE`

ERRO

# Erro 41: Não foi possível copiar o arquivo

## Descrição

Ocorreu um erro ao copiar um arquivo. Na maioria das vezes é um erro do sistema de destino (por exemplo, quando não há mais espaço disponível ou já existe um arquivo/uma pasta com esse nome). O erro também pode estar no nome do arquivo, se ele contiver caracteres inválidos.

## Veja também

`DESTINYERROR_CANTCOPYFILE`

ERRO

# Erro 42: Não foi possível excluir o arquivo

## Descrição

Ocorreu um erro ao excluir um arquivo. Na maioria das vezes é um erro do sistema de destino (por exemplo, quando o arquivo é somente leitura).

## Veja também

`DESTINYERROR_CANTDELETEFILE`

ERRO

# Erro 43: Não foi possível ler os atributos

## Descrição

Ocorreu um erro ao consultar os atributos de um arquivo/uma pasta. Na maioria das vezes é um erro do sistema de destino (por exemplo, quando não se tem os direitos necessários para acessar o arquivo/a pasta).

## Veja também

`DESTINYERROR_CANTREADATTRIBUTES`

ERRO

# Erro 44: Não foi possível gravar os atributos

## Descrição

Ocorreu um erro ao definir os atributos de um arquivo/uma pasta. Na maioria das vezes é um erro do sistema de destino (por exemplo, quando não se tem os direitos necessários para acessar o arquivo/a pasta).

## Veja também

`DESTINYERROR_CANTWRITEATTRIBUTES`

ERRO

# Erro 45: Busca já aberta

## Descrição

Este erro não pode mais ocorrer no DestinyPatch V2.0, pois as buscas em pastas são processadas por uma pilha LIFO (Last-In, First-Out).

> **Informação:** No DestinyPatch V1.0, só era possível ter uma busca em pasta iniciada por vez.

## Propriedades

- **Obsoleto:** sim

## Veja também

`DESTINYERROR_SEARCHSTILLOPEN`

ERRO

# Erro 46: Não foi possível iniciar a busca

## Descrição

Este erro ocorre sempre que uma busca em pasta não pôde ser iniciada. Isso pode ser causado pelo sistema de destino, mas também pela máscara de busca (por exemplo, quando se quer pesquisar uma pasta inválida).

## Exemplo

```
$
Directory.FindFirst("|\*.*");   // | não é um…
```

## Veja também

`DESTINYERROR_CANTSTARTSEARCH`

ERRO

# Erro 47: Busca não iniciada

## Descrição

Este erro ocorre sempre que se tenta obter a próxima entrada de uma busca em pasta sem que nenhuma busca em pasta tenha sido iniciada.

## Exemplo

```
$
// Fechar todas as buscas abertas
Directory.FindCloseAll();
// Obter a próxima entrada de uma busca
a[1] Directory.FindNext();
```

## Veja também

`DESTINYERROR_NOSEARCHSTARTED`

ERRO

# Erro 48: Função do Destiny bloqueada

## Descrição

Atualmente este erro não pode ocorrer, pois as funções necessárias para isso ainda não foram implementadas.

## Veja também

`DESTINYERROR_PROTECTED`

ERRO

# Erro 49: Execução de arquivo impossível

## Descrição

Atualmente este erro não pode ocorrer, pois as funções necessárias para isso ainda não foram implementadas.

## Veja também

`DESTINYERROR_PROTECTED`

ERRO

# Erro 50: Parâmetros insuficientes

## Descrição

Este erro ocorre sempre que, na chamada de uma propriedade/de um método, foram informados parâmetros de menos.

> **Informação:** Se e quais parâmetros de um comando são opcionais (ou seja, podem ser omitidos) consta na documentação do respectivo comando.

## Exemplo

```
$
// Chamar o método para definir o estado de uma tecla…
Keyboard.SetKeyState();  // Os parâmetros foram…
```

## Veja também

`DESTINYERROR_NOTENOUGHPARAMETERS`

ERRO

# Erro 51: Não comparável

## Descrição

Este erro ocorre sempre que se executam comparações com um operador de comparação em que os tipos de dados da comparação são tão diferentes que uma conversão não pode ser feita de forma inequívoca. O erro ocorre nos seguintes casos:

| Tipo de dado 1 | Tipo de dado 2 |
|---|---|
| String | Todos os tipos de dados que não sejam String |
| Bool | Todos os tipos de dados que não sejam Bool |
| Float | String e Bool; com todos os demais tipos de dados tenta-se converter para Float |

| Tipo de dado 1 | Tipo de dado 2 |
|---|---|
| Dword | String e Bool; com todos os demais tipos de dados tenta-se converter para Dword. Se o tipo de dado do segundo operando for Float, é feita uma comparação de Float. |

## Exemplo

Comparar uma string com um número:

```
$
// Comparar uma string com um número
if("3" == 3)
  // ...
endif;
```

## Veja também

`DESTINYERROR_NOTCOMPARABLE`

ERRO

# Erro 52: Sinal (prefixo) inválido

## Descrição

Este erro ocorre sempre que foi informado um sinal (prefixo ou sufixo) inválido. Além dos sinais normais (+ - ~ !), também podem ser usados um incremento/decremento (++ ou --). O incremento/decremento pode ficar antes ou depois da variável a ser alterada (mas não dos dois lados ao mesmo tempo).

## Exemplo

```
$
++v[1]++;  // Incrementar e decrementar a variável 1 ao mesmo tempo…
```

## Veja também

`DESTINYERROR_INVALIDSIGN`

ERRO

# Erro 53: O valor não pode ser gravado aqui

## Descrição

Este erro ocorre sempre que um valor é escrito em um lugar (por exemplo, no meio de um cálculo) onde isso não é permitido (ou seja, é usado um operador de atribuição).

## Exemplo

```
$
// Atribuir um valor ao resultado do cálculo de uma fórmula…
v[1] + 3 = 5;
```

## Veja também

`DESTINYERROR_SETNOTALLOWEDHERE`

ERRO

# Erro 54: EndIf ausente

## Descrição

Este erro ocorre sempre que falta o `EndIf` correspondente a um `If`. Isso pode acontecer, por exemplo, ao copiar código DestinyScript manualmente.

## Exemplo

```
$
if(true)
loop;
```

## Veja também

`DESTINYERROR_MISSINGENDIF`

ERRO

# Erro 55: Loop ausente

## Descrição

Este erro ocorre sempre que falta o `Loop` correspondente a um `Do`. Isso pode acontecer, por exemplo, ao copiar código DestinyScript manualmente.

## Exemplo

```
$
do
next;
```

## Veja também

`DESTINYERROR_MISSINGLOOP`

ERRO

# Erro 56: Condição ausente

## Descrição

Este erro ocorre sempre que, em um laço Do-Loop, nenhuma condição foi informada.

> **Informação:** Para criar laços infinitos de propósito, pode-se usar a seguinte sintaxe: `for(;;) next;`

## Exemplo

```
$
do
loop;
```

## Veja também

`DESTINYERROR_MISSINGCONDITION`

ERRO

# Erro 57: Next ausente

## Descrição

Este erro ocorre sempre que falta o `Next` correspondente a um `For`. Isso pode acontecer, por exemplo, ao copiar código DestinyScript manualmente.

## Exemplo

```
$
for(v[1] = 0; v[1] < 10; v[1]++)
loop;
```

## Veja também

`DESTINYERROR_MISSINGNEXT`

ERRO

# Erro 58: Case ausente

## Descrição

Este erro ocorre sempre que, dentro de um `Switch`, nenhum `Case` ou `Default` foi informado.

## Exemplo

```
$
switch(v[1])
endswitch;
```

## Veja também

`DESTINYERROR_MISSINGCASE`

ERRO

# Erro 59: EndSwitch ausente

## Descrição

Este erro ocorre sempre que falta o `EndSwitch` correspondente a um `Switch`. Isso pode acontecer, por exemplo, ao copiar código DestinyScript manualmente.

> **Informação:** Depois do bloco Default, não é mais possível informar outros blocos Case. Se isso for feito mesmo assim, este erro também ocorre.

## Exemplo

```
$
switch(v[1])
  default:
endif;
```

`DESTINYERROR_MISSINGENDSWITCH`

ERRO

# Erro 60: Break não é permitido aqui

## Descrição

Este erro ocorre sempre que um `Break` foi executado em um lugar onde isso não é permitido. O `Break` só pode ser usado dentro de laços e dentro de uma ramificação `Switch`.

## Exemplo

```
$
break;  // Executar Break na área global
```

## Veja também

`DESTINYERROR_BREAKNOTALLOWEDHERE`

ERRO

# Erro 61: Continue não é permitido aqui

## Descrição

Este erro ocorre sempre que um `Continue` foi executado em um lugar onde isso não é permitido. O `Continue` só pode ser usado dentro de laços.

## Exemplo

```
$
continue;  // Executar Continue na área global…
```

## Veja também

`DESTINYERROR_CONTINUENOTALLOWEDHERE`

ERRO

# Erro 62: Handle inválido

## Descrição

Este erro ocorre sempre que, em vez de um handle de AuroraSheet, foi passado outro valor.

## Exemplo

```
$
// Copiar a área da imagem de outro AuroraSheet…
Picture[1].AuroraSheet.Clone("Dieser Parameter is
```

## Veja também

`DESTINYERROR_INVALIDHANDLE`

ERRO

# Erro 63: Operador esperado

## Descrição

Este erro ocorre sempre que, dentro de uma fórmula, se espera um operador (ou um ponto e vírgula para encerrar a fórmula), mas nenhum dos dois é informado.

## Exemplo

```
$
v[1] = 3  5;  // Nenhum operador foi informado
```

## Veja também

`DESTINYERROR_OPERATOREXPECTED`

ERRO

# Erro 64: Acesso ao slot de memória impossível

## Descrição

Ocorreu um erro ao acessar um slot de save.

> **Informação:** Antes de carregar um jogo salvo, verifica-se se o slot de save pode ser aberto para leitura. Se, ao salvar um jogo, o `RPG_RT` lançar uma exceção, este erro é lançado no lugar do erro `DESTINYERROR_EXCEPTION`.

Pode ser um erro do sistema de destino (por exemplo, quando não se tem os direitos necessários para acessar o arquivo/a pasta ou quando, ao salvar, a pasta do jogo é somente leitura), mas, ao carregar, também pode ocorrer porque o save indicado não existe.

> **Informação:** Para verificar, antes de carregar um save, se ele existe, pode-se usar o método `SlotExists` do objeto `Game`.

## Veja também

`DESTINYERROR_CANTACCESSSAVESLOT`

ERRO

# Erro 65: Exceção ocorrida

## Descrição

Ocorreu um erro no `RPG_RT`, que fez com que ele lançasse uma exceção. Alguns comandos são repassados ao `RPG_RT` (por exemplo, os métodos do objeto `Command`). Se ocorrer um erro grave na execução de um desses comandos (por exemplo, porque se quer carregar um gráfico que não existe), o `RPG_RT` normalmente travaria com uma mensagem de erro. Essa falha (provocada pela exceção) foi capturada pela `Destiny.dll`. Algo deu errado, mas a falta de um gráfico ou algo semelhante geralmente não é motivo para derrubar o jogo inteiro…

## Veja também

`DESTINYERROR_EXCEPTION`

ERRO

# Erro 66: Nível de operador de laço inválido

## Descrição

Este erro ocorre sempre que `Break` ou `Continue` (incluindo a indicação de nível) não se referem a um laço (ou, no caso do `Break`, a uma ramificação `Switch`). A contagem do nível sempre começa em 1. Os números de `Break` e `Continue` correm sempre em sincronia (ou seja, dentro de uma ramificação `Switch`, o nível do `Continue` também precisa ser aumentado em 1).

## Exemplo

```
$
for(v[1] = 0; v[1] < 10; v[1]++)
  continue(2);  // 2 se refere a um laço…
next;
```

## Veja também

`DESTINYERROR_INVALIDLOOPOPERATIONLEVEL`

ERRO

# Erro 67: Continue não se refere a um laço

## Descrição

Este erro ocorre sempre que um `Continue` (com ou sem indicação de nível) se refere a uma ramificação `Switch` (em vez de a um laço). O `Continue` só pode se referir a laços.

## Exemplo

```
$
switch(v[1])
  default:
    continue(1);  // Continue se refere a um laço…
endswitch;
```

## Veja também

`DESTINYERROR_CONTINUEDOESNTPOINTTOLOOP`

PARTE 5

# Diagnóstico

MessageLink

DestinyDebugger

# MessageLink

## Descrição

Se o MessageLink foi incluído no `RPG_RT` (veja a opção correspondente no DestinyPatcher), ficam disponíveis alguns novos marcadores (placeholders) nas caixas de mensagem para as áreas de memória da `Destiny.dll`. A formatação dos marcadores é em grande parte idêntica à do método `Format` do objeto `String` (com a pequena diferença de que uma barra invertida ( \ ) é escrita antes e um índice depois). No total, o MessageLink acrescenta quatro novos marcadores:

- `A` para strings
- `D` para Dwords
- `F` para Floats
- `( … )` para código de script qualquer

(As letras são idênticas às abreviações das áreas de memória no código DestinyScript.)

Exemplo de uma caixa de mensagem:

```
A primeira string: "\a[1]"
O segundo Dword: \d[2]
O terceiro Float: \f[3]
O nome do primeiro herói: \(Actor[1].Name)
```

A formatação desses marcadores é exatamente igual à dos marcadores padrão do RPG Maker 2000. Cada um dos três marcadores, porém, tem as suas próprias características.

## O marcador de string

### Descrição

O marcador de string é usado para exibir no texto da mensagem o conteúdo de uma variável string (`a[]`).

> **Atenção:** quebras de linha na string não são aproveitadas como tal na caixa de mensagem. Isso ocorre porque o RPG Maker guarda internamente cada linha da caixa de mensagem separadamente. No lugar da quebra de linha aparecem alguns caracteres especiais.

### Exemplo

```
$
a[1] = "Linha 1" + CRLF + "Linha 2";
```

Com a caixa de mensagem:

```
"\a[1]"
```

Resulta na seguinte saída:

*Saída do marcador de string*

Como se vê, os dois caracteres da quebra de linha (Carriage Return e Line Feed) foram exibidos como setas. Para exibir uma string em várias linhas, é preciso dividi-la antes. Para isso são recomendados os métodos `Pos` e `SubStr` do objeto `String`.

## O marcador de Dword

### Descrição

O marcador de Dword é usado para exibir no texto da mensagem o conteúdo de uma variável Dword (`d[]`).

Esse marcador pode exibir números com uma quantidade mínima de dígitos. Para isso, basta escrever, depois da letra do marcador, a quantidade mínima de dígitos.

### Exemplo

```
$
d[1] = 1500;
```

Com a caixa de mensagem:

```
\d[1]
\d6[1]
\d2[1]
```

Resulta na seguinte saída:

*Saída do marcador de Dword*

Na primeira linha, o comprimento do número exibido foi determinado automaticamente. Na segunda linha foi indicada uma quantidade mínima de dígitos que o número, sozinho, não alcançava; por isso foram acrescentados 2 zeros à frente. Na terceira linha, o número ultrapassou a quantidade mínima de dígitos, e por isso ficou com mais de 2 dígitos.

## O marcador de Float

### Descrição

O marcador de Float é usado para exibir no texto da mensagem o conteúdo de uma variável Float (`f[]`).

Esse marcador pode exibir números com uma quantidade mínima de casas antes da vírgula e também com uma quantidade exata de casas depois da vírgula. Para isso, basta escrever, depois da letra do marcador, a quantidade mínima de casas antes da vírgula e a quantidade de casas depois da vírgula, separadas por um ponto.

### Exemplo

```
$
f[1] = 123.456;
```

Com a caixa de mensagem:

```
\f[1]
\f4[1]
```

```
\f0.2[1]
\f4.4[1]
```

Resulta na seguinte saída:

*Saída do marcador de Float*

Na primeira linha, o comprimento da parte inteira e da parte decimal do número foi determinado automaticamente. Na segunda linha foi indicada uma quantidade mínima de dígitos para a parte inteira que o número, sozinho, não alcançava; por isso foi acrescentado um zero à frente. Na terceira linha foram indicados no mínimo 0 dígitos na parte inteira (= modo normal) e exatamente 2 casas decimais. Na quarta linha foram indicados no mínimo 4 dígitos na parte inteira e exatamente 4 casas decimais, sendo que se acrescentou um zero à frente da parte inteira para alcançar a quantidade mínima de dígitos.

> **Informação:** se o número for definido de modo que não deva haver nenhuma casa decimal (por exemplo, `\f0.0[1]`), o separador decimal também não é exibido. (Por exemplo: 123, a partir do número original 123,456.)

## O marcador de script

O marcador de script é indicado por um abre-parênteses seguido de um fecha-parênteses ( ). Dentro dos parênteses fica uma expressão de fórmula que possa ser convertida em string.

> **Atenção:** não é possível acessar referências `THIS`, pois essa informação não está disponível dentro da saída de mensagens.

### Exemplo

Com a caixa de mensagem:

```
Nome: \(Actor[1].Name)
Título: \(Actor[1].Degree)
\("Isto é uma string")
Hora: \(String.Format("D2", Time.Hour) .. ":" ..…
```

O RPG Maker precisa estar ajustado para conseguir inserir a última linha (para o caso normal ela é um pouco longa demais).

Resulta na seguinte saída:

*Saída do marcador de script*

# DestinyDebugger

## Descrição

Como já descrito no objeto `Destiny.Debugger`, existe uma possibilidade de depurar (debugar) código DestinyScript. Não é possível, porém, pausar/continuar scripts (portanto, a funcionalidade é bastante limitada).

Existem três maneiras de depurar código DestinyScript:

1. A exibição de erros
2. O DestinyDebugger
3. A saída de depuração

GUIA

# Exibição de erros

## Descrição

Quando ocorre um erro em um script, por padrão é aberta uma janela que mostra o erro e o script que o causou. Esse comportamento pode ser alterado com os objetos `Errors` e `Error[]`.

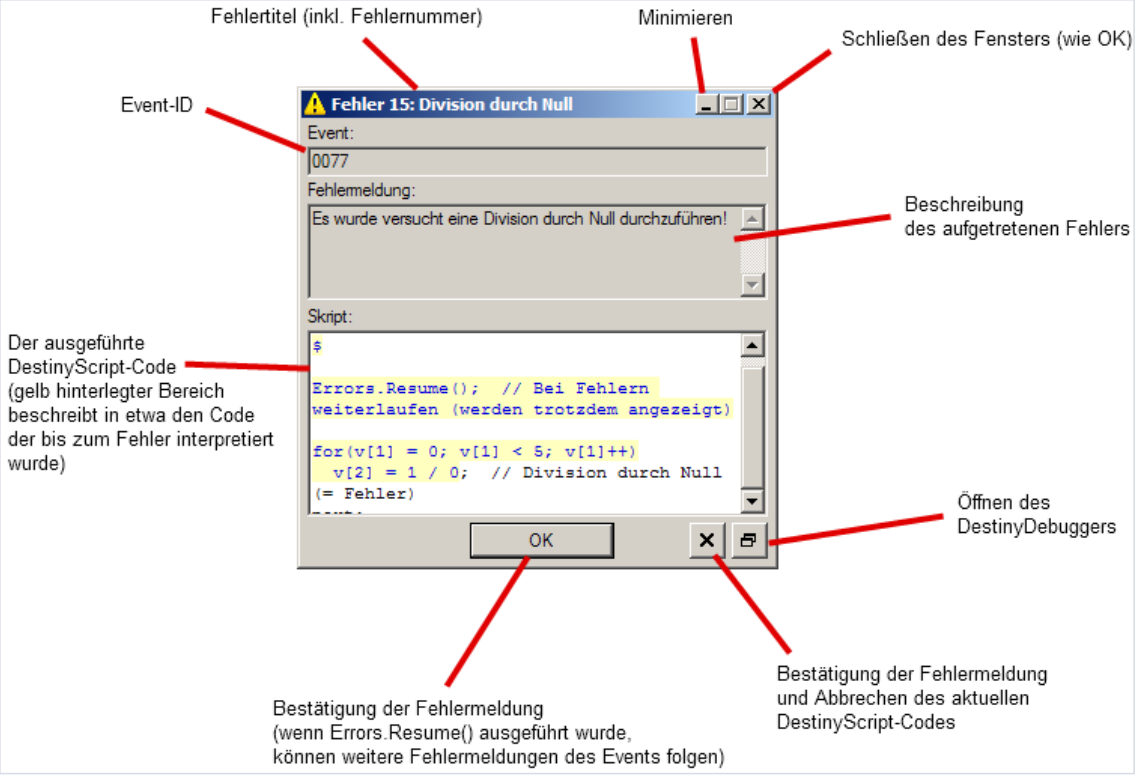
Como exemplo para o gráfico usado, considere o seguinte código DestinyScript com erro:

```
$

Errors.Resume();  // Continuar em caso de erros (…

for(v[1] = 0; v[1] < 5; v[1]++)
  v[2] = 1 / 0;  // Divisão por zero (= erro…
next;
```

Ao executar esse script, aparece a seguinte janela:




*Janela de exibição de erros*

O importante aqui é a diferença entre **OK** e o botão com a cruz:

- Com **OK** (ou ao fechar a janela do modo habitual), a mensagem é apenas confirmada. Se, como no exemplo, foi executado `Errors.Resume()`, ainda podem vir outras mensagens de erro do mesmo código DestinyScript (aqui o mesmo erro é executado cinco vezes por causa do laço).
- Com **X** (o botão ao lado de OK), a mensagem é confirmada e o DestinyScript atual é cancelado (independentemente de `Errors.Resume()` ter sido executado).

GUIA

# A janela do DestinyDebugger

## Descrição

Se, na exibição de erros, você clicar no botão que abre o DestinyDebugger, ou executar `Destiny.Debugger.Open()`, o DestinyDebugger é aberto. Ele permite, de modo semelhante à janela de depuração do `RPG_RT`, ver o conteúdo das propriedades de todos os objetos.

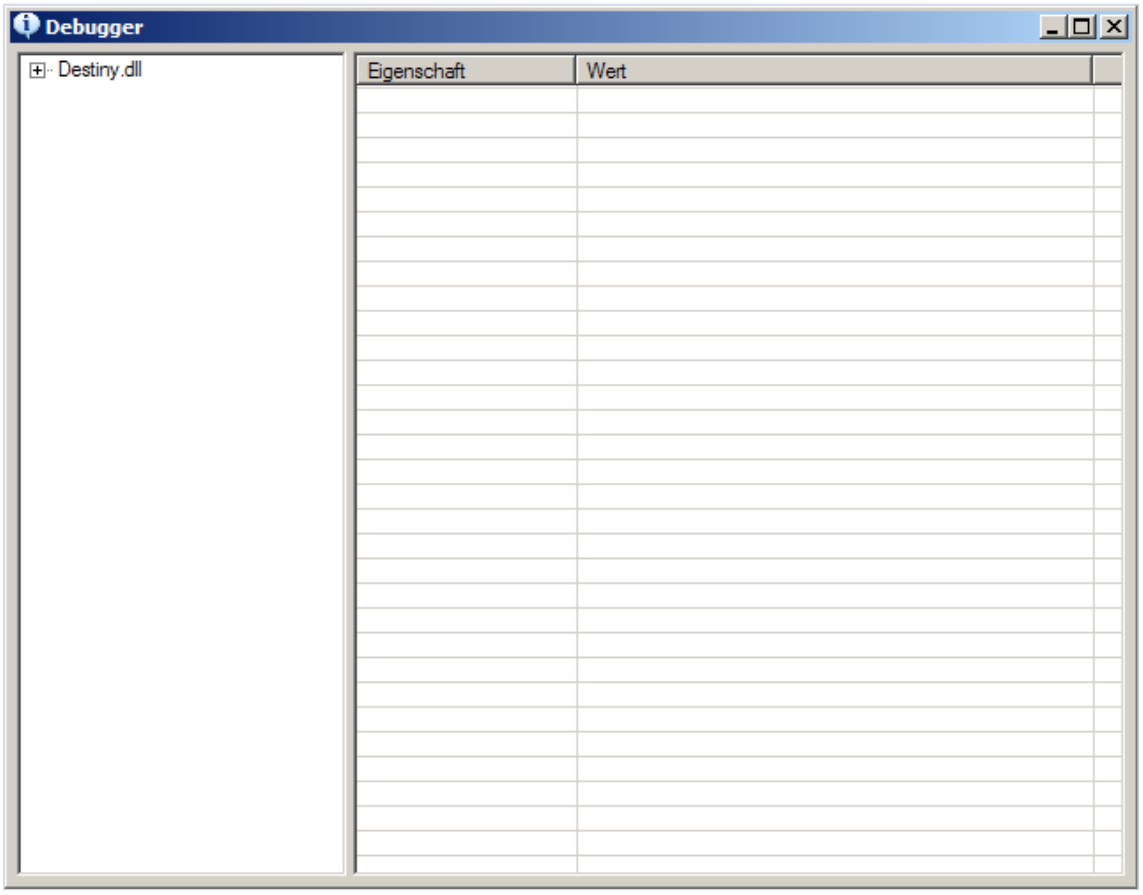
Como exemplo para o gráfico usado, considere o seguinte código DestinyScript:

```
$
// Inicializar algumas variáveis
v[1] = 123;
a[1] = "Olá";
a[2] = "Linha 1" .. CRLF .. "Linha 2";
Picture[1].LoadFromFile("Cloud");

// Abrir o DestinyDebugger
Destiny.Debugger.Open();

// Apagar a primeira Picture novamente
Picture[1].AuroraSheet.Free();
```

Durante a execução do script, o DestinyDebugger é aberto. Aparece a seguinte janela:

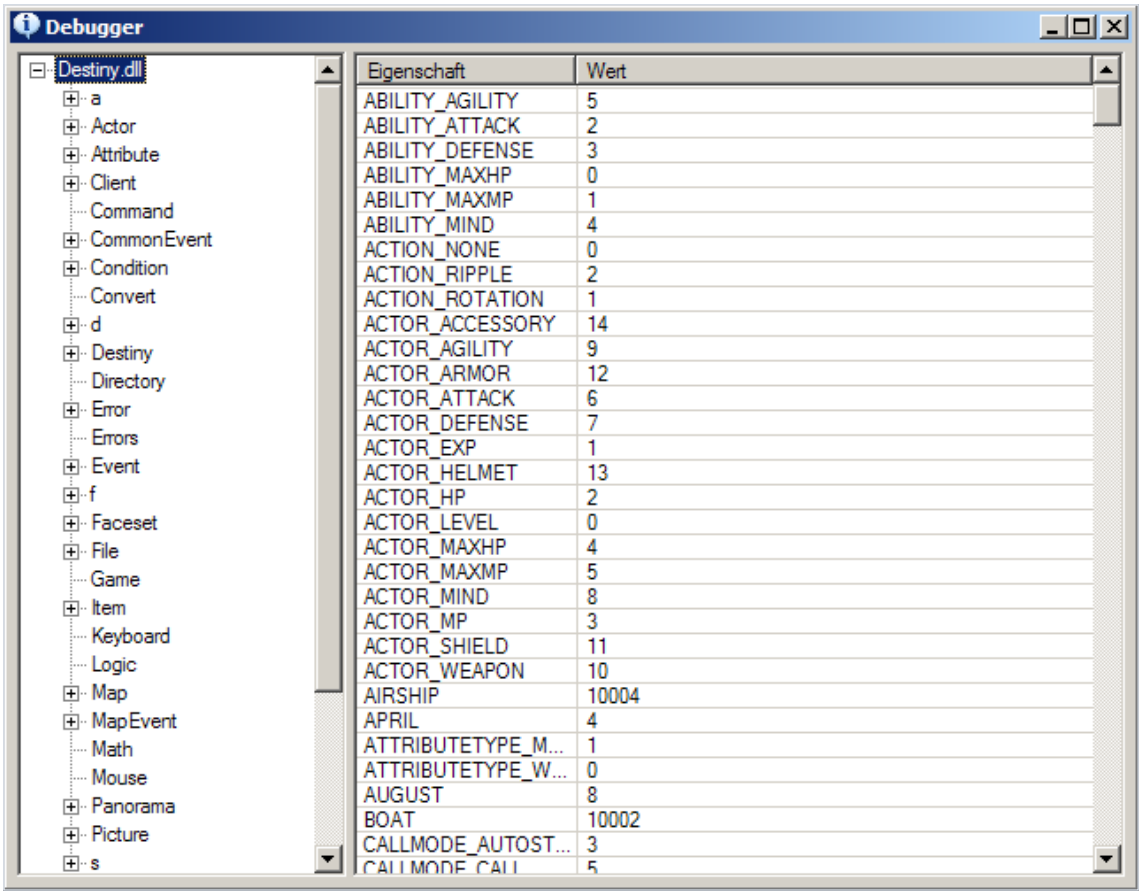



*DestinyDebugger recém-aberto*

Essa janela é claramente dividida em duas áreas:

- **Esquerda:** uma estrutura em árvore para escolher o objeto atual.
- **Direita:** uma lista de todas as propriedades/constantes do objeto atualmente selecionado. Aqui **não** são exibidos métodos!

Como se vê na captura de tela, à primeira vista a lista está vazia. Isso acontece porque, na estrutura em árvore (à esquerda), ainda nenhum objeto foi selecionado. Se você selecionar o objeto mais alto (`Destiny.dll`) e também expandi-lo, a lista fica assim:




*Objeto Destiny.dll selecionado*

## Exemplo: visualizando valores normais

Se, por exemplo, você navegar até o objeto `v`, verá abaixo dele a entrada `[...]`. Ela corresponde à lista dentro do objeto `v`, isto é, `v[]`.

Ao selecionar a entrada `[...]`, aparecem na lista da direita todas as variáveis dentro do objeto `v[]`. Como se vê, também aparece o valor atribuído à primeira variável no script de exemplo:

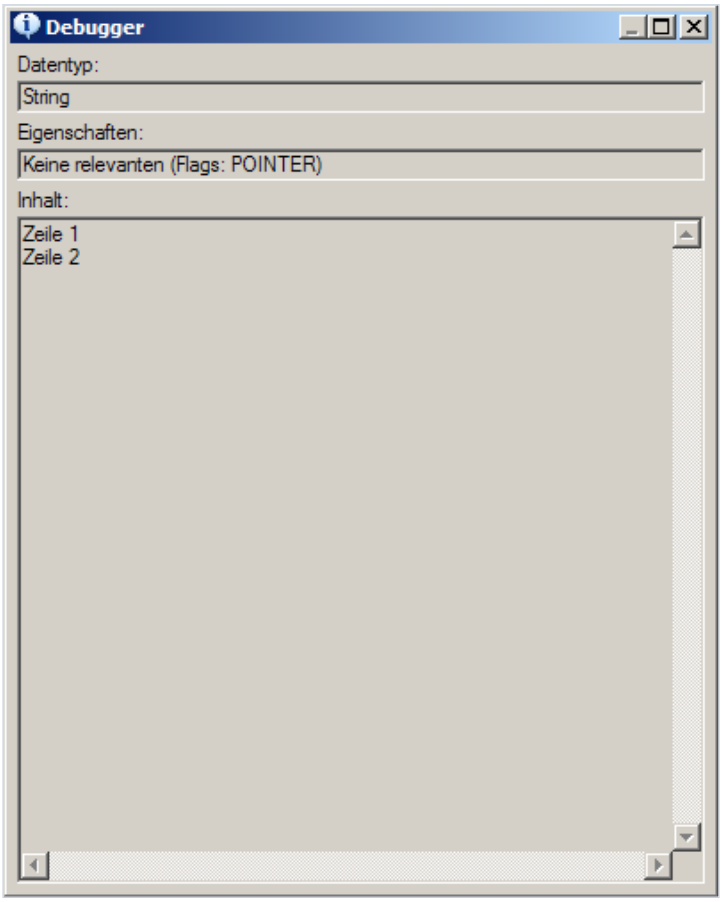
![Lista de variáveis v[]](img/fig-7eb1bdbc.png)

### Exemplo: visualizando valores de várias linhas

Se em seguida você navegar até o objeto `a`, ele terá, em princípio, a mesma estrutura do objeto `v`. Abaixo do objeto `a` fica a lista de valores (`a[]`). Ao selecionar essa entrada, aparecem as seguintes linhas:

![Lista de strings a[]](img/fig-4279a9f5.png)

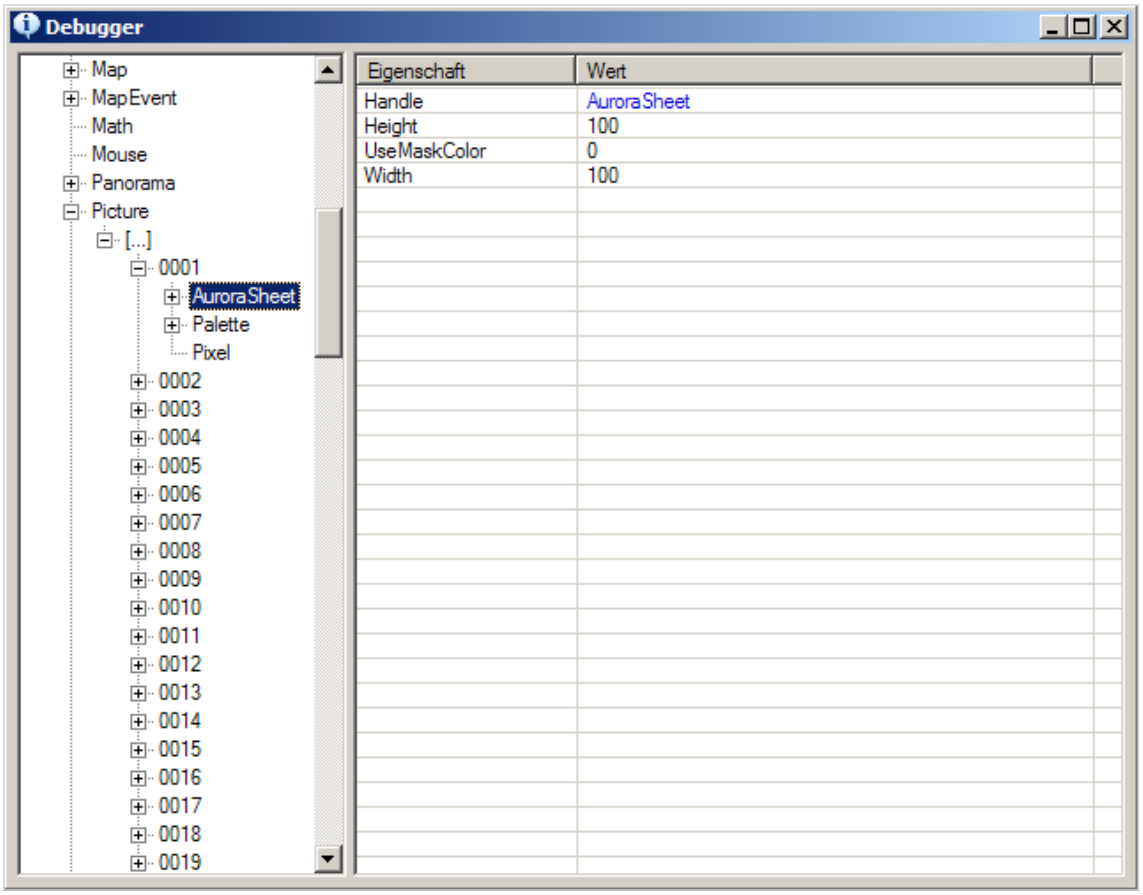
Como se vê, uma das entradas está colorida de **verde**. Uma entrada verde indica que ali há um valor de várias linhas. Como isso não pode ser representado de outra forma na lista (em que cada entrada tem apenas uma linha), a entrada é colorida. Para ver o valor exato, dê um **duplo clique** na respectiva linha da lista da direita. Abre-se então uma nova janela, na qual o valor selecionado pode ser examinado com mais detalhe:




*Valor de várias linhas*

## Exemplo: visualizando AuroraSheets

Se em seguida você navegar até o objeto `Picture`, a primeira mudança perceptível é que, ao selecionar a entrada `[...]`, a lista da direita fica vazia. Em vez disso, essa entrada pode ser expandida novamente. Isso acontece porque cada entrada na lista de Pictures é, ela própria, um objeto. Por isso, neste caso não é possível exibir os valores da lista da direita simplesmente como valores individuais. Se você selecionar agora um dos objetos `Picture[]` (no exemplo, a primeira Picture), poderá ver as propriedades dessa primeira Picture. Há, porém, uma particularidade nos objetos AuroraSheet. Se você selecionar o objeto AuroraSheet dentro da primeira Picture, verá que uma das entradas aparece em **azul**:




*Objeto AuroraSheet com o handle em azul*

Essa entrada é o handle do AuroraSheet, que também é necessário para copiar gráficos de outros AuroraSheets. Para ver o gráfico que está por trás do AuroraSheet, dê um duplo clique na respectiva linha da lista da direita. Abre-se então, como no exemplo anterior do texto de várias linhas, a janela de visualização do valor individual. Ali é possível ver o gráfico dentro do AuroraSheet:

*Gráfico dentro do AuroraSheet*

GUIA

# Saída de depuração

## Descrição

Usando as propriedades e os métodos do objeto `Destiny.Debugger.Output`, é possível criar uma saída de depuração. Para isso, basta adicionar texto com o método `Print` ou com a propriedade `Text` e exibir a saída depois com o método `Show`.

Como exemplo para o gráfico usado, considere o seguinte código DestinyScript:

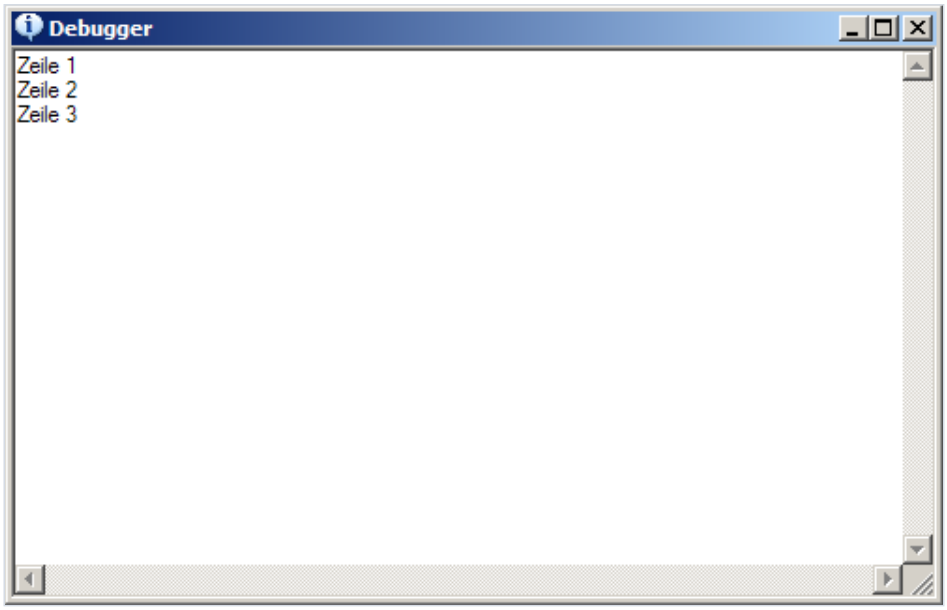
```
$

// Limpar o conteúdo da saída de depuração
Destiny.Debugger.Output.Clear();

// Inserir algumas linhas na saída de depuração
Destiny.Debugger.Output.Print("Linha 1");
Destiny.Debugger.Output.Print("Linha 2");
Destiny.Debugger.Output.Print("Linha 3");

// Exibir a saída de depuração
Destiny.Debugger.Output.Show();
```

Durante a execução do script, a janela de saída de depuração é aberta e mostra o seguinte:




*Janela de saída de depuração*

Como se vê, todo texto adicionado é exibido no campo de entrada.

PARTE 6

# Apêndice

Apêndice
Palavras finais
Informações legais (Impressum)

# Apêndice

Este apêndice reúne informações adicionais sobre o DestinyPatch:


- Alterações da linguagem de script
- Protocolo utilizado nos sockets do Destiny
- Comparação com outros patches
- Cálculo de chips inferiores e superiores (Lower/Upper)


GUIA

## Alterações da linguagem de script

### Correções de erros da versão 2.0.1.8 em comparação com a versão 2.0.0.0

- `Picture[].ActionValue`++ causava uma travada do programa
- `Destiny.Debugger.Output.Text` causava uma travada do programa
- Chamadas em que todos os parâmetros são opcionais não funcionavam (por exemplo, `Client[].Raw.Recv()` ou `Keyboard.GetKey()`)
- `Client[].Raw.Recv()` lançava o erro `DESTINYERROR_MEMORY` quando não havia dados pendentes e o tipo de dado de destino era String
- Em `Server.Accept()`, alguns valores não eram reiniciados
- `Destiny.Load()` deixava handles de arquivo abertos, de modo que um slot Destiny só podia ser carregado uma vez por jogo

### Melhorias da versão 2.0.1.8 em comparação com a versão 2.0.0.0

- `Event[].CanMoveTo()` foi adicionado
- `Event[].MoveRoute` foi adicionado

### Alterações da versão 2.0 em comparação com a versão 1.0

- "Double" foi substituído por "Float" em todas as descrições, já que Double e Dword (ambos começam com D) causavam confusão
- "Função" foi substituído por "Método" em todas as descrições, já que todos os métodos estão vinculados a objetos
- Os métodos do objeto `Logic` agora avaliam Strings como vazia/não vazia, em vez de "0" ou "1"
- `String.SubStr()` passou a aceitar valores negativos como posição inicial (comportamento proposital, mas agora interpretado de forma diferente)
- `String.Pos()` agora devolve a posição inicial quando a string de busca está vazia
- O parâmetro de posição inicial de `String.Pos()` agora é opcional
- O parâmetro de comprimento de `String.SubStr()` agora é opcional
- `String.Fill()` agora aceita qualquer quantidade de caracteres (o produto não pode ultrapassar 2 GiB; o limite anterior era 10000)
- `String.WeekdayName()` e `String.MonthName()` agora têm "Idioma" como parâmetro adicional opcional
- `Math.Sin()` (e métodos semelhantes) agora têm o parâmetro de formato como opcional (o valor padrão é RAD)
- `Math.Sqrt()` agora tem um parâmetro adicional opcional para indicar o índice da raiz
- `Math.Round()` (e métodos semelhantes) agora têm o parâmetro da casa de arredondamento como opcional

- `Math.Int()` agora tem o sinônimo `Math.Integer()`
- `Math.Scale()` agora tem os sinônimos `Math.Decimal()` e `Math.Dec()`
- `Math.RoundUp()` agora tem `Math.Ceil()` como sinônimo
- `Math.RoundDown()` agora tem `Math.Floor()` como sinônimo
- `Keyboard.GetKey()` e `Keyboard.GetKeyText()` agora têm um parâmetro adicional opcional que indica onde a busca pela tecla pressionada começa
- `File[].SetFilePointer()` agora tem um parâmetro adicional opcional para o ponto de referência
- `File[].WriteRawData()` e `File[].ReadRawData()` agora têm o parâmetro de comprimento como opcional
- `Game.SlotExists()` foi adicionado
- `Convert.Double()` agora tem o sinônimo `Convert.Float()` (o nome antigo agora é obsoleto)
- `Convert.Switch()` agora tem o sinônimo `Convert.Bool()` (o nome antigo agora é obsoleto)
- `Destiny.Language` agora pode ser alterado em tempo de execução (antes isso não funcionava, embora estivesse documentado de forma diferente, ou seja, errada)
- `Destiny.Save()` agora tem um parâmetro adicional opcional para indicar as áreas de memória a salvar
- O parâmetro de comprimento de `Client[].SendRawData()` agora é opcional
- O parâmetro de tipo de socket de `Server.Listen()` agora é opcional
- `Picture[].ActionValue` agora é arredondado para a terceira casa decimal ao ser gravado, para que ângulos como 90° (que, na sintaxe de ângulo do RPG Maker, resultam em uma fração periódica) sejam exibidos corretamente (obrigado pela informação, Cherry)
- O status de socket `STATE_CLOSING` foi adicionado (para permitir o acesso aos dados de um socket recém-fechado)
- `Client[].SendRawData()` agora não envia nada quando o comprimento é igual a 0. Antes, era enviado o comprimento máximo, o que não estava documentado. Se o parâmetro estiver ausente, agora é enviado o comprimento máximo (nesse caso, o valor do parâmetro é -1)
- `Event[].Dir1` foi renomeado para `Event[].DirMove` (foi adicionado um sinônimo para `Event[].Dir1`)
- `Event[].Dir2` foi renomeado para `Event[].DirLook` (foi adicionado um sinônimo para `Event[].Dir2`)
- `String.Chr()` agora também aceita o parâmetro 0 (alteração feita na versão 1.0.6)
- A mensagem de erro para `DESTINYERROR_NOVALUE` foi ajustada para ficar mais clara (alteração feita na versão 1.0.6)
- `Directory.FindFirst()` agora devolve uma string vazia quando nenhum arquivo corresponde ao padrão (alteração feita na versão 1.0.6)

## Melhorias da versão 2.0 em comparação com a versão 1.0

- Foram adicionados alguns elementos de estrutura (if, switch, do, for, break, continue, pause, exit)
- Foi adicionada a possibilidade de inserir comentários
- Foram adicionadas possibilidades extras de indicar strings
- Foi adicionada a possibilidade de indicar código de script no MessageLink
- Foi adicionado um depurador simples

**a** — `a.Count` e `a.ReAlloc()` foram adicionados

**d** — `d.Count` e `d.ReAlloc()` foram adicionados

**f** — `f.Count` e `f.ReAlloc()` foram adicionados

**v** — `v.Count` foi adicionado

**s** — `s.Count` foi adicionado

**Actor** — `Actor.Count` foi adicionado

**Actor[]** — foram adicionados Weapon, Shield, Armor, Helmet, Accessory, Charframe, Charset, Transparency, Faceframe, Faceset, Default, CriticalHitChance, FixedEquipment, StrongDefense, TwoSwordStyle, ControlledByAI, MaxLevel, AttributeEffect, LevelAttribute, Condition, ConditionEffect, Skill, HasCondition(), HasSkill(), UseSkillName, UnarmedBattleAnimation

- O objeto Attribute foi adicionado

**Client** — Count, ReAlloc(), NextFree(), CloseAll() foram adicionados

**Client[]** — Destiny e Raw foram adicionados

- Os objetos Command, CommonEvent e Condition foram adicionados

**Destiny** — Implemented() e Debugger foram adicionados

**Directory** — FindLevel e FindCloseAll() foram adicionados

**Error** — Count, DisableAll(), EnableAll() foram adicionados

**Error[]** — Raise() foi adicionado

**Event e MapEvent** — Event.Count foi adicionado; em Event[] foram adicionados Name, Charframe, FrameSimple, AuroraSheet

- O objeto Faceset foi adicionado

**File** — Count, ReAlloc(), NextFree(), CloseAll() foram adicionados

**Game** — RM2K, RM2K3, SlotExists() foram adicionados

- O objeto Item foi adicionado

**Map** — AuroraSheet, NumberOfSaves, ScreenX, ScreenY, ScrollType foram adicionados

**Math** — RandSeed e Rand() foram adicionados

- O objeto Panorama foi adicionado

**Picture** — Count foi adicionado; em Picture[] foram adicionados Filename, LoadFromFile() (alteração feita na versão 1.0.6), SaveAs() (alteração feita na versão 1.0.6), AuroraSheet

- Os objetos Screen e Skill foram adicionados

**String** — CompareLex() foi adicionado

## Correções de erros da versão 2.0 em comparação com a versão 1.0

- Keyboard.GetKey() não conseguia devolver 0 quando nenhuma tecla estava pressionada
- Math.Sin() (e métodos semelhantes) lançavam o erro DESTINYERROR_RANGE com um formato inválido (o correto agora é DESTINYERROR_VALUE)
- Math.Round() (e métodos semelhantes) aceitavam números quebrados como indicação de casa (agora é preciso usar um número inteiro)
- Math.Decimal(), no erro DESTINYERROR_FLOATERROR, deixava valores na pilha do FPU, o que podia travar o jogo
- String.Replace() terminava ao encontrar um caractere NULL
- String.Replace() podia criar uma string NULL
- String.ToUpper() e String.ToLower() criavam strings NULL
- String.WeekdayName() e String.MonthName(), em circunstâncias especiais, podiam gerar mensagens de erro incorretas
- Ocorria um erro interno insignificante ao carregar Pictures, pois um valor antes reservado era usado várias vezes

- `Client[]`, com um índice incorreto, lançava o erro `DESTINYERROR_VALUE` (o correto agora é `DESTINYERROR_ARRAYBOUND`)
- Em todos os métodos `Client[].RecvId()` (e outros métodos Recv), não se verificava se a conexão realmente existia
- Em circunstâncias especiais, ao receber dados pelo objeto `Client`, a pilha ficava inválida (o que travava o jogo)
- `Convert.Angle()` lançava o erro `DESTINYERROR_RANGE` com formatos de ângulo inválidos (o correto agora é `DESTINYERROR_VALUE`)
- `Client[].Connect()` e `Server.Listen()`, em circunstâncias especiais, simplesmente não funcionavam
- `String.Trim()` dava resultados errados quando o texto restante tinha apenas um caractere
- `String.Pos()` dava um resultado errado quando o resultado era pequeno e aparecia no fim da string de busca
- `String.Format()` nem sempre gerava a quantidade correta de dígitos antes da vírgula ao formatar números de ponto flutuante
- `String.Format()`, ao formatar o número de ponto flutuante 0.0, sempre gerava exatamente uma casa decimal
- `String.WeekdayName()`, em inglês, dava um resultado errado para domingo (na forma abreviada: "Son" em vez de "Sun") — por isso a versão antiga não era compatível com a RFC-2616
- `String.MonthName()` não abreviava junho e julho em alemão e inglês (na forma abreviada) — por isso a versão antiga não era compatível com a RFC-2616
- Se o jogo fosse iniciado por um atalho com diretório de trabalho incorreto, todas as funções de arquivo (`File[]` e `Directory`) acessavam diretórios errados
- Números de ponto flutuante muito grandes eram exibidos com notação exponencial incorreta ao serem convertidos em string (por exemplo: `Convert.String(123.4 * Math.Exp(100))` resultava em `1,234e+12`, quando o correto seria `1,234e+102`)
- Ao converter repetidamente o número de ponto flutuante 0.0 em string, valores ficavam na pilha do FPU, o que podia travar o jogo
- Em certos casos, `Client[].State` devolvia o valor `STATE_CLOSED`, mas um `Client[].Connect()` seguinte ainda assim gerava um erro (o socket estava fechado, mas o handle ainda estava aberto)
- `Destiny.Save()` e `Destiny.Load()` lançavam o erro `DESTINYERROR_RANGE` com valores grandes/pequenos demais (o correto agora é `DESTINYERROR_VALUE`)
- `Math.Sin()` (e métodos semelhantes), com um formato de ângulo inválido, deixavam valores na pilha do FPU, o que podia travar o jogo
- No erro `DESTINYERROR_FLOATERROR` (quando ocorria), ficavam valores na pilha do FPU, o que podia travar o jogo
- `Math.Tan()` (e métodos semelhantes de tangente) produziam um erro de "divisão por zero" com determinados valores
- `Game.Save()` e `Game.Load()` lançavam o erro `DESTINYERROR_RANGE` com um slot inválido como parâmetro (o correto agora é `DESTINYERROR_VALUE`)
- `Game.Save()` e `Game.Load()` travavam o jogo quando salvar/carregar falhava (ao salvar, o erro agora é capturado; ao carregar, agora se verifica antes se o respectivo arquivo pode ser aberto)
- `Convert.Angle()` lançava o erro `DESTINYERROR_RANGE` (o correto agora é `DESTINYERROR_VALUE`)
- `Math.Power()` e `Math.Sqrt()`, com zero e com números negativos (como base), em certos casos travavam o jogo
- `Math.Log()` (e métodos semelhantes), com zero e com números negativos, travavam o jogo
- `Event[].JumpTime` sempre apontava para um valor errado na memória, e por isso era ainda mais inútil do que já era
- `Event[].Speed` travava o jogo quando o valor era grande demais. Agora o valor está limitado à faixa de 1 a 6
- `Convert.String()` não funcionava corretamente (correção feita na versão 1.0.2)
- Ao encerrar o jogo, sempre aparecia uma mensagem de erro (correção feita na versão 1.0.2)

- No Windows XP com SP1, não era possível consultar o mouse (correção feita na versão 1.0.3)
- Os servidores só conseguiam aceitar a conexão de um único cliente (correção feita na versão 1.0.6)
- `Game.Load()` não funcionava com eventos AutoStart e eventos PushKey (correção feita na versão 1.0.6)
- Alguns pequenos erros nas mensagens de erro foram corrigidos (por exemplo, algumas mensagens de erro nunca apareciam em inglês) (correção feita na versão 1.0.6)
- A quantidade de imagens era limitada a 250 pela Destiny.dll, o que causava problemas com patches mais novos (correção feita na versão 1.0.8)
- Em circunstâncias especiais, deixava de ser possível exibir mensagens de erro (correção feita na versão 1.0.8)
- Em combinação com `String.Chr(0)`, os dados ficavam gravados incorretamente na memória numa concatenação (por exemplo, `a[1] = "Test" + String.Chr(0) + "Test2"` gerava, na memória de a[1], os dados `"TestTest2" + String.Chr(0)`) (correção feita na versão 1.0.10)
- Em combinação com `String.Chr(0)`, `String.Replace()` não funcionava corretamente (correção feita na versão 1.0.10)

GUIA

# Protocolo utilizado nos sockets do Destiny

## Descrição

O protocolo Destiny foi desenvolvido por Bananen-Joe especialmente para o RPG Maker. Ele oferece as funções básicas para enviar variáveis, switches, Dwords, Floats, Words, Bytes e strings. Todos eles são associados a um ID e se dividem em 8 comandos (um para cada tipo de dado, mais 1 extra para switches). Todos os valores enviados usam, como é comum em processadores I386, o formato "Little Endian".

> **Informação:** o protocolo Destiny não mudou em relação ao DestinyPatch 1.0.

## Estrutura de um comando do protocolo Destiny

```
B IIII W...
```

A estrutura é bem simples:

1. Primeiro é enviado 1 byte como comando (**B**). Ele determina o formato e o comprimento do valor.
2. O primeiro parâmetro do comando é um Dword (little endian) que corresponde ao ID associado (**IIII**).
3. O segundo parâmetro do comando é o valor no respectivo tipo de dado (**W...**).

## Lista de comandos do protocolo Destiny

| Byte do comando | Comando | Descrição | Formato do valor | Tamanho do valor (bytes) |
|---|---|---|---|---|
| `V` | `Client[].Destiny.SendVariable()` | O valor de uma variável ( `v[]`) | Dword com sinal | 4 |
| `S` | `Client[].Destiny.SendSwitch()` | Um switch com o valor `true` (`s[]`) | nenhum | 0 |
| `s` | `Client[].Destiny.SendSwitch()` | Um switch com o valor `false` (`s[]`) | nenhum | 0 |
| `D` | `Client[].Destiny.SendDword()` | O valor de um Dword ( `d[]`) | Dword com sinal | 4 |

| Byte do comando | Comando | Descrição | Formato do valor | Tamanho do valor (bytes) |
|---|---|---|---|---|
| `F` | `Client[].Destiny.SendFloat()` | O valor de um Float (`f[]`) | Float (Double) | 8 |
| `W` | `Client[].Destiny.SendWord()` | O valor de um Word | Word com sinal | 2 |
| `B` | `Client[].Destiny.SendByte()` | O valor de um Byte | Byte sem sinal | 1 |
| `A` | `Client[].Destiny.SendString()` | O valor de uma string (curta) (`a[]`) | Veja o formato de string | 1 + n |

## Formato de string

No formato de string, os dados naturalmente não são números; por isso o formato é um pouco diferente. O primeiro byte (depois do índice) indica o comprimento da string. Os bytes seguintes são o conteúdo da string.

Daí se percebe a fraqueza desse formato: como o comprimento é indicado por um byte, o comprimento máximo de uma string a enviar é de **255 caracteres**. O motivo dessa limitação foi a simplificação do código do programa. (Ou seja: o programador teve preguiça de pensar numa solução melhor...)

> **Informação:** para enviar strings mais longas, é possível "dar um jeito", por exemplo dividindo a string em partes de 255 caracteres cada (exceto a última parte, que pode ser mais curta). As partes seguintes podem ser enviadas com o ID 0, de modo que o valor recebido depois possa ser anexado à string anterior.

## Exemplos

Os exemplos a seguir se referem aos métodos de envio do objeto `Client[].Destiny`.

### SendVariable

```
$
Client[0].Destiny.SendVariable(1, 2);
```

O resultado do exemplo de SendVariable produziria o seguinte pacote de dados (hex dump):

```
01 00 00 00 02 00 00 00
```

### SendSwitch

```
$
Client[0].Destiny.SendSwitch(100, true);
Client[0].Destiny.SendSwitch(-200, false);
```

O resultado do exemplo de SendSwitch produziria o seguinte pacote de dados (hex dump):

```
64 00 00 00
38 FF FF FF
```

### SendDword

```
$
Client[0].Destiny.SendDword(10000, 0x12345678);
```

O resultado do exemplo de SendDword produziria o seguinte pacote de dados (hex dump):

```
10 27 00 00 78 56 34 12
```

### SendFloat

```
$
Client[0].Destiny.SendFloat(0xAABBCCDD, Math.Pi);
```

O resultado do exemplo de SendFloat produziria o seguinte pacote de dados (hex dump):

```
DD CC BB AA 18 2D 44 54 FB 21 09 40
```

### SendWord

```
$
Client[0].Destiny.SendWord(0x987654, 100);
```

O resultado do exemplo de SendWord produziria o seguinte pacote de dados (hex dump):

```
54 76 98 00 64 00
```

### SendByte

```
$
Client[0].Destiny.SendByte(0xD0C0B0A0, 3);
```

O resultado do exemplo de SendByte produziria o seguinte pacote de dados (hex dump):

```
A0 B0 C0 D0 03
```

### SendString

```
$
Client[0].Destiny.SendString(1111111, "Hallo");
```

O resultado do exemplo de SendString produziria o seguinte pacote de dados (hex dump):

```
47 F4 10 00 05 48 61 6C 6C 6F
```

> **Observação da tradução:** no manual original, os exemplos chamavam os métodos diretamente em `Client[0]` (e o de Float usava o nome antigo `SendDouble`); aqui foram ajustados para os nomes atuais em `Client[].Destiny`, como diz o texto. Além disso, os hex dumps do original mostram apenas o ID e o valor, sem o byte de comando descrito acima.

GUIA

# Comparação com outros patches

## Introdução

Este documento pretende dar uma visão geral rápida dos patches mais populares para o RPG Maker atualmente, e traçar as diferenças entre esses patches e o DestinyPatch.

## Power Patch

### Geral

Este patch foi criado por Cherry (http://www.cherrytree.at/) e oferece uma grande quantidade de funções. Como linguagem de script, usa-se aqui o Lua, que também está disponível em outros programas. Uma particularidade é que esse patch não modifica o `RPG_RT`, e sim a memória do programa em tempo de execução. Por isso, o código de script roda em paralelo aos comandos de evento (para sincronizar, portanto, é preciso usar tempos de espera). Além do Power Patch, Cherry também criou vários outros patches, como o Hyper Patcher (1 e 2), o Power Patch Compact, o CharExpandPatch, o PicPointerPatch etc. Alguns desses patches têm funcionalidades diferentes e/ou mais amplas. Com alguns desses patches (os que também modificam o `RPG_RT`), o DestinyPatch não é compatível (ou só por acaso).

### Comparação com o DestinyPatch 2.0

#### Script

| | Power Patch | DestinyPatch |
|---|---|---|
| Processamento do código de script | paralelo | serial (em relação aos eventos) |
| Ramificações | sim | sim |
| Laços | sim | sim |
| Funções próprias | sim | não |
| Classes próprias | não | não |
| Incluir scripts | sim | não |

#### Funções disponíveis

| | Power Patch | DestinyPatch |
|---|---|---|
| Matemática | sim (com o plugin StdLib) | sim |
| Strings | sim (com o plugin StdLib) | sim |
| Operações lógicas | sim | sim |
| Operações binárias | não | sim |

#### Acesso ao sistema

| | Power Patch | DestinyPatch |
|---|---|---|
| Mouse | sim | sim |
| Teclado | sim | sim |
| Hora | sim | sim |
| Arquivos | sim | sim |

| | Power Patch | DestinyPatch |
|---|---|---|
| Executar programas | sim | não |
| Internet | não | sim (apenas TCP/IP) |

### Acesso ao RPG_RT

| | Power Patch | DestinyPatch |
|---|---|---|
| Variáveis/Switches | sim | sim |
| Eventos | não | sim |
| Comandos de evento | não | sim |
| Mapas | não | sim |
| Panorama | não | sim |
| Pictures | parcial (com os plugins Propertymaps e DrawText) | sim |
| Banco de dados do jogo | parcial (apenas heróis, com o plugin HeroAccess) | parcial (heróis, itens, técnicas, atributos, condições e CommonEvents) |
| Jogo (salvar/carregar) | parcial (apenas teclas simuladas) | sim |

### Versões do RPG Maker

| | Power Patch | DestinyPatch |
|---|---|---|
| RPG Maker 2000 | todas | parcial (só 1.0.7) |
| RPG Maker 2003 | todas | nenhuma |

## Tastenpatch do Ineluki

### Geral

O Tastenpatch do Ineluki (site desconhecido) foi o primeiro patch popular/relevante para o RPG Maker, já que ele foi o primeiro a permitir ampliar o conjunto de funções do RPG Maker. Como o nome sugere, ele permite acessar todas as teclas do teclado. Além disso, também é possível acessar o mouse.

### Script

O script é feito através de arquivos WAV preparados, que depois são interpretados. A execução de um desses arquivos WAV acontece por comandos de evento, portanto o processamento é serial (ou seja, o jogo só continua depois que o script terminar de rodar). Porém, não se trata de uma linguagem de script de verdade, já que o "script" aqui funciona por meio de arquivos INI, que podem se referenciar uns aos outros.

### Comparação com o DestinyPatch 2.0

#### Script

| | Tastenpatch | DestinyPatch |
|---|---|---|
| Processamento do código de script | serial | serial (em relação aos eventos) |
| Ramificações | não | sim |
| Laços | não | sim |
| Funções próprias | não | não |
| Classes próprias | não | não |

| | Tastenpatch | DestinyPatch |
|---|---|---|
| Incluir scripts | sim | não |

### Funções disponíveis

| | Tastenpatch | DestinyPatch |
|---|---|---|
| Matemática | não | sim |
| Strings | não | sim |
| Operações lógicas | não | sim |
| Operações binárias | não | sim |

### Acesso ao sistema

| | Tastenpatch | DestinyPatch |
|---|---|---|
| Mouse | sim | sim |
| Teclado | sim | sim |
| Hora | não | sim |
| Arquivos | sim (apenas mensagens de log) | sim |
| Executar programas | sim | não |
| Internet | não | sim (apenas TCP/IP) |

### Acesso ao RPG_RT

| | Tastenpatch | DestinyPatch |
|---|---|---|
| Variáveis/Switches | parcial (só gravação de variáveis) | sim |
| Eventos | não | sim |
| Comandos de evento | não | sim |
| Mapas | não | sim |
| Panorama | não | sim |
| Pictures | não | sim |
| Banco de dados do jogo | não | parcial (heróis, itens, técnicas, atributos, condições e CommonEvents) |
| Jogo (salvar/carregar) | não | sim |

### Versões do RPG Maker

| | Tastenpatch | DestinyPatch |
|---|---|---|
| RPG Maker 2000 | todas (o valor do RPG_RT precisa ser repatchado) | parcial (só 1.0.7) |
| RPG Maker 2003 | todas (algumas só com o repatch do RPG_RT) | nenhuma |

## Disharmony

### Geral

Este patch foi criado por derula (http://www.uglyhorst.de/) e melhora a reprodução de música do RPG Maker. Esse patch não contém, ele próprio, uma linguagem de script, mas é compatível com o Tastenpatch do Ineluki. O Disharmony traz sua própria engine de reprodução de música e é usado como substituto da `Harmony.dll` do RPG

Maker. Essa engine suporta uma grande variedade de formatos (entre outros, MP3 e OGG). Todas as particularidades dos arquivos MIDI (como, por exemplo, comandos MIDI próprios do RPG Maker para definir loops) também foram consideradas no novo desenvolvimento. Além disso, com o Disharmony a reprodução de MIDIs tem menos erros: por exemplo, um MIDI pode alterar o "pitch" dos instrumentos na reprodução. No Disharmony, essa configuração é corretamente reiniciada ao trocar de MIDI; já na `Harmony.dll` da Enterbrain isso não acontece (por isso, às vezes uma música de fundo tocada depois de outra música de fundo soa errada).

GUIA

# Cálculo de chips inferiores e superiores (Lower/Upper)

O chipset pode ser dividido, em linhas gerais, em 6 blocos:

- **Bloco A:** água normal animada
- **Bloco B:** água profunda animada
- **Bloco C:** campos animados com 4 frames (por exemplo, uma cachoeira)
- **Bloco D:** campos cujas bordas se adaptam ao uso (por exemplo, grama alta, deserto com borda de grama...)
- **Bloco E:** campos individuais da área do Lower Chip
- **Bloco F:** campos individuais da área do Upper Chip

## Bloco A

A descrição deste bloco ainda não está disponível.

## Bloco B

A descrição deste bloco ainda não está disponível.

## Bloco C

Aqui os valores usados podem ser lidos simplesmente na tabela do gráfico.

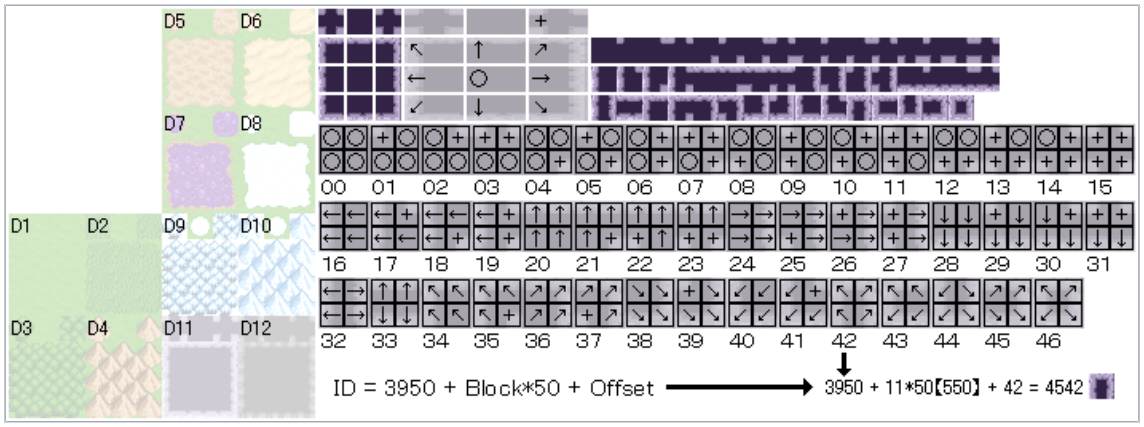
*Bloco C do chipset*

## Bloco D

Aqui um campo é calculado pela seguinte fórmula:

```
ID = 3950 + Bloco * 50 + Offset
```

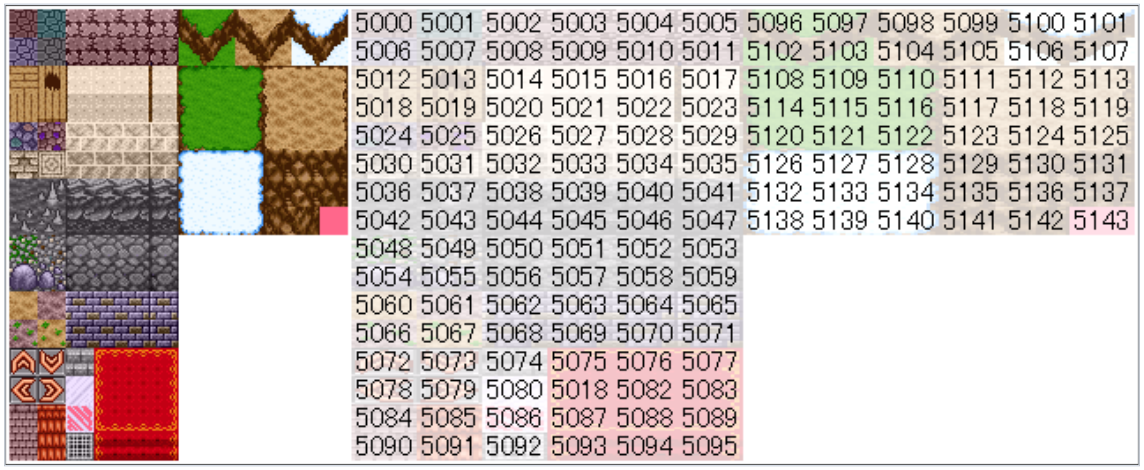
Sendo que **Bloco** está na faixa de 1 a 12 e **Offset** está, conforme o gráfico, na faixa de 0 a 46.




*Bloco D do chipset*

## Bloco E

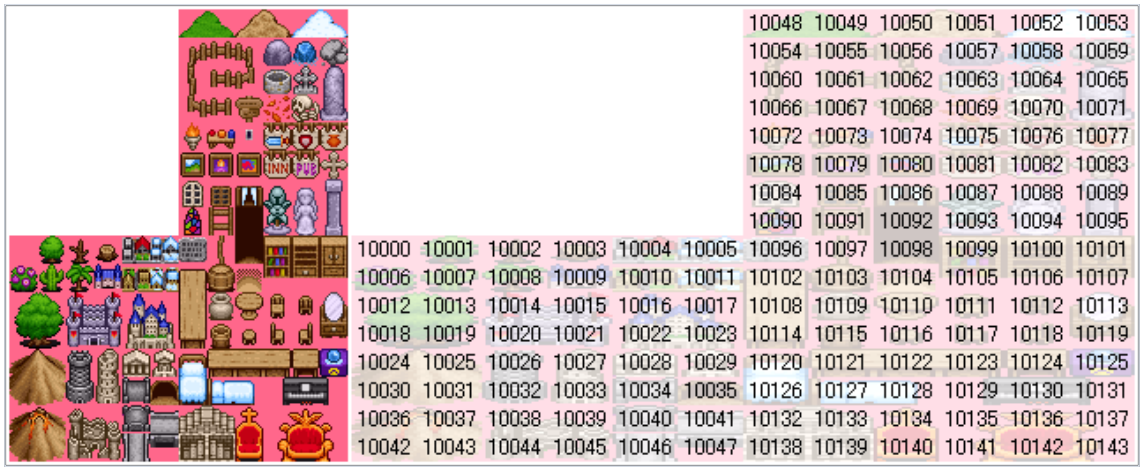
Aqui os valores usados podem ser lidos simplesmente na tabela do gráfico.




*Bloco E do chipset*

## Bloco F

Aqui os valores usados podem ser lidos simplesmente na tabela do gráfico.




*Bloco F do chipset*

# Palavras finais

## Palavras finais da versão 1.0

Para o grande final desta ajuda, eu queria acrescentar uma resenha sobre o DestinyPatch e o seu desenvolvimento. A ideia de escrever um patch para o RPG Maker 2000 eu já tinha havia alguns anos, mas na época ainda me faltava o conhecimento necessário para desenvolver algo consistente. Eu tinha tentado uma solução com um segundo arquivo EXE, que lesse e editasse os valores de memória do EXE verdadeiro. Essa solução trouxe, claro, problemas consideráveis, não só de desempenho, mas também por causa dos diferentes endereços de memória das várias versões do `RPG_RT.exe`. Além disso, outro problema seria que o processo de monitoramento teria rodado de certa forma em paralelo, e uma integração assim certamente traria problemas em alguns sistemas.

Na época, porém, eu já tinha desenvolvido a ideia de que um patch capaz de ler o conteúdo do comando de comentário seria o ideal. Isso teria a vantagem de que todos os comandos que o patch deveria executar seriam digitados no próprio RPG Maker, e também a vantagem de que a execução de um evento seria interrompida durante a execução de um comando do patch.

Três anos depois (fim de 2005 — nessa época eu já tinha bem mais conhecimento sobre arquivos EXE, além de ter aprendido a linguagem Assembly), retomei minha velha ideia e comecei a desenvolver o DestinyPatch. Para isso, eu desmontei (disassemblei) o `RPG_RT.exe` e depois o modifiquei manualmente (com a ajuda de um editor hexadecimal). O patch daquela época funcionava, mas era incrivelmente modesto, já que, por exemplo, não havia um interpretador de fórmulas de verdade. Algumas funções, como desativar a tecla F12, também me deram bastante trabalho. E como o patch era mais um incômodo do que uma ajuda, acabei engavetando o projeto na época.

No início de 2007, tive novamente a oportunidade de modificar um jogo para um amigo (mais vida, etc.) — e aí peguei gosto pela coisa. Por isso, retomei o projeto na época e o reescrevi por completo. Mas, dessa vez, fui com estrutura! Por exemplo, desenhei vários fluxogramas para o interpretador de fórmulas, o que simplificou bastante o trabalho. Além disso, problemas como o da tecla F12 se resolveram logo na primeira tentativa. Às vezes é mesmo útil dar uma (curta) pausa. Ao longo de todo o ano de 2007, trabalhei continuamente no patch (ou seja, pelo menos 1 hora por semana), até que ele finalmente ficou pronto no fim de 2007.

A parte mais difícil do projeto, porém, foi escrever este arquivo de ajuda aqui, já que escrever textos linguísticos (e, sobretudo, compreensíveis) é bem mais difícil do que desenvolver código que, por exemplo, acessa e gerencia a pilha de seus procedimentos-mãe. Mas documentar o patch também trouxe vantagens, já que, enquanto escrevia, percebi algumas funções sem sentido, que foram removidas, além de algumas funções úteis que ainda nem tinham sido incluídas (por exemplo, o método `If`). Em retrospecto, também percebi algumas funções desnecessárias que eu tinha incluído no primeiro período do projeto (2005), mas que mesmo assim deixei, já que, em certas circunstâncias, poderiam ser minimamente úteis (afinal, quem precisa de uma arco-secante?!).

Agora (início de 2008) o patch está testado e, por ora, pronto. Se novas versões vão sair, isso ainda é incerto, já que também depende da resposta dos usuários — se funções adicionais são desejadas, ou se novas versões são necessárias. Além disso, muita gente provavelmente desejaria novas funções tão difíceis de implementar que escrever um novo Maker seria mais fácil (e, de qualquer forma, mais sensato).

De qualquer forma, agora desejo a todas as pessoas que (apesar da já ultrapassada versão do RPG Maker 2000) queiram usar o patch, muita diversão com ele!

## Palavras finais da versão 2.0

Com o lançamento do DestinyPatch 1.0, passei a ter mais contato com outros criadores de patches. Por um lado, com Cherry, que desenvolve o Hyper-Patch, e por outro, com derula, que desenvolve o Disharmony. No período seguinte, mantive contato frequente com os dois, e também trocamos informações entre nós.

Em meados/fim de 2008, comecei o desenvolvimento da versão sucessora do DestinyPatch 1.0. No início, a versão sucessora ainda tinha o número de versão 1.1 e deveria conter apenas algumas pequenas alterações. Mas, como sempre acontece, as exigências foram crescendo, de modo que decidi reescrever o sistema central do zero e portar a funcionalidade já existente para ele.

Por isso, decidi recomeçar os dois projetos "DestinyScript" e "DestinyPatcher". O início foi com o "DestinyScript" ( `Destiny.dll`), que escrevi novamente em Assembly. A principal mudança seria tornar a linguagem de script simplificada mais poderosa, ou seja, com suporte a elementos de estrutura como if, do, for... e também subobjetos para estruturação. Com um pouco de raciocínio e alguns fluxogramas, os algoritmos necessários para isso também puderam ser desenhados rapidamente. Mais ou menos nessa época, meu curso técnico terminou. Por causa da crise econômica, não fui efetivado, e por isso fiquei alguns meses desempregado. Nesse período, usei meu tempo livre forçado para portar muitos dos objetos já existentes para o DestinyScript e adicionar outros novos.

Mais tarde, passei do projeto ativo "DestinyScript" para o "DestinyPatcher", para reescrevê-lo. Para isso, usei novamente C++ (MinGW), mas desta vez com o Code::Blocks em vez do já extinto Dev-C++. No processo de reformulação, decidi reformular completamente o conceito de diálogos. Assim, o programa deixou de ser baseado em Frame (`CreateWindowEx`) e passou a ser baseado em Dialog (`CreateDialog`). Essa abordagem tornou mais difícil o trabalho com precisão de pixel, mas simplificou outras chamadas (por exemplo, agora é possível usar a tecla Tab sem problemas para alternar entre os controles). Além disso, decidi não compilar mais a linguagem de script DFI (os arquivos da pasta Identify do patcher, que dizem ao patcher como editar um arquivo EXE) em código binário. Isso agora acontece automaticamente ao iniciar o DestinyPatcher.

No fim de 2009, decidi reescrever o projeto "DestinyScript" em C++. Alguns dos conceitos já me pareciam ultrapassados, além de o Assembly às vezes exigir bastante manutenção. No início de 2010, percebi que a versão em C++ exigiria muito trabalho, já que o código já implementado em Assembly era, em quantidade, muito grande. Por isso, cheguei a pensar em deixar "morrer" o DestinyScript (a linguagem; não os objetos) e fiz uma enquete democrática no fórum com algumas alternativas. A maioria das pessoas votou na linguagem de script Lua, que também é usada no Hyper-Patch do Cherry. Depois disso, estudei o Lua com mais atenção e cheguei à conclusão de que simplesmente não gosto dessa linguagem, ou melhor, que alguns conceitos dessa linguagem simplesmente não se encaixam com o sistema já existente. Por isso, agi de forma "democrática" e decidi simplesmente ignorar o resultado da votação. Depois de pensar mais, cheguei à conclusão de que o código já existente em Assembly era, em quantidade, tanto que uma reformulação seria provavelmente o mais simples (e o mais rápido).

Nessa época, comecei a trabalhar na documentação do DestinyPatch 2.0. Diferente da documentação anterior, eu queria gerar a documentação a partir de arquivos XML. Na versão anterior, eu tinha criado os documentos HTML manualmente, o que também gerava erros em alguns casos. A criação da documentação acabou crescendo tanto que virou um projeto próprio.

Ao documentar, porém, percebi que ainda havia alguns pontos do DestinyScript e do DestinyPatcher que precisavam ser revisados. Por isso, comecei a montar uma lista de todos os pontos ainda em aberto; essa lista era/é consultável como "plano de marcos" (Meilensteinplan) dentro da página do projeto do DestinyPatch. A vantagem dessa abordagem foi, por um lado, uma estrutura melhor, e por outro, envolver as partes ainda ativas (e interessadas) da comunidade do RPG Maker. Com base nessa lista, comecei a melhorar, estruturar e corrigir erros encontrados nos projetos DestinyScript e DestinyPatcher de forma considerável. Assim nasceu também o pequeno e bem delimitado subprojeto do pré-processador, que, ao compilar a `Destiny.dll`, disponibiliza as propriedades/métodos criados também dentro da linguagem de script.

Depois que todos os pontos em aberto dos projetos DestinyScript e DestinyPatcher foram resolvidos, "congelei" essas partes no plano de marcos e quis apenas finalizar a documentação. Essa última grande fase da documentação se estendeu como um chiclete. Essa fase durou cerca de um ano e meio e envolveu, essencialmente, a documentação do patch. Porém, ao documentar e testar, eu sempre descobria novos erros ou funções importantes que faltavam. Com isso, corrigia os erros e completava as funções faltantes, o que custava mais ou menos tempo adicional.

Agora, porém — cerca de 4 anos depois do lançamento do DestinyPatch 1.0 — posso finalmente apresentar o DestinyPatch 2.0. Alguns conceitos podem estar ultrapassados, mas pelo menos o lançamento, depois de quatro

anos, está finalmente pronto.

O que eu espero é não ter demorado demais, afinal quatro anos é muito tempo. A maioria das pessoas que usava o DestinyPatch na época já parou de usar o RPG Maker. Mas, para as pessoas restantes e para as que chegaram agora, desejo muita diversão e sucesso com a nova versão do patch!

# Informações legais (Impressum)

O DestinyPatch inteiro (DestinyScript e DestinyPatcher), bem como este manual, foi criado por David Gausmann (conhecido como **Bananen-Joe**). Meus agradecimentos também a MagicMaker, que enviou sugestões de melhoria e gráficos, que também foram incorporados a esta ajuda.

Em caso de dúvidas (que não puderam ser respondidas por este manual), comentários e críticas, você pode me contatar pelo e-mail: `destinydll[at]bananen-joe[ponto]de`

E dê uma olhada na minha página: http://www.bananen-joe.de/

Divirta-se — seu Bananen-Joe.

> **Observação:** esta é a tradução para o português do manual alemão; o texto acima reproduz o aviso de autoria do original.