

Bee Game

Final

Generated by Doxygen 1.8.13

Contents

1 Namespace Index	2
1.1 Packages	2
2 Hierarchical Index	2
2.1 Class Hierarchy	2
3 Class Index	5
3.1 Class List	5
4 File Index	8
4.1 File List	8
5 Namespace Documentation	11
5.1 BeeGame Namespace Reference	11
5.2 BeeGame.Blocks Namespace Reference	12
5.3 BeeGame.Core Namespace Reference	12
5.4 BeeGame.Core.Dictionaries Namespace Reference	13
5.5 BeeGame.Core.Enums Namespace Reference	13
5.5.1 Enumeration Type Documentation	14
5.6 BeeGame.Core.UnityTypeReplacements Namespace Reference	18
5.7 BeeGame.Exceptions Namespace Reference	18
5.8 BeeGame.Inventory Namespace Reference	19
5.9 BeeGame.Inventory.BlockInventory Namespace Reference	19
5.10 BeeGame.Inventory.Player_Inventory Namespace Reference	19
5.11 BeeGame.Items Namespace Reference	20
5.12 BeeGame.Player Namespace Reference	20
5.13 BeeGame.Quest Namespace Reference	20
5.14 BeeGame.Resources Namespace Reference	20
5.15 BeeGame.Serialization Namespace Reference	21
5.16 BeeGame.Terrain Namespace Reference	21
5.17 BeeGame.Terrain.Chunks Namespace Reference	21
5.18 BeeGame.Terrain.LandGeneration Namespace Reference	21
5.19 BeeGame.Terrain.LandGeneration.Noise Namespace Reference	22

6 Class Documentation	22
6.1 BeeGame.Items.AbstractItem Class Reference	22
6.1.1 Detailed Description	22
6.1.2 Member Function Documentation	22
6.2 BeeGame.Blocks.Air Class Reference	23
6.2.1 Detailed Description	24
6.2.2 Constructor & Destructor Documentation	24
6.2.3 Member Function Documentation	24
6.2.4 Member Data Documentation	27
6.3 BeeGame.Blocks.Apiary Class Reference	27
6.3.1 Detailed Description	28
6.3.2 Constructor & Destructor Documentation	28
6.3.3 Member Function Documentation	29
6.3.4 Member Data Documentation	38
6.4 BeeGame.Inventory.ApiaryCraftingOutputSlot Class Reference	39
6.4.1 Detailed Description	39
6.4.2 Member Function Documentation	39
6.5 BeeGame.Inventory.ApiaryInventory Class Reference	40
6.5.1 Detailed Description	41
6.5.2 Member Function Documentation	41
6.5.3 Member Data Documentation	43
6.6 BeeGame.Items.ApplyColour Class Reference	44
6.6.1 Detailed Description	45
6.6.2 Member Function Documentation	45
6.6.3 Member Data Documentation	45
6.7 BeeGame.Blocks.Bedrock Class Reference	46
6.7.1 Detailed Description	46
6.7.2 Constructor & Destructor Documentation	46
6.7.3 Member Function Documentation	47
6.7.4 Member Data Documentation	48

6.8 BeeGame.Items.Bee Class Reference	49
6.8.1 Detailed Description	50
6.8.2 Constructor & Destructor Documentation	50
6.8.3 Member Function Documentation	51
6.8.4 Member Data Documentation	55
6.8.5 Property Documentation	56
6.9 BeeGame.Items.BeeAlyzer Class Reference	57
6.9.1 Detailed Description	58
6.9.2 Constructor & Destructor Documentation	58
6.9.3 Member Function Documentation	59
6.9.4 Member Data Documentation	61
6.9.5 Property Documentation	61
6.10 BeeGame.Inventory.BeeAlyzerInventory Class Reference	62
6.10.1 Detailed Description	63
6.10.2 Member Function Documentation	63
6.10.3 Member Data Documentation	66
6.11 BeeGame.Core.Dictionaries.BeeCombinationDictionaryEqualityComparer Class Reference	67
6.11.1 Detailed Description	67
6.11.2 Member Function Documentation	68
6.12 BeeGame.Core.Dictionaries.BeeDictionaries Class Reference	69
6.12.1 Detailed Description	69
6.12.2 Member Function Documentation	69
6.12.3 Member Data Documentation	72
6.13 BeeGame.Blocks.Block Class Reference	74
6.13.1 Detailed Description	75
6.13.2 Constructor & Destructor Documentation	75
6.13.3 Member Function Documentation	76
6.13.4 Member Data Documentation	80
6.14 BeeGame.Blocks.Chest Class Reference	81
6.14.1 Detailed Description	82

6.14.2 Constructor & Destructor Documentation	82
6.14.3 Member Function Documentation	82
6.14.4 Member Data Documentation	86
6.15 BeeGame.Inventory.ChestInventory Class Reference	86
6.15.1 Detailed Description	87
6.15.2 Member Function Documentation	87
6.15.3 Member Data Documentation	90
6.16 BeeGame.Terrain.Chunks.Chunk Class Reference	91
6.16.1 Detailed Description	92
6.16.2 Member Function Documentation	92
6.16.3 Member Data Documentation	97
6.17 BeeGame.Terrain.ChunkWorldPos Struct Reference	99
6.17.1 Detailed Description	99
6.17.2 Constructor & Destructor Documentation	99
6.17.3 Member Function Documentation	100
6.17.4 Member Data Documentation	102
6.18 BeeGame.Inventory.CraftingOutputSlot Class Reference	102
6.18.1 Detailed Description	103
6.18.2 Member Function Documentation	103
6.19 BeeGame.Exceptions.CraftingRecipeAdditionException Class Reference	104
6.19.1 Detailed Description	104
6.19.2 Constructor & Destructor Documentation	104
6.20 BeeGame.Core.Dictionaries.CraftingRecipies Class Reference	105
6.20.1 Detailed Description	105
6.20.2 Member Function Documentation	106
6.20.3 Member Data Documentation	111
6.21 BeeGame.Blocks.CraftingTable Class Reference	112
6.21.1 Detailed Description	113
6.21.2 Constructor & Destructor Documentation	113
6.21.3 Member Function Documentation	113

6.21.4 Member Data Documentation	118
6.22 BeeGame.Inventory.BlockInventory.CraftingTableInventory Class Reference	119
6.22.1 Detailed Description	120
6.22.2 Member Function Documentation	120
6.22.3 Member Data Documentation	124
6.23 BeeGame.Blocks.Dirt Class Reference	124
6.23.1 Detailed Description	125
6.23.2 Constructor & Destructor Documentation	125
6.23.3 Member Function Documentation	125
6.23.4 Member Data Documentation	127
6.24 BeeGame.Core.Extensions Class Reference	127
6.24.1 Detailed Description	128
6.24.2 Member Function Documentation	128
6.25 BeeGame.Blocks.Grass Class Reference	130
6.25.1 Detailed Description	131
6.25.2 Constructor & Destructor Documentation	131
6.25.3 Member Function Documentation	132
6.25.4 Member Data Documentation	134
6.26 BeeGame.Items.Honey Class Reference	134
6.26.1 Detailed Description	135
6.26.2 Member Function Documentation	135
6.26.3 Member Data Documentation	136
6.27 BeeGame.Items.HoneyComb Class Reference	136
6.27.1 Detailed Description	137
6.27.2 Constructor & Destructor Documentation	137
6.27.3 Member Function Documentation	138
6.27.4 Member Data Documentation	139
6.27.5 Property Documentation	140
6.28 BeeGame.Exceptions.InputException Class Reference	140
6.28.1 Detailed Description	141

6.28.2 Constructor & Destructor Documentation	141
6.29 BeeGame.Inventory.Inventory Class Reference	142
6.29.1 Detailed Description	143
6.29.2 Member Function Documentation	143
6.29.3 Member Data Documentation	147
6.30 BeeGame.Inventory.InventorySlot Class Reference	149
6.30.1 Detailed Description	150
6.30.2 Member Function Documentation	150
6.30.3 Member Data Documentation	156
6.31 BeeGame.Items.Item Class Reference	158
6.31.1 Detailed Description	160
6.31.2 Constructor & Destructor Documentation	160
6.31.3 Member Function Documentation	160
6.31.4 Member Data Documentation	172
6.31.5 Property Documentation	172
6.32 BeeGame.Items.ItemGameObject Class Reference	173
6.32.1 Detailed Description	174
6.32.2 Member Function Documentation	174
6.32.3 Member Data Documentation	175
6.33 BeeGame.Inventory.ItemsInInventory Class Reference	175
6.33.1 Detailed Description	176
6.33.2 Constructor & Destructor Documentation	176
6.33.3 Member Function Documentation	176
6.33.4 Member Data Documentation	177
6.34 BeeGame.Blocks.Leaves Class Reference	178
6.34.1 Detailed Description	178
6.34.2 Constructor & Destructor Documentation	178
6.34.3 Member Function Documentation	179
6.34.4 Member Data Documentation	181
6.35 BeeGame.Terrain.Chunks.LoadChunks Class Reference	181

6.35.1 Detailed Description	182
6.35.2 Member Function Documentation	182
6.35.3 Member Data Documentation	185
6.36 BeeGame.LoadResources Class Reference	186
6.36.1 Detailed Description	187
6.36.2 Member Function Documentation	187
6.36.3 Member Data Documentation	188
6.37 BeeGame.Terrain.Chunks.MeshData Class Reference	188
6.37.1 Detailed Description	188
6.37.2 Member Function Documentation	189
6.37.3 Member Data Documentation	191
6.38 BeeGame.Items.NormalBee Class Reference	192
6.38.1 Detailed Description	193
6.38.2 Member Function Documentation	193
6.38.3 Member Data Documentation	193
6.39 BeeGame.Blocks.Planks Class Reference	195
6.39.1 Detailed Description	196
6.39.2 Constructor & Destructor Documentation	196
6.39.3 Member Function Documentation	196
6.39.4 Member Data Documentation	198
6.40 BeeGame.Inventory.Player_Inventory.PlayerInventory Class Reference	198
6.40.1 Detailed Description	199
6.40.2 Member Function Documentation	199
6.40.3 Member Data Documentation	203
6.41 BeeGame.Player.PlayerLook Class Reference	204
6.41.1 Detailed Description	204
6.41.2 Member Function Documentation	204
6.41.3 Member Data Documentation	205
6.42 BeeGame.Player.PlayerMove Class Reference	207
6.42.1 Detailed Description	208

6.42.2 Member Function Documentation	208
6.42.3 Member Data Documentation	210
6.43 BeeGame.Core.Dictionaries.PrefabDictionary Class Reference	211
6.43.1 Detailed Description	211
6.43.2 Member Function Documentation	211
6.43.3 Member Data Documentation	212
6.44 BeeGame.Items.QueenBee Class Reference	212
6.44.1 Detailed Description	213
6.44.2 Constructor & Destructor Documentation	213
6.44.3 Member Function Documentation	213
6.44.4 Property Documentation	214
6.45 BeeGame.Exceptions.QuestAlreadyExistsException Class Reference	214
6.45.1 Detailed Description	214
6.45.2 Constructor & Destructor Documentation	214
6.46 BeeGame.Items.QuestBook Class Reference	215
6.46.1 Detailed Description	216
6.46.2 Constructor & Destructor Documentation	216
6.46.3 Member Function Documentation	216
6.46.4 Member Data Documentation	218
6.46.5 Property Documentation	218
6.47 BeeGame.Inventory.QuestBookInventory Class Reference	218
6.47.1 Detailed Description	219
6.47.2 Member Function Documentation	219
6.47.3 Member Data Documentation	222
6.48 BeeGame.Quest.QuestEvents Class Reference	222
6.48.1 Detailed Description	223
6.48.2 Member Function Documentation	223
6.48.3 Event Documentation	224
6.49 BeeGame.Quest.Quests Class Reference	225
6.49.1 Detailed Description	226

6.49.2 Member Function Documentation	226
6.49.3 Member Data Documentation	229
6.50 BeeGame.Resources.Resources Class Reference	231
6.50.1 Detailed Description	231
6.50.2 Constructor & Destructor Documentation	231
6.50.3 Member Function Documentation	232
6.50.4 Member Data Documentation	233
6.50.5 Property Documentation	234
6.51 BeeGame.Terrain.Chunks.SaveChunk Class Reference	234
6.51.1 Detailed Description	235
6.51.2 Constructor & Destructor Documentation	235
6.51.3 Member Data Documentation	235
6.52 BeeGame.Player.SavePlayerPosition Class Reference	236
6.52.1 Detailed Description	236
6.52.2 Member Function Documentation	236
6.52.3 Member Data Documentation	237
6.53 BeeGame.Player.Selector Class Reference	237
6.53.1 Detailed Description	238
6.53.2 Member Function Documentation	238
6.53.3 Member Data Documentation	241
6.54 BeeGame.Serialization.Serialization Class Reference	242
6.54.1 Detailed Description	243
6.54.2 Member Function Documentation	243
6.54.3 Member Data Documentation	249
6.55 BeeGame.Items.SetBeeGOColours Class Reference	250
6.55.1 Detailed Description	251
6.55.2 Member Function Documentation	251
6.55.3 Member Data Documentation	251
6.56 BeeGame.Terrain.LandGeneration.Noise.SimplexNoise Class Reference	252
6.56.1 Detailed Description	253

6.56.2 Member Function Documentation	253
6.56.3 Member Data Documentation	258
6.57 BeeGame.SpawnItem Class Reference	259
6.57.1 Detailed Description	259
6.57.2 Member Function Documentation	259
6.58 BeeGame.Core.Dictionaries.SpriteDictionary Class Reference	260
6.58.1 Detailed Description	261
6.58.2 Member Function Documentation	261
6.58.3 Member Data Documentation	262
6.59 BeeGame.Terrain.LandGeneration.Terrain Class Reference	262
6.59.1 Detailed Description	263
6.59.2 Member Function Documentation	263
6.59.3 Member Data Documentation	270
6.60 BeeGame.Terrain.LandGeneration.TerrainGeneration Class Reference	270
6.60.1 Detailed Description	271
6.60.2 Member Function Documentation	272
6.60.3 Member Data Documentation	277
6.61 BeeGame.Test Class Reference	279
6.61.1 Detailed Description	279
6.61.2 Member Function Documentation	280
6.62 BeeGame.Core.THInput Class Reference	280
6.62.1 Detailed Description	281
6.62.2 Member Function Documentation	281
6.62.3 Member Data Documentation	284
6.63 BeeGame.Core.UnityTypeReplacements.THQuaternion Struct Reference	285
6.63.1 Detailed Description	286
6.63.2 Constructor & Destructor Documentation	286
6.63.3 Member Function Documentation	287
6.63.4 Member Data Documentation	290
6.64 BeeGame.Core.THVector2 Struct Reference	291

6.64.1 Detailed Description	292
6.64.2 Constructor & Destructor Documentation	292
6.64.3 Member Function Documentation	293
6.64.4 Member Data Documentation	298
6.65 BeeGame.Core.THVector3 Struct Reference	298
6.65.1 Detailed Description	300
6.65.2 Constructor & Destructor Documentation	300
6.65.3 Member Function Documentation	301
6.65.4 Member Data Documentation	311
6.66 BeeGame.Items.Tile Struct Reference	312
6.66.1 Detailed Description	312
6.66.2 Member Data Documentation	312
6.67 BeeGame.Blocks.Wood Class Reference	313
6.67.1 Detailed Description	313
6.67.2 Constructor & Destructor Documentation	314
6.67.3 Member Function Documentation	314
6.67.4 Member Data Documentation	315
6.68 BeeGame.Terrain.LandGeneration.World Class Reference	316
6.68.1 Detailed Description	316
6.68.2 Member Function Documentation	316
6.68.3 Member Data Documentation	321

7 File Documentation	321
7.1 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Air.cs File Reference	321
7.2 Air.cs	322
7.3 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Apiary.cs File Reference	322
7.4 Apiary.cs	322
7.5 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Bedrock.cs File Reference	326
7.6 Bedrock.cs	326
7.7 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Block.cs File Reference	327
7.8 Block.cs	327
7.9 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Chest.cs File Reference	328
7.10 Chest.cs	329
7.11 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Crafting← Table.cs File Reference	330
7.12 CraftingTable.cs	330
7.13 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Dirt.cs File Reference	331
7.14 Dirt.cs	332
7.15 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Grass.cs File Reference	332
7.16 Grass.cs	333
7.17 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Leaves.cs File Reference	333
7.18 Leaves.cs	334
7.19 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Planks.cs File Reference	334
7.20 Planks.cs	335
7.21 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Wood.cs File Reference	335
7.22 Wood.cs	335
7.23 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys← BeeDictionaries.cs File Reference	336

7.24 BeeDictionaries.cs	336
7.25 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/← CraftingRecipies.cs File Reference	338
7.26 CraftingRecipies.cs	338
7.27 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/← EqualityComperors.cs File Reference	340
7.28 EqualityComperors.cs	340
7.29 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/← PrefabDictionary.cs File Reference	341
7.30 PrefabDictionary.cs	341
7.31 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/← SpriteDictionary.cs File Reference	341
7.32 SpriteDictionary.cs	342
7.33 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Enums/← Enums.cs File Reference	342
7.34 Enums.cs	343
7.35 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Extensions.cs File Reference	344
7.36 Extensions.cs	344
7.37 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityType← Replacements/THInput.cs File Reference	346
7.38 THInput.cs	346
7.39 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityType← Replacements/THQuaternion.cs File Reference	348
7.40 THQuaternion.cs	348
7.41 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityType← Replacements/THVector2.cs File Reference	350
7.42 THVector2.cs	350
7.43 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityType← Replacements/THVector3.cs File Reference	352
7.44 THVector3.cs	352
7.45 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Exceptions/← CraftingRecipeAdditionException.cs File Reference	354
7.46 CraftingRecipeAdditionException.cs	355
7.47 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Exceptions/Input← Exception.cs File Reference	355

7.48 InputException.cs	356
7.49 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Exceptions/Quest← AlreadyExistsException.cs File Reference	356
7.50 QuestAlreadyExistsException.cs	356
7.51 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Apiary← CraftingOutputSlot.cs File Reference	357
7.52 ApiaryCraftingOutputSlot.cs	357
7.53 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Block← Inventory/ApiaryInventory.cs File Reference	357
7.54 ApiaryInventory.cs	358
7.55 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Block← Inventory/ChestInventory.cs File Reference	359
7.56 ChestInventory.cs	359
7.57 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Block← Inventory/CraftingTableInventory.cs File Reference	360
7.58 CraftingTableInventory.cs	361
7.59 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Crafting← OutputSlot.cs File Reference	362
7.60 CraftingOutputSlot.cs	363
7.61 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Inventory.cs File Reference	363
7.62 Inventory.cs	363
7.63 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Inventory← Slot.cs File Reference	365
7.64 InventorySlot.cs	365
7.65 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Item← Inventory/BeeAlyzerInventory.cs File Reference	368
7.66 BeeAlyzerInventory.cs	369
7.67 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Item← Inventory/QuestBookInventory.cs File Reference	370
7.68 QuestBookInventory.cs	371
7.69 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/ItemsIn← Inventory.cs File Reference	372
7.70 ItemsInInventory.cs	372
7.71 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Player← Inventory/PlayerInventory.cs File Reference	373

7.72 PlayerInventory.cs	373
7.73 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/Abstract← Item.cs File Reference	375
7.74 AbstractItem.cs	375
7.75 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/ApplyColour.cs File Reference	376
7.76 ApplyColour.cs	376
7.77 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/Bee.cs File Reference	376
7.78 Bee.cs	377
7.79 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/BeeAlyzer.cs File Reference	379
7.80 BeeAlyzer.cs	380
7.81 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/GameObject← Stuff/ItemGameObject.cs File Reference	381
7.82 ItemGameObject.cs	381
7.83 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/GameObject← Stuff/SetBeeGOColours.cs File Reference	382
7.84 SetBeeGOColours.cs	382
7.85 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/Honey.cs File Reference	383
7.86 Honey.cs	383
7.87 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/Honey← Comb.cs File Reference	383
7.88 HoneyComb.cs	384
7.89 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/Item.cs File Reference	384
7.90 Item.cs	385
7.91 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/QuestBook.cs File Reference	388
7.92 QuestBook.cs	389
7.93 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/LoadResources.cs File Reference	389
7.94 LoadResources.cs	390
7.95 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/obj/Debug/Temporary← GeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs File Reference	390

7.96 TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs	390
7.97 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/obj/Debug/TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs File Reference	390
7.98 TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs	390
7.99 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/obj/Debug/TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs File Reference	390
7.100TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs	390
7.101C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/PlayerLook.cs File Reference	391
7.102PlayerLook.cs	391
7.103C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/PlayerMove.cs File Reference	391
7.104PlayerMove.cs	392
7.105C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/SavePlayerPosition.cs File Reference	393
7.106SavePlayerPosition.cs	393
7.107C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/Selector.cs File Reference	393
7.108Selector.cs	394
7.109C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Properties/AssemblyInfo.cs File Reference	395
7.110AssemblyInfo.cs	395
7.111C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Quest/QuestEvents.cs File Reference	396
7.112QuestEvents.cs	396
7.113C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Quest/Quests.cs File Reference	397
7.114Quests.cs	397
7.115C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Resources/Resources.Designer.cs File Reference	399
7.116Resources.Designer.cs	399
7.117C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Serialization/Serialization.cs File Reference	401
7.118Serialization.cs	401
7.119C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/SpawnItem.cs File Reference	404

7.120SpawnItem.cs	404
7.121C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/← Chunk.cs File Reference	405
7.122Chunk.cs	405
7.123C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/← LoadChunks.cs File Reference	408
7.124LoadChunks.cs	408
7.125C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/← MeshData.cs File Reference	411
7.126MeshData.cs	412
7.127C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/← SaveChunk.cs File Reference	412
7.128SaveChunk.cs	413
7.129C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunk← WorldPos.cs File Reference	413
7.130ChunkWorldPos.cs	413
7.131C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Land← Generation/Noise/SimplexNoise.cs File Reference	414
7.132SimplexNoise.cs	414
7.133C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Land← Generation/Terrain.cs File Reference	418
7.134Terrain.cs	418
7.135C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Land← Generation/TerrainGeneration.cs File Reference	421
7.136TerrainGeneration.cs	421
7.137C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Land← Generation/World.cs File Reference	423
7.138World.cs	423
7.139C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/test.cs File Reference	426
7.140test.cs	426
Index	427

1 Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

BeeGame	11
BeeGame.Blocks	12
BeeGame.Core	12
BeeGame.Core.Dictionaries	13
BeeGame.Core.Enums	13
BeeGame.Core.UnityTypeReplacements	18
BeeGame.Exceptions	18
BeeGame.Inventory	19
BeeGame.Inventory.BlockInventory	19
BeeGame.Inventory.Player_Inventory	19
BeeGame.Items	20
BeeGame.Player	20
BeeGame.Quest	20
BeeGame.Resources	20
BeeGame.Serialization	21
BeeGame.Terrain	21
BeeGame.Terrain.Chunks	21
BeeGame.Terrain.LandGeneration	21
BeeGame.Terrain.LandGeneration.Noise	22

2 Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

BeeGame.Items.AbstractItem	22
BeeGame.Items.Item	158
BeeGame.Blocks.Block	74
BeeGame.Blocks.Air	23

BeeGame.Blocks.Apiary	27
BeeGame.Blocks.Bedrock	46
BeeGame.Blocks.Chest	81
BeeGame.Blocks.CraftingTable	112
BeeGame.Blocks.Dirt	124
BeeGame.Blocks.Grass	130
BeeGame.Blocks.Leaves	178
BeeGame.Blocks.Planks	195
BeeGame.Blocks.Wood	313
BeeGame.Items.Bee	49
BeeGame.Items.BeeAlyzer	57
BeeGame.Items.QuestBook	215
BeeGame.Items.Honey	134
BeeGame.Items.HoneyComb	136
BeeGame.Core.Dictionaries.BeeDictionaries	69
BeeGame.Terrain.ChunkWorldPos	99
BeeGame.Core.Dictionaries.CraftingRecipies Exception	105
BeeGame.Exceptions.CraftingRecipeAdditionException	104
BeeGame.Exceptions.InputException	140
BeeGame.Exceptions.QuestAlreadyExistsException	214
BeeGame.Core.Extensions ICloneable	127
BeeGame.Items.Item IEqualityComparer	158
BeeGame.Core.Dictionaries.BeeCombinationDictionaryEqualityComparer IPointerClickHandler	67
BeeGame.Inventory.ApiaryCraftingOutputSlot	39
BeeGame.Inventory.CraftingOutputSlot	102
BeeGame.Inventory.InventorySlot	149
BeeGame.Inventory.ApiaryCraftingOutputSlot	39
BeeGame.Inventory.CraftingOutputSlot IPointerEnterHandler	102
BeeGame.Inventory.ApiaryCraftingOutputSlot	39

BeeGame.Inventory.CraftingOutputSlot	102
BeeGame.Inventory.InventorySlot	149
IPointerExitHandler	
BeeGame.Inventory.ApiaryCraftingOutputSlot	39
BeeGame.Inventory.CraftingOutputSlot	102
BeeGame.Inventory.InventorySlot	149
BeeGame.Inventory.ItemsInInventory	175
BeeGame.Terrain.Chunks.MeshData	188
MonoBehaviour	
BeeGame.Inventory.Inventory	142
BeeGame.Inventory.BeeAlyzerInventory	62
BeeGame.Inventory.ChestInventory	86
BeeGame.Inventory.ApiaryInventory	40
BeeGame.Inventory.BlockInventory.CraftingTableInventory	119
BeeGame.Inventory.Player_Inventory.PlayerInventory	198
BeeGame.Inventory.QuestBookInventory	218
BeeGame.Inventory.InventorySlot	149
BeeGame.Items.ApplyColour	44
BeeGame.Items.ItemGameObject	173
BeeGame.Items.SetBeeGOColours	250
BeeGame.LoadResources	186
BeeGame.Player.PlayerLook	204
BeeGame.Player.PlayerMove	207
BeeGame.Player.SavePlayerPosition	236
BeeGame.Player.Selector	237
BeeGame.SpawnItem	259
BeeGame.Terrain.Chunks.Chunk	91
BeeGame.Terrain.Chunks.LoadChunks	181
BeeGame.Terrain.LandGeneration.World	316
BeeGame.Test	279
BeeGame.Items.NormalBee	192
BeeGame.Core.Dictionaries.PrefabDictionary	211
BeeGame.Items.QueenBee	212

BeeGame.Quest.QuestEvents	222
BeeGame.Quest.Quests	225
BeeGame.Resources.Resources	231
BeeGame.Terrain.Chunks.SaveChunk	234
BeeGame.Serialization.Serialization	242
BeeGame.Terrain.LandGeneration.Noise.SimplexNoise	252
BeeGame.Core.Dictionaries.SpriteDictionary	260
BeeGame.Terrain.LandGeneration.Terrain	262
BeeGame.Terrain.LandGeneration.TerrainGeneration	270
BeeGame.Core.THInput	280
BeeGame.Core.UnityTypeReplacements.THQuaternion	285
BeeGame.Core.THVector2	291
BeeGame.Core.THVector3	298
BeeGame.Items.Tile	312

3 Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

BeeGame.Items.AbstractItem Does this need to exist?	22
BeeGame.Blocks.Air Air Block is an empty block that does not render and has no collider	23
BeeGame.Blocks.Apiary Apiary Block	27
BeeGame.Inventory.ApiaryCraftingOutputSlot Overrides the	39
BeeGame.Inventory.ApiaryInventory Inventory for Apiarys Apiary	40
BeeGame.Items.ApplyColour Applies a given colour to a gameobject	44
BeeGame.Blocks.Bedrock Bedrock Block	46
BeeGame.Items.Bee The bee item	49

BeeGame.Items.BeeAlyzer	57
BeeGame.Inventory.BeeAlyzerInventory	
Incentory for the chests	62
BeeGame.Core.Dictionaries.BeeCombinationDictionaryEqualityComparer	67
BeeGame.Core.Dictionaries.BeeDictionaries	69
BeeGame.Blocks.Block	
Base class for blocks	74
BeeGame.Blocks.Chest	
Chest Block	81
BeeGame.Inventory.ChestInventory	
Incentory for the chests	86
BeeGame.Terrain.Chunks.Chunk	
A section of land for the game, used so that land can be generated in parts and not all at once	91
BeeGame.Terrain.ChunkWorldPos	
Serializable int version of THVector3	99
BeeGame.Inventory.CraftingOutputSlot	
Overrides the	102
BeeGame.Exceptions.CraftingRecipeAdditionException	104
BeeGame.Core.Dictionaries.CraftingRecipies	105
BeeGame.Blocks.CraftingTable	
The Workbench Block class	112
BeeGame.Inventory.BlockInventory.CraftingTableInventory	
Invnetory for the CraftingTable Block	119
BeeGame.Blocks.Dirt	
Dirt Block	124
BeeGame.Core.Extensions	127
BeeGame.Blocks.Grass	
Grass Block	130
BeeGame.Items.Honey	134
BeeGame.Items.HoneyComb	
Honey comb item produced by bees	136
BeeGame.Exceptions.InputException	140
BeeGame.Inventory.Inventory	
Base class for all inventorys in the game	142
BeeGame.Inventory.InventorySlot	149
BeeGame.Items.Item	
Base class for all Items and Blocks in the game	158
BeeGame.Items.ItemGameObject	
Interface between item and unity gameobjects	173

BeeGame.Inventory.ItemsInInventory	Class that holds all of the items in the inventory. Can be serialized so inventory may be saved	175
BeeGame.Blocks.Leaves		178
BeeGame.Terrain.Chunks.LoadChunks	Loads the Chunks around the player	181
BeeGame.LoadResources	Loads all of the resources in the game	186
BeeGame.Terrain.Chunks.MeshData	The data for a Chunks 's Mesh	188
BeeGame.Items.NormalBee		192
BeeGame.Blocks.Planks	Planks Block	195
BeeGame.Inventory.Player_Inventory.PlayerInventory	Controls the player inventory	198
BeeGame.Player.PlayerLook	The look for the player	204
BeeGame.Player.PlayerMove	Moves the player	207
BeeGame.Core.Dictionaries.PrefabDictionary	The prefabs available to the game	211
BeeGame.Items.QueenBee		212
BeeGame.Exceptions.QuestAlreadyExistsException		214
BeeGame.Items.QuestBook		215
BeeGame.Inventory.QuestBookInventory		218
BeeGame.Quest.QuestEvents		222
BeeGame.Quest.Quests		225
BeeGame.Resources.Resources	A strongly-typed resource class, for looking up localized strings, etc.	231
BeeGame.Terrain.Chunks.SaveChunk	Saves a Chunks modified Blocks for save optimisation	234
BeeGame.Player.SavePlayerPosition	Saves the player position	236
BeeGame.Player.Selector	Moves the Block selector	237
BeeGame.Serialization.Serialization	Serializes and Deserialises things	242
BeeGame.Items.SetBeeGOColours		250

BeeGame.Terrain.LandGeneration.Noise.SimplexNoise	
Implementation of the Perlin simplex noise, an improved Perlin noise algorithm. Based loosely on SimplexNoise1234 by Stefan Gustavson http://staffwww.itn.liu.se/~stegu/aqsis/aqsis-newnoise/	252
BeeGame.SpawnItem	259
BeeGame.Core.Dictionaries.SpriteDictionary	
All of the sprites available to the game	260
BeeGame.Terrain.LandGeneration.Terrain	
Should use as an interface between the rest of the game and the terrain	262
BeeGame.Terrain.LandGeneration.TerrainGeneration	
Generates the terrain for the game	270
BeeGame.Test	279
BeeGame.Core.ThInput	
My implementation of the unity input system. Acts as a buffer layer to the unity system so that the input keys can be changed at runtime	280
BeeGame.Core.UnityTypeReplacements.ThQuaternion	285
BeeGame.Core.ThVector2	
Serializable version of Vector2	291
BeeGame.Core.ThVector3	
Serializable version of Vector3	298
BeeGame.Items.Tile	
Position of the items texture	312
BeeGame.Blocks.Wood	313
BeeGame.Terrain.LandGeneration.World	
Allows inter Chunk communication as it stores a list of active chunks	316

4 File Index

4.1 File List

Here is a list of all files with brief descriptions:

C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/LoadResources.cs	389
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/SpawnItem.cs	404
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/test.cs	426
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Air.cs	321
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Apiary.cs	322
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Bedrock.cs	326
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Block.cs	327

C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/ Chest.cs	328
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/ CraftingTable.cs	330
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/ Dirt.cs	331
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/ Grass.cs	332
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/ Leaves.cs	333
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/ Planks.cs	334
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/ Wood.cs	335
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/ Extensions.cs	344
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/ Dictionarys/BeeDictionaries.cs	336
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/ Dictionarys/CraftingRecipies.cs	338
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/ Dictionarys/EqualityComperors.cs	340
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/ Dictionarys/PrefabDictionary.cs	341
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/ Dictionarys/SpriteDictionary.cs	341
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/ Enums.cs	342
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/ UnityTypeReplacements/THInput.cs	346
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/ UnityTypeReplacements/THQuaternion.cs	348
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/ UnityTypeReplacements/THVector2.cs	350
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/ UnityTypeReplacements/THVector3.cs	352
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Exceptions/ CraftingRecipeAdditionException.cs	354
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Exceptions/ InputException.cs	355
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Exceptions/ QuestAlreadyExistsException.cs	356
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/ ApiaryCraftingOutputSlot.cs	357
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/ CraftingOutputSlot.cs	362

C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Inventory.cs	363
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/InventorySlot.cs	365
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/ItemsInInventory.cs	372
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/BlockInventory/ApiaryInventory.cs	357
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/BlockInventory/ChestInventory.cs	359
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/BlockInventory/CraftingTableInventory.cs	360
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/ItemInventory/BeeAlyzerInventory.cs	368
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/ItemInventory/QuestBookInventory.cs	370
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/PlayerInventory/PlayerInventory.cs	373
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/AbstractItem.cs	375
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/ApplyColour.cs	376
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/Bee.cs	376
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/BeeAlyzer.cs	379
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/Honey.cs	383
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/HoneyComb.cs	383
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/Item.cs	384
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/QuestBook.cs	388
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/GameObjectStuff/ItemGameObject.cs	381
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/GameObjectStuff/SetBeeGOColours.cs	382
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/obj/Debug/TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs	390
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/obj/Debug/TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs	390
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/obj/Debug/TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs	390
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/PlayerLook.cs	391
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/PlayerMove.cs	391

C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/ SavePlayer.cs	393
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/ Selector.cs	393
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Properties/ AssemblyInfo.cs	395
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Quest/ QuestEvents.cs	396
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Quest/ Quests.cs	397
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Resources/ ResourcesDesigner.cs	399
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Serialization/ Serialization.cs	401
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/ ChunkWorldPos.cs	413
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/ Chunk.cs	405
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/ LoadChunks.cs	408
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/ MeshData.cs	411
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/ SaveChunk.cs	412
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/LandGeneration/ Terrain.cs	418
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/LandGeneration/ TerrainGeneration.cs	421
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/LandGeneration/ World.cs	423
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/LandGeneration/ Noise/SimplexNoise.cs	414

5 Namespace Documentation

5.1 BeeGame Namespace Reference

Namespaces

- namespace [Blocks](#)
- namespace [Core](#)
- namespace [Exceptions](#)
- namespace [Inventory](#)
- namespace [Items](#)

- namespace [Player](#)
- namespace [Quest](#)
- namespace [Resources](#)
- namespace [Serialization](#)
- namespace [Terrain](#)

Classes

- class [LoadResources](#)
Loads all of the resources in the game
- class [SpawnItem](#)
- class [Test](#)

5.2 BeeGame.Blocks Namespace Reference

Classes

- class [Air](#)
Air Block is an empty block that does not render and has no collider
- class [Apiary](#)
Apiary Block
- class [Bedrock](#)
Bedrock Block
- class [Block](#)
Base class for blocks
- class [Chest](#)
Chest Block
- class [CraftingTable](#)
The Workbench Block class
- class [Dirt](#)
Dirt Block
- class [Grass](#)
Grass Block
- class [Leaves](#)
- class [Planks](#)
Planks Block
- class [Wood](#)

5.3 BeeGame.Core Namespace Reference

Namespaces

- namespace [Dictionaries](#)
- namespace [Enums](#)
- namespace [UnityTypeReplacements](#)

Classes

- class [Extensions](#)
- class [THInput](#)
My implementation of the unity input system. Acts as a buffer layer to the unity system so that the input keys can be changed at runtime
- struct [THVector2](#)
Serializable version of Vector2
- struct [THVector3](#)
Serializable version of Vector3

5.4 BeeGame.Core.Dictionaries Namespace Reference**Classes**

- class [BeeCombinationDictionaryEqualityComparer](#)
- class [BeeDictionaries](#)
- class [CraftingRecipes](#)
- class [PrefabDictionary](#)
The prefabs available to the game
- class [SpriteDictionary](#)
All of the sprites available to the game

5.5 BeeGame.Core.Enums Namespace Reference**Enumerations**

- enum [HoneyCombType](#) { [HoneyCombType.HONEY](#), [HoneyCombType.ICEY](#) }
Honey Comb Types
- enum [BeeSpecies](#) {
[BeeSpecies.FOREST](#), [BeeSpecies.MEADOWS](#), [BeeSpecies.TROPICAL](#), [BeeSpecies.WINTRY](#),
[BeeSpecies.MODEST](#), [BeeSpecies.MARSHY](#), [BeeSpecies.ENDER](#), [BeeSpecies.MONASTIC](#),
[BeeSpecies.STEADFAST](#), [BeeSpecies.VALIANT](#), [BeeSpecies.COMMON](#), [BeeSpecies.CULTIVATED](#),
[BeeSpecies.DILIGENT](#), [BeeSpecies.RURAL](#), [BeeSpecies.FARMERLY](#), [BeeSpecies.AGRARIAN](#),
[BeeSpecies.UNWEARY](#), [BeeSpecies.INDUSTRIOUS](#), [BeeSpecies.ICY](#), [BeeSpecies.GLACIAL](#),
[BeeSpecies.NOBLE](#), [BeeSpecies.IMPERIAL](#), [BeeSpecies.MAJESTIC](#), [BeeSpecies.MIRY](#),
[BeeSpecies.BOGGY](#), [BeeSpecies.HERIOC](#), [BeeSpecies.PHANTASMAL](#), [BeeSpecies.SPECTRAL](#),
[BeeSpecies.HERMETIC](#), [BeeSpecies.SECLUDED](#), [BeeSpecies.SINISTER](#), [BeeSpecies.FIENDISH](#),
[BeeSpecies.DEMONIC](#), [BeeSpecies.FRUGAL](#), [BeeSpecies.AUSTER](#), [BeeSpecies.VINDICTIVE](#),
[BeeSpecies.EXOTIC](#), [BeeSpecies.ENDEMIC](#), [BeeSpecies.VENGEFUL](#), [BeeSpecies.AVENGING](#),
[BeeSpecies.SETADFAST](#), [BeeSpecies.HEROIC](#) }
The different possible bee Species
- enum [BeeType](#) { [BeeType.QUEEN](#), [BeeType.DRONE](#), [BeeType.PRINCESS](#) }
The different bee types
- enum [BeeTempPreference](#) {
[BeeTempPreference.FROZEN](#), [BeeTempPreference.COLD](#), [BeeTempPreference.TEMPERATE](#), [BeeTempPreference.HOT](#),
[BeeTempPreference.HELL](#) }
The different bee temp preferences
- enum [BeeLifeSpan](#) {
[BeeLifeSpan.HUMMINGBIRD](#), [BeeLifeSpan.SHORTEST](#), [BeeLifeSpan.SHORT](#), [BeeLifeSpan.NORMAL](#),
[BeeLifeSpan.LONG](#), [BeeLifeSpan.LONGEST](#), [BeeLifeSpan.SEATURTLE](#) }

- The lifespan of the bee*
- enum BeeProductionSpeed { BeeProductionSpeed.SLOW, BeeProductionSpeed.NORMAL, BeeProductionSpeed.FAST }

How fast the bee produces items

 - enum BeeEffect { BeeEffect.NONE, BeeEffect.POISON }

Any effects of the bee

 - enum BeeHumidityPreference {
 BeeHumidityPreference.ARID, BeeHumidityPreference.DRY, BeeHumidityPreference.TEMPERATE, BeeHumidityPreference.MOIST,
 BeeHumidityPreference.HUMID
 }

Humidity preferences of the bee

 - enum Direction {
 Direction.NORTH, Direction.EAST, Direction.SOUTH, Direction.WEST,
 Direction.UP, Direction.DOWN
 }

Direction in the game

5.5.1 Enumeration Type Documentation

5.5.1.1 BeeEffect

```
enum BeeGame.Core.Enums.BeeEffect [strong]
```

Any effects of the bee

Enumerator

NONE	
POISON	

Definition at line 55 of file [Enums.cs](#).

```
00056  {
00057      NONE, POISON
00058 }
```

5.5.1.2 BeeHumidityPreference

```
enum BeeGame.Core.Enums.BeeHumidityPreference [strong]
```

Humidity preferences of the bee

Enumerator

ARID	
DRY	
TEMPERATE	
MOIST	
HUMID	

Definition at line 63 of file [Enums.cs](#).

```
00064      {
00065          ARID, DRY, TEMPERATE, MOIST, HUMID
00066      };
```

5.5.1.3 BeeLifeSpan

```
enum BeeGame.Core.Enums.BeeLifeSpan [strong]
```

The lifespan of the bee

Enumerator

HUMMINGBIRD	
SHORTEST	
SHORT	
NORMAL	
LONG	
LONGEST	
SEATURTLE	

Definition at line 39 of file [Enums.cs](#).

```
00040      {
00041          HUMMINGBIRD, SHORTEST, SHORT, NORMAL, LONG,
00042          LONGEST, SEATURTLE
00042      };
```

5.5.1.4 BeeProductionSpeed

```
enum BeeGame.Core.Enums.BeeProductionSpeed [strong]
```

How fast the bee produces items

Enumerator

SLOW	
NORMAL	
FAST	

Definition at line 47 of file [Enums.cs](#).

```
00048      {
00049          SLOW, NORMAL, FAST
00050      };
```

5.5.1.5 BeeSpecies

```
enum BeeGame.Core.Enums.BeeSpecies [strong]
```

The different possible bee Species

Enumerator

FOREST	
MEADOWS	
TROPICAL	
WINTRY	
MODEST	
MARSHY	
ENDER	
MONASTIC	
STEADFAST	
VALIANT	
COMMON	
CULTIVATED	
DILIGENT	
RURAL	
FARMERLY	
AGRARIAN	
UNWEARY	
INDUSTRIOUS	
ICY	
GLACIAL	
NOBLE	
IMPERIAL	
MAJESTIC	
MIRY	
BOGGY	
HERIOC	
PHANTASMAL	
SPECTRAL	
HERMETIC	
SECLUDED	
SINISTER	
FIENDISH	
DEMONIC	
FRUGAL	
AUSTER	
VINDICTIVE	
EXOTIC	
ENDEMICK	
VENGEFUL	
AVENGING	
SETADFAST	
HEROIC	

Definition at line 15 of file [Enums.cs](#).

```

00016      {
00017          FOREST, MEADOWS, TROPICAL, WINTRY, MODEST,
00018          MARSHY, ENDER, MONASTIC, STEADFAST, VALIANT,
00019          COMMON, CULTIVATED, DILIGENT, RURAL, FARMERLY,
00020          AGRARIAN, UNWEARY, INDUSTRIOUS, ICY, GLACIAL,
00021          NOBLE, IMPERIAL, MAJESTIC, MIRY, BOGGY, HEROIC,
00022          PHANTASMAL, SPECTRAL, HERMETIC, SECLUDED,
00023          SINISTER, FIENDISH, DEMONIC, FRUGAL, AUSTER,
00024          VINDICTIVE, EXOTIC, ENDEMIC, VENGEFUL, AVENGING,
00025          SETADFAST, HEROIC
00026      };

```

5.5.1.6 BeeTempPreference

enum [BeeGame.Core.Enums.BeeTempPreference](#) [strong]

The different bee temp preferences

Enumerator

FROZEN	
COLD	
TEMPERATE	
HOT	
HELL	

Definition at line 31 of file [Enums.cs](#).

```

00032      {
00033          FROZEN, COLD, TEMPERATE, HOT, HELL
00034      };

```

5.5.1.7 BeeType

enum [BeeGame.Core.Enums.BeeType](#) [strong]

The different bee types

Enumerator

QUEEN	
DRONE	
PRINCESS	

Definition at line 23 of file [Enums.cs](#).

```

00024      {
00025          QUEEN, DRONE, PRINCESS
00026      };

```

5.5.1.8 Direction

```
enum BeeGame.Core.Enums.Direction [strong]
```

Direction in the game

Enumerator

NORTH	
EAST	
SOUTH	
WEST	
UP	
DOWN	

Definition at line 72 of file [Enums.cs](#).

```
00073     {
00074         NORTH, EAST, SOUTH, WEST, UP, DOWN
00075     };
```

5.5.1.9 HoneyCombType

```
enum BeeGame.Core.Enums.HoneyCombType [strong]
```

Honey Comb Types

Enumerator

HONEY	
ICEY	

Definition at line 6 of file [Enums.cs](#).

```
00007     {
00008         HONEY, ICEY
00009     };
```

5.6 BeeGame.Core.UnityTypeReplacements Namespace Reference

Classes

- struct [THQuaternion](#)

5.7 BeeGame.Exceptions Namespace Reference

Classes

- class [CraftingRecipeAdditionException](#)
- class [InputException](#)
- class [QuestAlreadyExistsException](#)

5.8 BeeGame.Inventory Namespace Reference

Namespaces

- namespace [BlockInventory](#)
- namespace [Player_Inventory](#)

Classes

- class [ApiaryCraftingOutputSlot](#)
Overrides the
 Inventory for Apiarys Apiary
- class [ApiaryInventory](#)
Inventory for the chests
- class [BeeAlyzerInventory](#)
Inventory for the chests
- class [ChestInventory](#)
Inventory for the chests
- class [CraftingOutputSlot](#)
Overrides the
 Inventory
- class [Inventory](#)
Base class for all inventories in the game
- class [InventorySlot](#)
- class [ItemsInInventory](#)
Class that holds all of the items in the inventory. Can be serialized so inventory may be saved
- class [QuestBookInventory](#)

5.9 BeeGame.Inventory.BlockInventory Namespace Reference

Classes

- class [CraftingTableInventory](#)
Inventory for the CraftingTable Block

5.10 BeeGame.Inventory.Player_Inventory Namespace Reference

Classes

- class [PlayerInventory](#)
Controls the player inventory

5.11 BeeGame.Items Namespace Reference

Classes

- class [AbstractItem](#)
Does this need to exist?
- class [ApplyColour](#)
Applies a given colour to a gameobject
- class [Bee](#)
The bee item
- class [BeeAlyzer](#)
- class [Honey](#)
- class [HoneyComb](#)
Honey comb item produced by bees
- class [Item](#)
Base class for all [Items](#) and [Blocks](#) in the game
- class [ItemGameObject](#)
Interface between item and unity gameobjects
- class [NormalBee](#)
- class [QueenBee](#)
- class [QuestBook](#)
- class [SetBeeGOColours](#)
- struct [Tile](#)
Position of the items texture

5.12 BeeGame.Player Namespace Reference

Classes

- class [PlayerLook](#)
The look for the player
- class [PlayerMove](#)
Moves the player
- class [SavePlayerPosition](#)
Saves the player position
- class [Selector](#)
Moves the Block selector

5.13 BeeGame.Quest Namespace Reference

Classes

- class [QuestEvents](#)
- class [Quests](#)

5.14 BeeGame.Resources Namespace Reference

Classes

- class [Resources](#)
A strongly-typed resource class, for looking up localized strings, etc.

5.15 BeeGame.Serialization Namespace Reference

Classes

- class [Serialization](#)
Serializes and Deserialises things

5.16 BeeGame.Terrain Namespace Reference

Namespaces

- namespace [Chunks](#)
- namespace [LandGeneration](#)

Classes

- struct [ChunkWorldPos](#)
Serializable int version of THVector3

5.17 BeeGame.Terrain.Chunks Namespace Reference

Classes

- class [Chunk](#)
A section of land for the game, used so that land can be generated in parts and not all at once
- class [LoadChunks](#)
Loads the [Chunks](#) around the player
- class [MeshData](#)
The data for a [Chunks](#)'s Mesh
- class [SaveChunk](#)
Saves a [Chunks](#) modified Blocks for save optimisation

5.18 BeeGame.Terrain.LandGeneration Namespace Reference

Namespaces

- namespace [Noise](#)

Classes

- class [Terrain](#)
Should use as an interface between the rest of the game and the terrain
- class [TerrainGeneration](#)
Generates the terrain for the game
- class [World](#)
Allows inter Chunk communication as it stores a list of active chunks

5.19 BeeGame.Terrain.LandGeneration.Noise Namespace Reference

Classes

- class [SimplexNoise](#)

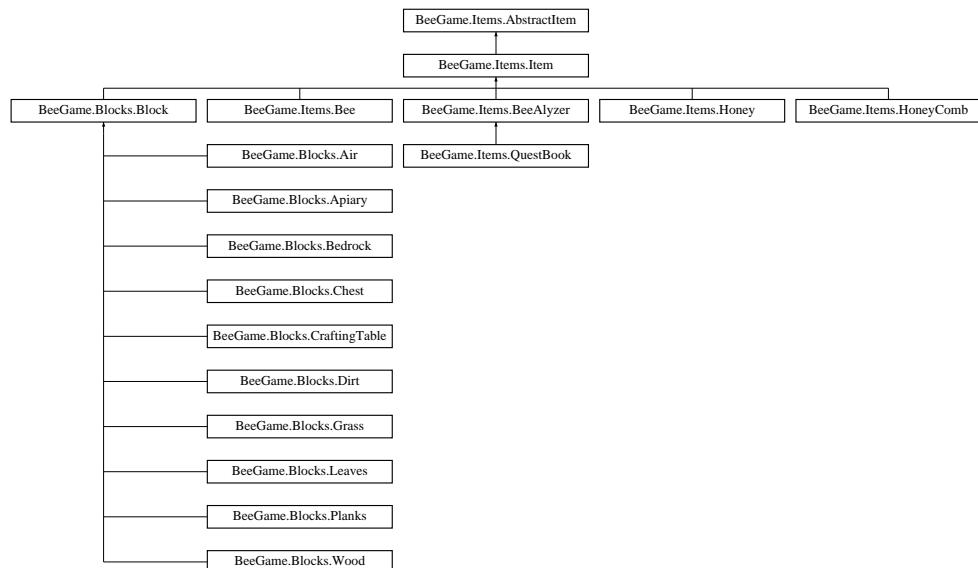
Implementation of the Perlin simplex noise, an improved Perlin noise algorithm. Based loosely on SimplexNoise1234 by Stefan Gustavson <http://staffwww.itn.liu.se/~stegu/aqsis/aqsis-newnoise/>

6 Class Documentation

6.1 BeeGame.Items.AbstractItem Class Reference

Does this need to exist?

Inheritance diagram for BeeGame.Items.AbstractItem:



Public Member Functions

- abstract string [GetItemName \(\)](#)
- abstract string [GetItemID \(\)](#)
- abstract override int [GetHashCode \(\)](#)

6.1.1 Detailed Description

Does this need to exist?

Definition at line 12 of file [AbstractItem.cs](#).

6.1.2 Member Function Documentation

6.1.2.1 GetHashCode()

```
abstract override int BeeGame.Items.AbstractItem.GetHashCode ( ) [pure virtual]
```

Implemented in [BeeGame.Items.Item](#), [BeeGame.Items.Bee](#), [BeeGame.Blocks.CraftingTable](#), [BeeGame.Blocks.Block](#), [BeeGame.Blocks.Chest](#), [BeeGame.Items.BeeAlyzer](#), [BeeGame.Blocks.Apiary](#), [BeeGame.Items.HoneyComb](#), [BeeGame.Blocks.Grass](#), [BeeGame.Items.QuestBook](#), [BeeGame.Blocks.Bedrock](#), [BeeGame.Blocks.Air](#), [BeeGame.Blocks.Dirt](#), [BeeGame.Blocks.Planks](#), [BeeGame.Blocks.Leaves](#), [BeeGame.Blocks.Wood](#), and [BeeGame.Items.Honey](#).

6.1.2.2 GetItemID()

```
abstract string BeeGame.Items.AbstractItem.GetItemID ( ) [pure virtual]
```

Implemented in [BeeGame.Items.Bee](#), [BeeGame.Items.BeeAlyzer](#), [BeeGame.Items.HoneyComb](#), [BeeGame.Items.Item](#), and [BeeGame.Items.Honey](#).

6.1.2.3 GetItemName()

```
abstract string BeeGame.Items.AbstractItem.GetItemName ( ) [pure virtual]
```

Implemented in [BeeGame.Items.Item](#).

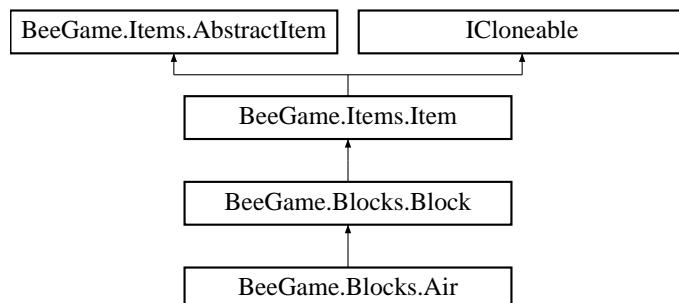
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/[AbstractItem.cs](#)

6.2 BeeGame.Blocks.Air Class Reference

[Air Block](#) is an empty block that does not render and has no collider

Inheritance diagram for BeeGame.Blocks.Air:



Public Member Functions

- [Air \(\)](#)
- [override void BreakBlock \(THVector3 pos\)](#)

No item should be made when air is broken
- [override MeshData BlockData \(Chunk chunk, int x, int y, int z, MeshData meshData, bool addRoRender←Mesh=true\)](#)

Returns the given MeshData as Air does not add anything to the mesh
- [override bool IsSolid \(Direction direction\)](#)
- [override int GetHashCode \(\)](#)

Hashcode acts as the base ID for an item
- [override string ToString \(\)](#)

Gets the item name and ID in a nice format

Static Public Attributes

- [static new int ID => 0](#)

Additional Inherited Members

6.2.1 Detailed Description

[Air Block](#) is an empty block that does not render and has no collider

Definition at line [12](#) of file [Air.cs](#).

6.2.2 Constructor & Destructor Documentation

6.2.2.1 Air()

BeeGame.Blocks.Air.Air ()

Definition at line [16](#) of file [Air.cs](#).

```
00016           : base("Air")
00017           {
00018           }
```

6.2.3 Member Function Documentation

6.2.3.1 BlockData()

```
override MeshData BeeGame.Blocks.Air.BlockData (
    Chunk chunk,
    int x,
    int y,
    int z,
    MeshData meshData,
    bool addRoRenderMesh = true ) [virtual]
```

Returns the given MeshData as Air does not add anything to the mesh

Returns

Given MeshData

Reimplemented from [BeeGame.Blocks.Block](#).

Definition at line 33 of file [Air.cs](#).

```
00034     {
00035         return meshData;
00036     }
```

6.2.3.2 BreakBlock()

```
override void BeeGame.Blocks.Air.BreakBlock (
    THVector3 pos ) [virtual]
```

No item should be made when air is broken

Parameters

<i>pos</i>	position to spawn the Item
------------	--

Reimplemented from [BeeGame.Blocks.Block](#).

Definition at line 24 of file [Air.cs](#).

```
00025     {
00026         return;
00027     }
```

6.2.3.3 GetHashCode()

```
override int BeeGame.Blocks.Air.GetHashCode ( ) [virtual]
```

Hashcode acts as the base ID for an item

Returns**2**

Implements [BeeGame.Items.AbstractItem](#).

Definition at line [52](#) of file [Air.cs](#).

```
00053     {
00054         return ID;
00055     }
```

6.2.3.4 IsSolid()

```
override bool BeeGame.Blocks.Air.IsSolid (
    Direction direction) [virtual]
```

Parameters

<i>direction</i>	Direction wanted to check solid
------------------	---------------------------------

Returns**false**

Reimplemented from [BeeGame.Blocks.Block](#).

Definition at line [43](#) of file [Air.cs](#).

```
00044     {
00045         return false;
00046     }
```

6.2.3.5 ToString()

```
override string BeeGame.Blocks.Air.ToString ()
```

Gets the item name and ID in a nice format

Returns

Definition at line [61](#) of file [Air.cs](#).

```
00062     {
00063         return $"{itemName} \nID: {GetItemID()}";
00064     }
```

6.2.4 Member Data Documentation

6.2.4.1 ID

```
new int BeeGame.Blocks.Air.ID => 0 [static]
```

Definition at line 14 of file [Air.cs](#).

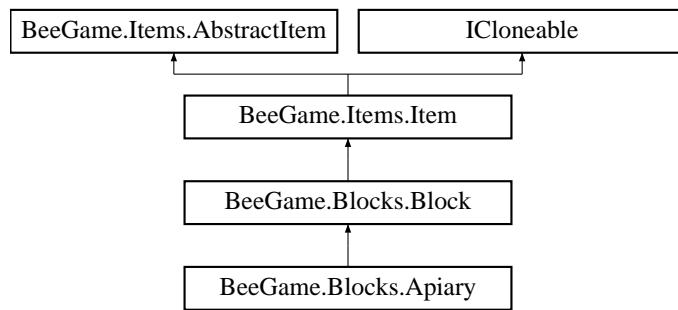
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/[Air.cs](#)

6.3 BeeGame.Blocks.Apiary Class Reference

[Apiary Block](#)

Inheritance diagram for BeeGame.Blocks.Apiary:



Public Member Functions

- [Apiary \(\)](#)
Constructor
- [override GameObject GetGameObject \(\)](#)
Gets the game object for this apiary
- [override Tile TexturePosition \(Direction direction\)](#)
Returns the texture for the apiary Block
- [override MeshData BlockData \(Chunk chunk, int x, int y, int z, MeshData meshData, bool addToRender ← Mesh=true\)](#)
The data that this block adds to the mesh
- [override void BreakBlock \(THVector3 pos\)](#)
Breaks the block
- [override Sprite GetItemSprite \(\)](#)
Returns the sprite for the item
- [override int GetHashCode \(\)](#)
ID of the item
- [override string ToString \(\)](#)
The item name and ID as a string
- [override bool InteractWithBlock \(Inventory.Inventory inv\)](#)
Toggles the ApiaryInventory for the block
- [void MakeBees \(Bee queen, ref Item\[\] inventory\)](#)
Will make new Bee/Items from the given BeeType.QUEEN Bee
- [Bee MakeBee \(BeeType beeType, QueenBee queen\)](#)
Makes a new Bee
- [BeeProductionSpeed CombineProductionSpeed \(BeeProductionSpeed b1, BeeProductionSpeed b2\)](#)
Combines the BeeProductionSpeed of the given BeeProductionSpeed

Public Attributes

- int [mutationMultiplier](#)

Static Public Attributes

- static new int [ID](#) => 10

Private Member Functions

- [BeeSpecies CombineSpecies \(BeeSpecies s1, BeeSpecies s2\)](#)
Returns a BeeSpecies depending on the given BeeSpecies
- float [Rand \(float\[\] weights\)](#)
Returns a random float bewteen 0 and the sum of weights rounded to 2dp
- [BeeLifeSpan CombineLifespan \(BeeLifeSpan b1, BeeLifeSpan b2\)](#)
Combines the BeeLifeSpan of the given BeeLifeSpan
- uint [CombineFertility \(uint b1, uint b2\)](#)
Combines the fertility of the given fertility
- [BeeEffect CombineEffect \(BeeEffect b1, BeeEffect b2\)](#)
Combines the BeeEffect of the given BeeEffect
- int [ReturnChange \(int b1, int b2, int maxChange, int minChange=0\)](#)
Returns a number between maxChange and minChange based of b1 and b2

Private Attributes

- GameObject [myGameobject](#)

Additional Inherited Members

6.3.1 Detailed Description

Apiary Block

Definition at line [17](#) of file [Apiary.cs](#).

6.3.2 Constructor & Destructor Documentation

6.3.2.1 Apiary()

```
BeeGame.Blocks.Apiary.Apiary ( )
```

Constructor

Definition at line [30](#) of file [Apiary.cs](#).

```
00030             : base("Apiary")
00031         {
00032             usesGameObject = true;
00033         }
```

6.3.3 Member Function Documentation

6.3.3.1 BlockData()

```
override MeshData BeeGame.Blocks.Apiary.BlockData (
    Chunk chunk,
    int x,
    int y,
    int z,
    MeshData meshData,
    bool addToRenderMesh = true ) [virtual]
```

The data that this block adds to the mesh

Parameters

<i>chunk</i>	Chunk the block is in
<i>x</i>	X pos of the block
<i>y</i>	Y pos of the block
<i>z</i>	Z pos of the block
<i>meshData</i>	meshdata to add to
<i>addToRenderMesh</i>	should the block also be added to the render mesh not just the collision mesh

Returns

Given *meshData* with this blocks data added to it

Only adds to the collision mesh as the model is handled by the unity prefab system

Reimplemented from [BeeGame.Blocks.Block](#).

Definition at line [72](#) of file [Apiary.cs](#).

```
00073     {
00074         if (myGameObject == null)
00075         {
00076             myGameObject = UnityEngine.Object.Instantiate(
00077                 PrefabDictionary.GetPrefab("Apiary"), new THVector3(x, y, z) + chunk.
00078                 chunkWorldPos, Quaternion.identity, chunk.transform);
00079             myGameObject.GetComponent<ChestInventory>().inventoryPosition =
00080                 new THVector3(x, y, z) + chunk.chunkWorldPos;
00081             myGameObject.GetComponent<ChestInventory>().SetChestInventory();
00082         }
00083         return base.BlockData(chunk, x, y, z, meshData, true);
00084     }
```

6.3.3.2 BreakBlock()

```
override void BeeGame.Blocks.Apiary.BreakBlock (
    THVector3 pos ) [virtual]
```

Breaks the block

Parameters

<i>pos</i>	Position of the block
------------	-----------------------

Reimplemented from [BeeGame.Blocks.Block](#).

Definition at line 87 of file [Apiary.cs](#).

```
00088      {
00089          /* removes the blocks blocks inventory save file and destroys the game object
00090          Serialization.Serialization.DeleteFile(myGameObject.GetComponent<
00091              ApiaryInventory>().inventoryName);
00092          UnityEngine.Object.Destroy(myGameObject);
00093          /* removes the collision mesh from the chunk
00094          base.BreakBlock(pos);
00094 }
```

6.3.3.3 CombineEffect()

```
BeeEffect BeeGame.Blocks.Apiary.CombineEffect (
    BeeEffect b1,
    BeeEffect b2 ) [private]
```

Combines the BeeEffect of the given BeeEffect

Parameters

<i>b1</i>	First BeeEffect
<i>b2</i>	Second BeeEffect

Returns

A new BeeEffect

Definition at line 315 of file [Apiary.cs](#).

```
00316      {
00317          return (BeeEffect)ReturnChange((int)b1, (int)b2, (int)
00318              BeeEffect.POISON);
00318 }
```

6.3.3.4 CombineFertility()

```
uint BeeGame.Blocks.Apiary.CombineFertility (
    uint b1,
    uint b2 ) [private]
```

Combines the fertility of the given fertility

Parameters

<i>b1</i>	First Bees fertility
<i>b2</i>	Second Bees fertility

Returns

A new fertility, uint

Definition at line 304 of file [Apiary.cs](#).

```
00305      {
00306          return (uint)ReturnChange((int)b1, (int)b2, 5, 1);
00307      }
```

6.3.3.5 CombineLifespan()

```
BeeLifeSpan BeeGame.Blocks.Apiary.CombineLifespan (
    BeeLifeSpan b1,
    BeeLifeSpan b2 ) [private]
```

Combines the BeeLifeSpan of the given BeeLifeSpan

Parameters

<i>b1</i>	First BeeLifeSpan
<i>b2</i>	Second BeeLifeSpan

Returns

A new BeeLifeSpan

Definition at line 293 of file [Apiary.cs](#).

```
00294      {
00295          return (BeeLifeSpan)ReturnChange((int)b1, (int)b2, (int)
00296          BeeLifeSpan.SEATURTLE);
00296      }
```

6.3.3.6 CombineProductionSpeed()

```
BeeProductionSpeed BeeGame.Blocks.Apiary.CombineProductionSpeed (
    BeeProductionSpeed b1,
    BeeProductionSpeed b2 )
```

Combines the BeeProductionSpeed of the given BeeProductionSpeed

Parameters

<i>b1</i>	First BeeProductionSpeed
<i>b2</i>	Second BeeProductionSpeed

Returns

A new BeeProductionSpeed

Definition at line 326 of file [Apiary.cs](#).

```
00327      {
00328          return (BeeProductionSpeed)ReturnChange((int)b1, (int)b2, (int)
00329              BeeProductionSpeed.FAST);
00329 }
```

6.3.3.7 CombineSpecies()

```
BeeSpecies BeeGame.Blocks.Apiary.CombineSpecies (
    BeeSpecies s1,
    BeeSpecies s2 ) [private]
```

Returns a BeeSpecies depending on the given BeeSpecies

Parameters

<i>s1</i>	First BeeSpecies
<i>s2</i>	Second BeeSpecies

Returns

A new BeeSpecies

Definition at line 246 of file [Apiary.cs](#).

```
00247      {
00248          BeeSpecies[] possibleSpecies = BeeDictionaries.
00249              GetCombinations(s1, s2);
00249          float[] weights = possibleSpecies.Length > 2 ? BeeDictionaries.
00249              GetWeights(possibleSpecies) : new float[] { 0.5f, 0.5f };
00250
00251          var randomNum = Rand(weights);
00252          var weightsSum = 0f;
00253
00254          /* when the number generated is less than the current sum of the weights return that bee
00255          for (int i = 0; i < weights.Length; i++)
00256          {
00257              if(randomNum <= weightsSum)
00258              {
00259                  return possibleSpecies[i];
00260              }
00261
00262              weightsSum += weights[i];
00263          }
00264
00265          /* if for some reason the weights cannot work return the first bee in the combination list
00266          return possibleSpecies[0];
00267 }
```

6.3.3.8 GetGameObject()

```
override GameObject BeeGame.Blocks.Apiary.GetGameObject () [virtual]
```

Gets the game object for this apiary

Returns

The chest game object

Reimplemented from [BeeGame.Items.Item](#).

Definition at line 41 of file [Apiary.cs](#).

```
00042     {
00043         return PrefabDictionary.GetPrefab("Apiary");
00044     }
```

6.3.3.9 GetHashCode()

```
override int BeeGame.Blocks.Apiary.GetHashCode () [virtual]
```

ID of the item

Returns

3

Implements [BeeGame.Items.AbstractItem](#).

Definition at line 107 of file [Apiary.cs](#).

```
00108     {
00109         return ID;
00110     }
```

6.3.3.10 GetItemSprite()

```
override Sprite BeeGame.Blocks.Apiary.GetItemSprite () [virtual]
```

Returns the sprite for the item

Returns

Sprite for this item

Reimplemented from [BeeGame.Items.Item](#).

Definition at line 96 of file [Apiary.cs](#).

```
00097     {
00098         return SpriteDictionary.GetSprite("Apiary");
00099     }
```

6.3.3.11 InteractWithBlock()

```
override bool BeeGame.Blocks.Apiary.InteractWithBlock (
    Inventory.Inventory inv )
```

Toggles the ApiaryInventory for the block

Parameters

<i>inv</i>	
------------	--

Returns

Definition at line 127 of file [Apiary.cs](#).

```
00128      {
00129          myGameObject.GetComponent<ApiaryInventory>().myblock = this;
00130          myGameObject.GetComponent<ApiaryInventory>().ToggleInventory(inv);
00131          return true;
00132      }
```

6.3.3.12 MakeBee()

```
Bee BeeGame.Blocks.Apiary.MakeBee (
    BeeType beeType,
    QueenBee queen )
```

Makes a new Bee

Parameters

<i>beeType</i>	The type of bee to make, BeeType
<i>queen</i>	The stats the new Bee should be made with, QueenBee

Returns

A new Bee

Definition at line 213 of file [Apiary.cs](#).

```
00214      {
00215          /* gives all of the primary and secondary stats to the bee
00216          NormalBee nb = new NormalBee()
00217          {
00218              pSpecies = CombineSpecies(queen.queen.
00219              sSpecies, queen.drone.sSpecies),
00220              sSpecies = CombineSpecies(queen.queen.
00221              sSpecies, queen.drone.sSpecies),
00222              pEffect = CombineEffect(queen.queen.sEffect, queen.
00223              drone.sEffect),
00224              sEffect = CombineEffect(queen.queen.sEffect, queen.
00225              drone.sEffect),
00226              pFertility = CombineFertility(queen.queen.
00227              sFertility, queen.drone.sFertility),
00228              sFertility = CombineFertility(queen.queen.
00229              sFertility, queen.drone.sFertility),
00230              pLifespan = CombineLifespan(queen.queen.
00231              sLifespan, queen.drone.sLifespan),
00232              sLifespan = CombineLifespan(queen.queen.
00233              sLifespan, queen.drone.sLifespan),
```

```

00229             pProdSpeed = CombineProductionSpeed(queen.
00230                 queen.sProdSpeed, queen.drone.sProdSpeed),
00231                 sProdSpeed = CombineProductionSpeed(queen.
00232                     queen.sProdSpeed, queen.drone.sProdSpeed)
00233             );
00234         //QuestEvents.CallBeeCraftedEvent(nb.pSpecies);
00235
00236         /* returns the new bee
00237         return new Bee(beeType, nb);
00238     }

```

6.3.3.13 MakeBees()

```

void BeeGame.Blocks.Apiary.MakeBees (
    Bee queen,
    ref Item [ ] inventory )

```

Will make new Bee/Items from the given BeeType.QUEEN Bee

Parameters

<i>queen</i>	The BeeType.QUEEN to make the new Bees from
<i>inventory</i>	Inventory.Inventory to put the new Bees/Items into

Inventory is passed by reference to make it easier to modify the inventory. However is not necessarily needed as a class array is being passed so a reference would be created anyway however so **ref** is their more for clarity due to the function modifying the inventory directly

Definition at line 143 of file [Apiary.cs](#).

```

00144         {
00145             Item[] producedItems = new Item[9];
00146
00147             /* will always return a new princess and drone
00148             producedItems[0] = MakeBee(BeeType.PRINCESS, queen.
00149                 queenBee);
00150             producedItems[1] = MakeBee(BeeType.DRONE, queen.
00151                 queenBee);
00152
00153             var repeats = UnityEngine.Random.Range(0, queen.queenBee.
00154                 queen.pFertility);
00155
00156             /* produces as many other children as the bee stats will allow
00157             for (int i = 0; i < repeats; i++)
00158             {
00159                 producedItems[i + 2] = MakeBee(queen.queenBee.
00160                     queen.pFertility > 6 ? (BeeType)UnityEngine.Random.Range(1, 3) :
00161                     BeeType.DRONE, queen.queenBee);
00162
00163                 if (producedItems[i + 2] is Bee b && b.beeType !=
00164                     BeeType.PRINCESS)
00165                     producedItems[i + 2].itemStackCount =
00166                         UnityEngine.Random.Range(1, (int)queen.queenBee.queen.
00167                             pFertility + 1);
00168
00169             /* gets the produced items
00170             var beeProduce = BeeDictionaries.GetBeeProduce(queen.
00171                 queenBee.queen.pSpecies);
00172
00173             /* changes the stack count of the produced items to the correct number
00174             for (int i = 0; i < beeProduce.Length; i++)
00175             {
00176                 beeProduce[i].itemStackCount += UnityEngine.Random.Range(1, (int)
00177                     queen.queenBee.queen.sProdSpeed + 1);
00178             }

```

```

00170
00171     /* adds the items that the bee species produces into the produced item array
00172     for (int i = (int)queen.queenBee.queen.pFertility + 2, prod = 0; prod <
00173         beeProduce.Length; i++, prod++)
00174     {
00175         producedItems[i] = beeProduce[prod];
00176     }
00177
00178     /* puts the items into the inventory
00179     for (int i = 0; i < 9; i++)
00180     {
00181         if (inventory[i + 2] != null)
00182         {
00183             /* if the slot has the same item in it and it won't be more than the max stack count
00184             but the new item into it
00185             if (producedItems[i] == inventory[i + 2] && inventory[i + 2].
00186                 itemStackCount + 1 <= inventory[i + 2].maxStackCount)
00187                 inventory[i + 2].itemStackCount++;
00188             else
00189                 /* otherwise find a new slot to put the item into
00190                 for (int j = i; j < (9 - i); j++)
00191                 {
00192                     if (inventory[j + 2] == null)
00193                     {
00194                         inventory[j + 2] = producedItems[i];
00195                         break;
00196                     }
00197                     else if (producedItems[i] == inventory[j + 2] && inventory[j + 2].
00198                         itemStackCount + 1 <= inventory[j + 2].maxStackCount)
00199                         {
00200                             inventory[j + 2].itemStackCount++;
00201                             break;
00202                         }
00203                     }
00204                 }
00205             /* if the slot is empty put the item into it
00206             else
00207                 inventory[i + 2] = producedItems[i];
00208         }
00209     }

```

6.3.3.14 Rand()

```

float BeeGame.Blocks.Apiary.Rand (
    float [] weights ) [private]

```

Returns a random float bewteen 0 and the sum of *weights* rounded to 2dp

Parameters

<i>weights</i>	The weights
----------------	-------------

Returns

float bewteen 0 and the sum of *weights* rounded to 2dp

Definition at line 274 of file [Apiary.cs](#).

```

00275     {
00276         var totalWeights = 0f;
00277
00278         /* sums the weights
00279         for (int i = 0; i < weights.Length; i++)
00280         {
00281             totalWeights += weights[i];
00282         }
00283
00284         return (float)Math.Round(UnityEngine.Random.Range(0, totalWeights), 2);
00285     }

```

6.3.3.15 ReturnChange()

```
int BeeGame.Blocks.Apiary.ReturnChange (
    int b1,
    int b2,
    int maxChange,
    int minChange = 0 ) [private]
```

Returns a number between *maxChange* and *minChange* based of *b1* and *b2*

Parameters

<i>b1</i>	First number
<i>b2</i>	Second number
<i>maxChange</i>	Max return value
<i>minChange</i>	Min return value

Returns

A number between *maxChange* and *minChange*

If *b1* and *b2* are the same their is still a chance of change due to this function also takeing [mutationMultipler](#), the value of wich is dictated by the apairy

Definition at line [342](#) of file [Apiary.cs](#).

```
00343      {
00344          /* b1 and b2 are checked for which one is larger here as the
00345          /* queen may have a lower stat than the drone as the drone is always passed in second
00346          var change = UnityEngine.Random.Range(b1 < b2 ? b1 : b2, (b2 > b1 ? b2 : b1) + 2);
00347          /* this will make it possible for the bees to mutate during combination of the stats are the
00348          same
00349          /* it will also cause more random mutation more mimicking nature
00350          change += UnityEngine.Random.Range(-mutationMultipler,
00351          mutationMultipler);
00352          /* as all of the stats are enums they have a min/max value so need to check that this is not
00353          exceeded
00354          if (change > maxChange)
00355              change = maxChange;
00356          else if (minChange > change)
00357              change = minChange;
00358
00359          return change;
00360      }
```

6.3.3.16 TexturePosition()

```
override Tile BeeGame.Blocks.Apiary.TexturePosition (
    Direction direction ) [virtual]
```

Returns the texture for the apairy [Block](#)

Parameters

<i>direction</i>	Direction of the desired face
------------------	-------------------------------

Returns

Tile with the texture coordinates of the [Block](#) texture

Returns a transparent texture as the chest model already has a texture applied

Reimplemented from [BeeGame.Items.Item](#).

Definition at line 54 of file [Apiary.cs](#).

```
00055      {  
00056          return new Tile() { x = 0, y = 9 };  
00057      }
```

6.3.3.17 [ToString\(\)](#)

```
override string BeeGame.Blocks.Apiary.ToString ()
```

The item name and ID as a string

Returns

A nicely formatted string

Definition at line 116 of file [Apiary.cs](#).

```
00117      {  
00118          return $"{itemName} \nID: {GetItemID()}";  
00119      }
```

6.3.4 Member Data Documentation

6.3.4.1 [ID](#)

```
new int BeeGame.Blocks.Apiary.ID => 10 [static]
```

Definition at line 24 of file [Apiary.cs](#).

6.3.4.2 [mutationMultiplyer](#)

```
int BeeGame.Blocks.Apiary.mutationMultiplyer
```

Definition at line 22 of file [Apiary.cs](#).

6.3.4.3 myGameobject

GameObject BeeGame.Blocks.Apiary.myGameobject [private]

Definition at line 20 of file [Apiary.cs](#).

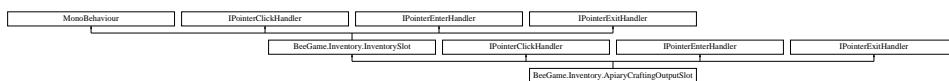
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/[Apiary.cs](#)

6.4 BeeGame.Inventory.ApiaryCraftingOutputSlot Class Reference

Overrides the

Inheritance diagram for BeeGame.Inventory.ApiaryCraftingOutputSlot:



Public Member Functions

- override void [OnPointerClick](#) (PointerEventData eventData)

Gives extra functionality to the base slot

Protected Member Functions

- new void [Update](#) ()

Updates the base slot things

Additional Inherited Members

6.4.1 Detailed Description

Overrides the

Definition at line 13 of file [ApiaryCraftingOutputSlot.cs](#).

6.4.2 Member Function Documentation

6.4.2.1 OnPointerClick()

```
override void BeeGame.Inventory.ApiaryCraftingOutputSlot.OnPointerClick (
    PointerEventData eventData) [virtual]
```

Gives extra functionality to the base slot

Parameters

<i>eventData</i>	
------------------	--

Reimplemented from [BeeGame.Inventory.InventorySlot](#).

Definition at line 28 of file [ApiaryCraftingOutputSlot.cs](#).

```

00029      {
00030          /* recorded what item was in the slot before it is moved
00031          Item before = item;
00032
00033          base.OnPointerClick(eventData);
00034
00035          /* if the item is different now then the crafting result must have been removed so call the
00036          event
00037          if (before != item && before != null)
00038              ((CraftingTableInventory)myInventory).
00039          craftingResultRemoved.Invoke();
00040
00041          if (before is Bee b)
00042              QuestEvents.CallBeeCraftedEvent(b.normalBee?.pSpecies ?? b.
00043          queenBee.queen.pSpecies);
00044          else
00045              QuestEvents.CallItemCraftedEvent(before.
00046          GetHashCode());
00047      }

```

6.4.2.2 Update()

```
new void BeeGame.Inventory.ApiaryCraftingOutputSlot.Update () [protected]
```

Updates the base slot things

Definition at line 18 of file [ApiaryCraftingOutputSlot.cs](#).

```

00019      {
00020          CheckItem();
00021          UpdateIcon();
00022      }

```

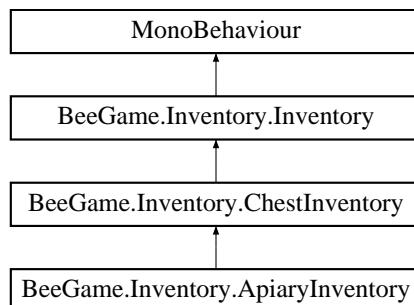
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/[ApiaryCraftingOutputSlot.cs](#)

6.5 BeeGame.Inventory.ApiaryInventory Class Reference

[Inventory](#) for Apiarys Apiary

Inheritance diagram for BeeGame.Inventory.ApiaryInventory:



Public Member Functions

- override void [SetChestInventory](#) (string invName="Apiary")
Sets the size and name of this [Inventory](#)

Public Attributes

- float [combinationTime](#) = 0
How long does the current combineing bee have left
- Slider [timerSlideer](#)
Slider to give a visual indication of [combinationTime](#)

Private Member Functions

- void [Update](#) ()
Updates the block every frame
- void [FixedUpdate](#) ()
Updates the combination time because of this was frame rate dependand weird things would happen
- void [CheckforBees](#) ()
Checks and combines bees in inventory slots 1 and 2 (items.itemsInInventory index 0 and 1)

Private Attributes

- bool [beesCombineing](#)
Are bees currently combineing

Additional Inherited Members

6.5.1 Detailed Description

[Inventory](#) for Apiarys Apiary

The Apiary can exted thhe normal inventory as the basic functionality is the same ([Items](#) inside need to be saved, input/optut items, etc)

Definition at line 13 of file [ApiaryInventory.cs](#).

6.5.2 Member Function Documentation

6.5.2.1 CheckforBees()

```
void BeeGame.Inventory.ApiaryInventory.CheckforBees () [private]
```

Checks and combines bees in inventory slots 1 and 2 (items.itemsInInventory index 0 and 1)

Definition at line 73 of file [ApiaryInventory.cs](#).

```
00074      {
00075          Items.Item posOneItem = items.itemsInInventory[0];
00076          Items.Item posTwoItem = items.itemsInInventory[1];
00077
00078          /* the item is checkd if it is a bee and if it is then a new variable is made for convenience
00079          //if it is a queen then just set the combination time and go
00080          if (posOneItem is Items.Bee b && b.beeType == Core.Enums.BeeType.QUEEN)
00081          {
00082              combinationTime = ((float)b.queenBee.queen.pLifespan + 1) * 2;
00083              beesCombineing = true;
00084              SaveInv();
00085
00086              timerSlideer maxValue = combinationTime;
00087
00088              return;
00089          }
00090
00091          /* of one bee is a princess and another is a drone in the correct slots combine them
00092          if (posOneItem is Items.Bee b1 && posTwoItem is Items.Bee b2 && b1.beeType == Core.Enums.BeeType
00093 .PRINCESS && b2.beeType == Core.Enums.BeeType.DRONE)
00094          {
00095              /* convert the princess to a queen with the paired drone
00096              Items.Bee.ConvertToQueen(ref b1, b2.normalBee);
00097
00098              /* reduce number of drones in slot by 1 and check it is a valid stack number
00099              items.itemsInInventory[1].itemStackCount -= 1;
00100              slots[0].item = b1;
00101
00102              if (items.itemsInInventory[1].itemStackCount <= 0)
00103                  items.itemsInInventory[1] = null;
00104
00105              /* set the combination time
00106              combinationTime = ((float)b1.queenBee.queen.pLifespan + 1) * 2;
00107              beesCombineing = true;
00108
00109              SaveInv();
00110
00111              /* set the slider max to the combination time
00112              timerSlideer maxValue = combinationTime;
00113          }
00114      }
```

6.5.2.2 FixedUpdate()

```
void BeeGame.Inventory.ApiaryInventory.FixedUpdate () [private]
```

Updates the combination time because of this was frame rate dependand weird things would happen

Definition at line 61 of file [ApiaryInventory.cs](#).

```
00062      {
00063          /* if bees are combineing reduce the combination time
00064          if (beesCombineing)
00065              timerSlideer.value = combinationTime -= 0.1f;
00066      }
```

6.5.2.3 SetChestInventory()

```
override void BeeGame.Inventory.ApiaryInventory.SetChestInventory (
    string invName = "Apiary" ) [virtual]
```

Sets the size and name of this [Inventory](#)

Parameters

<i>invName</i>	
----------------	--

Reimplemented from [BeeGame.Inventory.ChestInventory](#).

Definition at line 121 of file [ApiaryInventory.cs](#).

```
00122     {
00123         base.SetChestInventory("Apiary");
00124     }
```

6.5.2.4 Update()

```
void BeeGame.Inventory.ApiaryInventory.Update () [private]
```

Updates the block every frame

Definition at line 36 of file [ApiaryInventory.cs](#).

```
00037     {
00038         /* Updates the base class as unity Update function does not run on parent classes
00039         UpdateChestInventory();
00040
00041         /* if the apiary is not an item on the ground and bees are not currently combineing check is
00042         bees should be combineing
00043         if(items.itemsInInventory.Length > 0 && !
00044             beesCombineing)
00045             CheckforBees();
00046
00047         /* if the currently combineing bees has finished combineing
00048         if (combinationTime < 0 && beesCombineing)
00049         {
00050             /* make the items that the bees should make and destroy the spent queen
00051             ((Apiary)myblock).MakeBees(items.
00052             itemsInInventory[0] as Items.Bee, ref items.
00053             itemsInInventory);
00054             beesCombineing = false;
00055             items.itemsInInventory[0] = null;
00056
00057             /* save the changes to the inventory
00058             SaveInv();
00059         }
00060     }
```

6.5.3 Member Data Documentation**6.5.3.1 beesCombineing**

```
bool BeeGame.Inventory.ApiaryInventory.beesCombineing [private]
```

Are bees currently combineing

Definition at line 19 of file [ApiaryInventory.cs](#).

6.5.3.2 combinationTime

```
float BeeGame.Inventory.ApiaryInventory.combinationTime = 0
```

How long does the current combineing bee have left

Definition at line 24 of file [ApiaryInventory.cs](#).

6.5.3.3 timerSlider

```
Slider BeeGame.Inventory.ApiaryInventory.timerSlider
```

Slider to give a visual indication of [combinationTime](#)

Definition at line 29 of file [ApiaryInventory.cs](#).

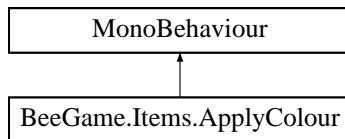
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/BlockInventory/[ApiaryInventory.cs](#)

6.6 BeeGame.Items.ApplyColour Class Reference

Applies a given colour to a gameobject

Inheritance diagram for BeeGame.Items.ApplyColour:



Public Attributes

- Color [colour](#)
Colour to apply
- GameObject [] [objects](#)
Objects to apply the colour too

Private Member Functions

- void [Start](#) ()
Applies the colour to the GameObjects in the [objects](#) array

6.6.1 Detailed Description

Applies a given colour to a gameobject

Definition at line 12 of file [ApplyColour.cs](#).

6.6.2 Member Function Documentation

6.6.2.1 Start()

```
void BeeGame.Items.ApplyColour.Start ( ) [private]
```

Applies the colour to the GameObjects in the `objects` array

Definition at line 32 of file [ApplyColour.cs](#).

```
00033     {
00034         /* applies the correct colour to each object in the array
00035         for (int i = 0; i < objects.Length; i++)
00036         {
00037             objects[i].GetComponent<Renderer>().material.SetColor("_OverlayColour",
00038             colour);
00039         }
}
```

6.6.3 Member Data Documentation

6.6.3.1 colour

```
Color BeeGame.Items.ApplyColour.colour
```

Colour to apply

Definition at line 18 of file [ApplyColour.cs](#).

6.6.3.2 objects

```
GameObject [ ] BeeGame.Items.ApplyColour.objects
```

Objects to apply the colour too

Array set in the editor

Definition at line 25 of file [ApplyColour.cs](#).

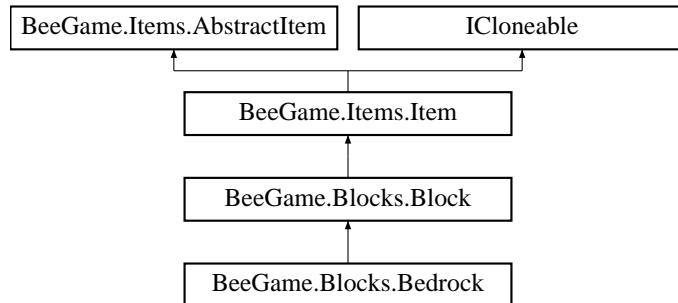
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/[ApplyColour.cs](#)

6.7 BeeGame.Blocks.Bedrock Class Reference

Bedrock Block

Inheritance diagram for BeeGame.Blocks.Bedrock:



Public Member Functions

- [Bedrock \(\)](#)
Constructor
- [override void BreakBlock \(THVector3 pos\)](#)
The block cannot be broken so nothing is done
- [override Tile TexturePosition \(Direction direction\)](#)
Position if the bedrock texture in the atlas
- [override int GetHashCode \(\)](#)
Returns the ID of the item
- [override string ToString \(\)](#)
The item name and ID as a string

Static Public Attributes

- [static new int ID => -1](#)

Additional Inherited Members

6.7.1 Detailed Description

Bedrock Block

Definition at line 12 of file [Bedrock.cs](#).

6.7.2 Constructor & Destructor Documentation

6.7.2.1 Bedrock()

```
BeeGame.Blocks.Bedrock.Bedrock ( )
```

Constructor

Definition at line 22 of file [Bedrock.cs](#).

```
00022             : base("Bedrock")
00023     {
00024         breakable = false;
00025     }
```

6.7.3 Member Function Documentation

6.7.3.1 BreakBlock()

```
override void BeeGame.Blocks.Bedrock.BreakBlock (
    THVector3 pos) [virtual]
```

The block cannot be broken so nothing is done

Parameters

<i>pos</i>	positon of the block
------------	----------------------

Reimplemented from [BeeGame.Blocks.Block](#).

Definition at line 33 of file [Bedrock.cs](#).

```
00034     {
00035         return;
00036     }
```

6.7.3.2 GetHashCode()

```
override int BeeGame.Blocks.Bedrock.GetHashCode () [virtual]
```

Returns the ID of the item

Returns

-1

Implements [BeeGame.Items.AbstractItem](#).

Definition at line 56 of file [Bedrock.cs](#).

```
00057     {
00058         return ID;
00059     }
```

6.7.3.3 TexturePosition()

```
override Tile BeeGame.Blocks.Bedrock.TexturePosition (
    Direction direction ) [virtual]
```

Position if the bedrock texture in the atlas

Parameters

<i>direction</i>	Direction
------------------	-----------

Returns

Position in the texture atlas

Reimplemented from [BeeGame.Items.Item](#).

Definition at line 45 of file [Bedrock.cs](#).

```
00046      {
00047          return new Tile() { x = 0, y = 0 };
00048      }
```

6.7.3.4 ToString()

```
override string BeeGame.Blocks.Bedrock.ToString ( )
```

The item name and ID as a string

Returns

A nicely formatted string

Definition at line 65 of file [Bedrock.cs](#).

```
00066      {
00067          return $"{itemName} \nID: {GetItemID()}";
00068      }
```

6.7.4 Member Data Documentation

6.7.4.1 ID

```
new int BeeGame.Blocks.Bedrock.ID => -1 [static]
```

Definition at line 15 of file [Bedrock.cs](#).

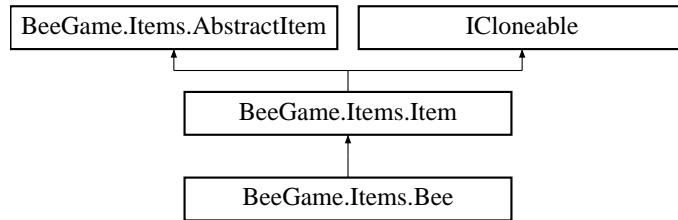
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/[Bedrock.cs](#)

6.8 BeeGame.Items.Bee Class Reference

The bee item

Inheritance diagram for BeeGame.Items.Bee:



Public Member Functions

- `Bee ()`
- `Bee (BeeType beeType, NormalBee normalBee)`
Create a bee from NormalBee
- `Bee (BeeType beeType, QueenBee queenBee)`
Create a bee from QueenBee
- override Sprite `GetItemSprite ()`
Returns the sprite for this, of the correct colour
- override GameObject `GetGameObject ()`
Returns the GameObject for the item of it has one
- override string `GetItemID ()`
Makes the item ID. For this it is the Normal ID \ the int value of the queenBee.GetHashCode() or normalBee.GetHashCode() as a string
- `Bee MakeBeeWithStats (BeeType beeType=BeeType.DRONE, BeeSpecies species=BeeSpecies.FOREST, BeeLifeSpan lifespan=BeeLifeSpan.NORMAL, uint fertility=2, BeeEffect effect=BeeEffect.NONE, BeeProductionSpeed prodSpeed=BeeProductionSpeed.NORMAL)`
Make a bee with given stats
- override int `GetHashCode ()`
Retuens the hashcode for this Item

Static Public Member Functions

- static void `ConvertToQueen (Bee princess, NormalBee drone)`
Will convery this bee to a BeeType.QUEEN useing this bees stats as the BeeType.PRINCESS stats
- static void `ConvertToQueen (ref Bee princess, NormalBee drone)`
Will Convert this bee into a BeeType.QUEEN Bee

Public Attributes

- bool `canSeeBeeData` = false
Can all of the bee data be seen when hovered over?

Static Public Attributes

- static new int `ID => 11`

Properties

- **BeeType beeType** [get, set]
This bees BeeType
- **BeeType previousBeeType** [get, set]
What was this bees BeeType?
- **override int maxStackCount** [get]
Overrided so can be set
- **QueenBee queenBee** [get, set]
If this bee is a BeeType.QUEEN this will be not null
- **NormalBee normalBee** [get, set]
If this bee is not a BeeType.QUEEN this will be not null

Private Attributes

- **int maxStack = 64**
- **Sprite itemSprite**
This bees Sprite

Additional Inherited Members**6.8.1 Detailed Description**

The bee item

Definition at line 14 of file [Bee.cs](#).

6.8.2 Constructor & Destructor Documentation**6.8.2.1 Bee() [1/3]**

```
BeeGame.Items.Bee.Bee ( )
```

Definition at line 58 of file [Bee.cs](#).

```
00059      {
00060          usesGameObject = true;
00061          normalBee = new NormalBee();
00062      }
```

6.8.2.2 Bee() [2/3]

```
BeeGame.Items.Bee.Bee (
    BeeType beeType,
    NormalBee normalBee )
```

Create a bee from [NormalBee](#)

Parameters

<i>beeType</i>	BeeType of the bee
<i>normalBee</i>	NormalBee data

Definition at line 70 of file [Bee.cs](#).

```
00070 : base(new CultureInfo("en-US", false).TextInfo.
00071     ToTitleCase($"{normalBee.pSpecies} {beeType}".ToLower()))
00072     {
00073         usesGameObject = true;
00074         if (beeType == BeeType.PRINCESS || beeType ==
00075             BeeType.QUEEN)
00076             maxStack = 1;
00077         this.beeType = beeType;
00078         this.normalBee = normalBee;
00079     }
```

6.8.2.3 Bee() [3/3]

```
BeeGame.Items.Bee.Bee (
    BeeType beeType,
    QueenBee queenBee )
```

Create a bee from [QueenBee](#)

Parameters

<i>beeType</i>	BeeType of the bee
<i>normalBee</i>	QueenBee data

Definition at line 84 of file [Bee.cs](#).

```
00084 : base(new CultureInfo("en-US", false).TextInfo.
00085     ToTitleCase($"{queenBee.queen.pSpecies} {beeType}".ToLower()))
00086     {
00087         usesGameObject = true;
00088         if (beeType == BeeType.PRINCESS || beeType ==
00089             BeeType.QUEEN)
00090             maxStack = 1;
00091         this.beeType = beeType;
00092         this.queenBee = queenBee;
00093     }
```

6.8.3 Member Function Documentation**6.8.3.1 ConvertToQueen() [1/2]**

```
static void BeeGame.Items.Bee.ConvertToQueen (
    Bee princess,
    NormalBee drone ) [static]
```

Will convert this bee to a BeeType.QUEEN using this bees stats as the BeeType.PRINCESS stats

Parameters

<i>drone</i>	<input type="checkbox"/>
--------------	--------------------------

Definition at line 155 of file [Bee.cs](#).

```
00156      {
00157          ConvertToQueen(ref princess, drone);
00158      }
```

6.8.3.2 ConvertToQueen() [2/2]

```
static void BeeGame.Items.Bee.ConvertToQueen (
    ref Bee princess,
    NormalBee drone )  [static]
```

Will Convert this bee into a BeeType.QUEEN Bee

Parameters

<i>princess</i>	The BeeType.PRINCESS Stats
<i>drone</i>	The BeeType.DRONE

Definition at line 165 of file [Bee.cs](#).

```
00166      {
00167          princess.beeType = BeeType.QUEEN;
00168          princess.queenBee = new QueenBee(princess.normalBee, drone);
00169          princess.normalBee = null;
00170
00171          princess.itemName = new CultureInfo("en-US", false).TextInfo.ToTitleCase($""
00172 {princess.queenBee.queen.pSpecies} {princess.beeType}" .ToLower());
00172 }
```

6.8.3.3 GetGameObject()

```
override GameObject BeeGame.Items.Bee.GetGameObject ()  [virtual]
```

Returns the GameObject for the item of it has one

Returns

GameObject for the item

Reimplemented from [BeeGame.Items.Item](#).

Definition at line 130 of file [Bee.cs](#).

```
00131      {
00132          var go = PrefabDictionary.GetPrefab("Bee");
00133
00134          go.GetComponent<SetBeeGOColours>().colour = BeeDictionaries.
00134 GetBeeColour(normalBee?.pSpecies ?? queenBee.
00134 queen.pSpecies);
00135          go.GetComponent<SetBeeGOColours>().beeType = beeType;
00136
00137          return go;
00138      }
```

6.8.3.4 GetHashCode()

```
override int BeeGame.Items.Bee.GetHashCode ( ) [virtual]
```

Returns the hashcode for this [Item](#)

Returns

9

Implements [BeeGame.Items.AbstractItem](#).

Definition at line 215 of file [Bee.cs](#).

```
00216     {
00217         return ID;
00218     }
```

6.8.3.5 GetItemID()

```
override string BeeGame.Items.Bee.GetItemID ( ) [virtual]
```

Makes the item ID. For this it is the Normal ID \ the int value of the queenBee.GetHashCode() or normalBee.GetHashCode() as a string

Returns

[Item](#) ID as a string

Implements [BeeGame.Items.AbstractItem](#).

Definition at line 144 of file [Bee.cs](#).

```
00145     {
00146         return $"{GetHashCode () }\\{(int)beeType}{queenBee?.GetHashCode () ?? normalBee?.GetHashCode () }";
00147     }
```

6.8.3.6 GetItemSprite()

```
override Sprite BeeGame.Items.Bee.GetItemSprite () [virtual]
```

Returns the sprite for this, of the correct colour

Returns

Sprite

Reimplemented from [BeeGame.Items.Item](#).

Definition at line 99 of file [Bee.cs](#).

```
00100      {
00101          /* if the bee has not change in any way dont rebuild the sprite as that takes time
00102          if(previousBeeType == beeType && itemSprite != null)
00103          {
00104              return itemSprite;
00105          }
00106
00107          previousBeeType = beeType;
00108
00109          /* set the correct sprite and colour
00110          if (beeType == BeeType.QUEEN)
00111          {
00112              /* avoids the crown, black body, yellow body, and both colours of the wings
00113              Color[] colorsToAvoid = { new Color(0, 0, 0), new Color(232f, 200f, 42f, 255f) / 255f, new
00114              Color(232f, 213f, 106f, 255f) / 255f, new Color(156f, 146f, 130f, 255f) / 255f, new Color(225f, 223f, 219f,
00115              255f) / 255f };
00116              return itemSprite = SpriteDictionary.
00117                  GetSprite("Queen").ColourSprite(BeeDictionaries.
00118                  GetBeeColour((BeeSpecies)(queenBee?.queen.
00119                      pSpecies)), coloursToAvoid: colorsToAvoid);
00120          }
00121          else if (beeType == BeeType.PRINCESS)
00122          {
00123              /* avoids the tiara, black body, yellow body, and both colours of the wings
00124              Color[] colorsToAvoid = { new Color(0, 0, 0), new Color(191f, 195f, 45f, 255f) / 255f, new
00125              Color(191f, 195f, 44f, 255f) / 255f, new Color(156f, 146f, 130f, 255f) / 255f, new Color(225f, 223f, 219f, 2
00126              55f) / 255f, new Color(232f, 200, 42, 255f) / 255f };
00127              return itemSprite = SpriteDictionary.
00128                  GetSprite("Princess").ColourSprite(BeeDictionaries.
00129                  GetBeeColour((BeeSpecies)(normalBee?.pSpecies)), coloursToAvoid:
00130                      colorsToAvoid);
00131          }
00132          else
00133          {
00134              /* avoids the block body, yellow body, and both wing colours
00135              Color[] colorsToAvoid = { new Color(0, 0, 0), new Color(156f, 146f, 130f, 255f) / 255f, new
00136              Color(225f, 223f, 219f, 255f) / 255f, new Color(232f, 200, 42, 255f) / 255f };
00137              return itemSprite = SpriteDictionary.
00138                  GetSprite("Drone").ColourSprite(BeeDictionaries.
00139                  GetBeeColour((BeeSpecies)normalBee?.pSpecies), coloursToAvoid:
00140                      colorsToAvoid);
00141          }
00142      }
```

6.8.3.7 MakeBeeWithStats()

```
Bee BeeGame.Items.Bee.MakeBeeWithStats (
    BeeType beeType = BeeType.DRONE,
    BeeSpecies species = BeeSpecies.FOREST,
    BeeLifeSpan lifespan = BeeLifeSpan.NORMAL,
    uint fertility = 2,
    BeeEffect effect = BeeEffect.NONE,
    BeeProductionSpeed prodSpeed = BeeProductionSpeed.NORMAL )
```

Make a bee with given stats

Parameters

<i>beeType</i>	BeeType
<i>species</i>	BeeSpecies
<i>lifespan</i>	BeeLifeSpan
<i>fertility</i>	1 or greater
<i>effect</i>	BeeEffect
<i>prodSpeed</i>	BeeProductionSpeed

Returns

A Bee with the given stats

Definition at line 184 of file [Bee.cs](#).

```

00185     {
00186         NormalBee normBee = new NormalBee()
00187     {
00188         pSpecies = species,
00189         pLifespan = lifespan,
00190         pFertility = fertility,
00191         pProdSpeed = prodSpeed,
00192         pEffect = effect,
00193         sEffect = effect,
00194         sFertility = fertility,
00195         sLifespan = lifespan,
00196         sProdSpeed = prodSpeed,
00197         sSpecies = species
00198     };
00199
00200     switch (beeType)
00201     {
00202         case BeeType.QUEEN:
00203             return new Bee(beeType, new QueenBee(normBee, normBee));
00204         default:
00205             return new Bee(beeType, normBee);
00206     }
00207 }
```

6.8.4 Member Data Documentation

6.8.4.1 canSeeBeeData

```
bool BeeGame.Items.Bee.canSeeBeeData = false
```

Can all of the bee data be seen when hovered over?

Definition at line 20 of file [Bee.cs](#).

6.8.4.2 ID

```
new int BeeGame.Items.Bee.ID => 11 [static]
```

Definition at line 54 of file [Bee.cs](#).

6.8.4.3 itemSprite

```
Sprite BeeGame.Items.Bee.itemSprite [private]
```

This bees Sprite

Definition at line [40](#) of file [Bee.cs](#).

6.8.4.4 maxStack

```
int BeeGame.Items.Bee.maxStack = 64 [private]
```

Definition at line [34](#) of file [Bee.cs](#).

6.8.5 Property Documentation

6.8.5.1 beeType

```
BeeType BeeGame.Items.Bee.beeType [get], [set]
```

This bees BeeType

Definition at line [25](#) of file [Bee.cs](#).

6.8.5.2 maxStackCount

```
override int BeeGame.Items.Bee.maxStackCount [get]
```

Overrided so can be set

Definition at line [33](#) of file [Bee.cs](#).

6.8.5.3 normalBee

```
NormalBee BeeGame.Items.Bee.normalBee [get], [set]
```

If this bee is not a BeeType.QUEEN this will be not null

Definition at line [52](#) of file [Bee.cs](#).

6.8.5.4 previousBeeType

`BeeType BeeGame.Items.Bee.previousBeeType [get], [set], [private]`

What was this bees BeeType?

Definition at line 29 of file [Bee.cs](#).

6.8.5.5 queenBee

`QueenBee BeeGame.Items.Bee.queenBee [get], [set]`

If this bee is a BeeType.QUEEN this will be not null

Possibly change this to an array to 2 [NormalBees](#)

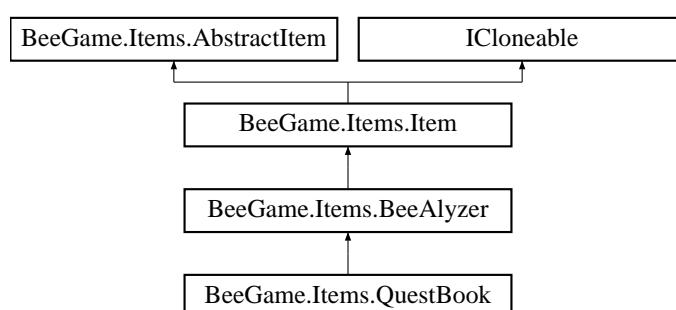
Definition at line 48 of file [Bee.cs](#).

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/[Bee.cs](#)

6.9 BeeGame.Items.BeeAlyzer Class Reference

Inheritance diagram for BeeGame.Items.BeeAlyzer:



Public Member Functions

- `BeeAlyzer ()`
- `BeeAlyzer (string s)`

Used of another object wants to use this as a parent class
- `virtual void OpenItemInventory (Inventory.Inventory playerInventory=null)`

opens and closes the items inventory
- `override void InteractWithItem (Inventory.Inventory playerInventory)`

Tells the rest of the game how to interact with this item
- `override bool InteractWithObject ()`

this object can be interacted with
- `override Sprite GetItemSprite ()`

Returns the items Sprite
- `override string GetItemID ()`

Returns the items ID
- `override int GetHashCode ()`

Public Attributes

- new int **ID** = 13
Item ID
- GameObject **myInventory**
Inventory this item is attached to

Properties

- override int **maxStackCount** [get]
1
- override bool **placeable** [get]
False

Additional Inherited Members

6.9.1 Detailed Description

Definition at line 9 of file [BeeAlyzer.cs](#).

6.9.2 Constructor & Destructor Documentation

6.9.2.1 BeeAlyzer() [1/2]

BeeGame.Items.BeeAlyzer.BeeAlyzer ()

Definition at line 47 of file [BeeAlyzer.cs](#).

```
00047             : base("BeeAlyzer")
00048         {
00049     }
```

6.9.2.2 BeeAlyzer() [2/2]

BeeGame.Items.BeeAlyzer.BeeAlyzer (
 string s)

Used of another object wants to use this as a parent class

Parameters

s	Name of child object
---	----------------------

Definition at line 55 of file [BeeAlyzer.cs](#).

```
00055           : base(s)
00056       { }
```

6.9.3 Member Function Documentation

6.9.3.1 GetHashCode()

override int BeeGame.Items.BeeAlyzer.GetHashCode () [virtual]

Implements [BeeGame.Items.AbstractItem](#).

Reimplemented in [BeeGame.Items.QuestBook](#).

Definition at line 121 of file [BeeAlyzer.cs](#).

```
00122     {
00123         return ID;
00124     }
```

6.9.3.2 GetItemID()

override string BeeGame.Items.BeeAlyzer.GetItemID () [virtual]

Returns the items [ID](#)

Returns

Implements [BeeGame.Items.AbstractItem](#).

Definition at line 114 of file [BeeAlyzer.cs](#).

```
00115     {
00116         return $"{GetHashCode()}";
00117     }
```

6.9.3.3 GetItemSprite()

override Sprite BeeGame.Items.BeeAlyzer.GetItemSprite () [virtual]

Returns the items [Sprite](#)

Returns

Reimplemented from [BeeGame.Items.Item](#).

Reimplemented in [BeeGame.Items.QuestBook](#).

Definition at line 105 of file [BeeAlyzer.cs](#).

```
00106     {
00107         return SpriteDictionary.GetSprite("BeeAlyzer");
00108     }
```

6.9.3.4 InteractWithItem()

```
override void BeeGame.Items.BeeAlyzer.InteractWithItem (
    Inventory.Inventory playerInventory )
```

Tells the rest of the game how to interact with this item

Parameters

<i>playerInventory</i>	
------------------------	--

Definition at line 87 of file [BeeAlyzer.cs](#).

```
00088      {
00089          OpenItemInvnetory(playerInventory);
00090      }
```

6.9.3.5 InteractWithObject()

```
override bool BeeGame.Items.BeeAlyzer.InteractWithObject () [virtual]
```

this object can be intereacted with

Returns

Reimplemented from [BeeGame.Items.Item](#).

Definition at line 96 of file [BeeAlyzer.cs](#).

```
00097      {
00098          return true;
00099      }
```

6.9.3.6 OpenItemInvnetory()

```
virtual void BeeGame.Items.BeeAlyzer.OpenItemInvnetory (
    Inventory.Inventory playerInventory = null ) [virtual]
```

opens and closes the items inventory

Parameters

<i>playerInventory</i>	Used when opening inventory to give the new inventory the players inventory(Inventory.Player_Inventory.PlayerInventory)
------------------------	---

Reimplemented in [BeeGame.Items.QuestBook](#).

Definition at line 64 of file [BeeAlyzer.cs](#).

```
00065      {
00066          if (myInventory == null)
00067          {
```

```
00068     //* makes the inventory
00069     myInventory = (GameObject)UnityEngine.Object.Instantiate(
00070     UnityEngine.Resources.Load("Prefabs/BeeAlyzerInventory"));
00071     /* opens the inventory and gives it the players inventory
00072     myInventory.GetComponent<BeeAlyzerInventory>().ToggleInventory
00073     (playerInventory);
00074     myInventory.GetComponent<BeeAlyzerInventory>().myItem = this;
00075     }
00076     {
00077         myInventory = null;
00078     }
00079 }
```

6.9.4 Member Data Documentation

6.9.4.1 ID

```
new int BeeGame.Items.BeeAlyzer.ID = 13
```

[Item ID](#)

Definition at line [15](#) of file [BeeAlyzer.cs](#).

6.9.4.2 myInventory

```
GameObject BeeGame.Items.BeeAlyzer.myInventory
```

[Inventory](#) this item is attached to

Definition at line [43](#) of file [BeeAlyzer.cs](#).

6.9.5 Property Documentation

6.9.5.1 maxStackCount

```
override int BeeGame.Items.BeeAlyzer.maxStackCount [get]
```

1

Definition at line [21](#) of file [BeeAlyzer.cs](#).

6.9.5.2 placeable

`override bool BeeGame.Items.BeeAlyzer.placeable [get]`

`False`

Definition at line 32 of file [BeeAlyzer.cs](#).

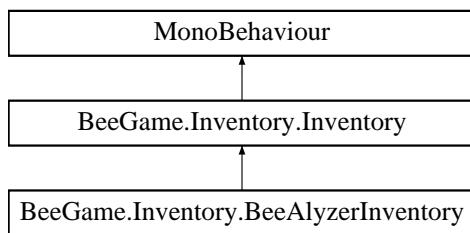
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/[BeeAlyzer.cs](#)

6.10 BeeGame.Inventory.BeeAlyzerInventory Class Reference

Incentory for the chests

Inheritance diagram for BeeGame.Inventory.BeeAlyzerInventory:



Public Member Functions

- `override void SavelInv ()`
This Inventory should not be saved
- `override void ToggleInventory (Inventory inv)`
Opens and closes this inventory
- `virtual void DropItemsFromInventory ()`
Removes the items from the inventory when it is closed, so they are not destroyed
- `void CheckForBeeAndHoney ()`
checks if a Bees and Honey(currently disabled) are in the correct slots

Public Attributes

- Text `infoText`
Text box that shows the bee data

Protected Member Functions

- `new void Update ()`

Package Attributes

- **BeeAlyzer myItem**
Item that this is attached to

Private Member Functions

- string **ReturnData (Bee b)**
Returns the formatted bee data
- void **SetPlayerItems ()**
Applies the players inventory to this inventory
- void **ApplyPlayerItems ()**
Applies this inventory to the player once it is closed

Private Attributes

- **Inventory playerInventory**
The players inventory

Additional Inherited Members

6.10.1 Detailed Description

Incentory for the chests

Definition at line 13 of file [BeeAlyzerInventory.cs](#).

6.10.2 Member Function Documentation

6.10.2.1 ApplyPlayerItems()

```
void BeeGame.Inventory.BeeAlyzerInventory.ApplyPlayerItems ( ) [private]
```

Applies this inventory to the player once it is closed

Definition at line 177 of file [BeeAlyzerInventory.cs](#).

```
00178      {
00179          for (int i = 0; i < playerInventory.items.
00180              itemsInInventory.Length; i++)
00181          {
00182              playerInventory.items.itemsInInventory[i] =
00183                  items.itemsInInventory[i + (slots.Length - 36)];
00184          }
00185      playerInventory.SaveInv();
00186  }
```

6.10.2.2 CheckForBeeAndHoney()

```
void BeeGame.Inventory.BeeAlyzerInventory.CheckForBeeAndHoney ( )
```

checks if a Bees and Honey(currently disabled) are in the correct slots

Definition at line 118 of file [BeeAlyzerInventory.cs](#).

```
00119      {
00120          if(slots[0].item == null)
00121          {
00122              //if (slots[1].item.GetHashCode() == Honey.ID && slots[1].item.itemStackCount >= 1)
00123              if (slots[2].item is Bee b)
00124              {
00125                  b.canSeeBeeData = true;
00126                  items.itemsInInventory[0] = slots[2].
00127                  item;
00128                  items.itemsInInventory[2] = null;
00129                  //items.itemsInInventory[1].itemStackCount -= 1;
00130                  infoText.text = ReturnData(b);
00131              }
00132              else
00133              {
00134                  infoText.text = "";
00135              }
00136          }
00137      }
```

6.10.2.3 DroItemsFromInventory()

```
virtual void BeeGame.Inventory.BeeAlyzerInventory.DropItemsFromInventory ( ) [virtual]
```

Removes the items from the inventory when it is closed, so they are not destroyed

Definition at line 95 of file [BeeAlyzerInventory.cs](#).

```
00096      {
00097          /* looks at every item in the crafting grid
00098          for (int i = 0; i < 3; i++)
00099          {
00100              if (items.itemsInInventory[i] != null)
00101              {
00102                  /* spawns it and removes it from the inventory if an items exists within
00103                  itemStackCount; j++)
00104                  {
00105                      MonoBehaviour.print(GameObject.FindGameObjectWithTag("Player").transform.position);
00106                      items.itemsInInventory[i].SpawnItem(
00107                          THVector3.GameObject.FindGameObjectWithTag("Player").transform.position + new
00108                          THVector3(0, 1, 0));
00109                  }
00110              }
00111      }
```

6.10.2.4 ReturnData()

```
string BeeGame.Inventory.BeeAlyzerInventory.ReturnData (
    Bee b ) [private]
```

Returns the formatted bee data

Parameters

<i>b</i>	
----------	--

Returns

Definition at line 144 of file [BeeAlyzerInventory.cs](#).

```

00145      {
00146          string returnString = "";
00147
00148          if (b.beeType == Core.Enums.BeeType.QUEEN)
00149          {
00150
00151          }
00152
00153          /* calls to check if a pure bread bee quest should be called
00154          QuestEvents.CallPureBeeCraftedEvent(b.
normalBee.pSpecies, b.normalBee.sSpecies);
00155
00156          returnString += $"Primary Species: {b.normalBee.pSpecies}\nSecondary Species:
{b.normalBee.sSpecies}\nPrimary Fertility: {b.normalBee.pFertility}\nSecondary Fertility: {b.normalBee.sFertility}\n
Lifespan: {b.normalBee.pLifespan}\nSecondary Lifespan: {b.normalBee.sLifespan}\nPrimary Production Speed:
{b.normalBee.pProdSpeed}\nSecondary Production Speed: {b.normalBee.sProdSpeed}";
00157
00158          return returnString;
00159      }

```

6.10.2.5 SaveInv()

```
override void BeeGame.Inventory.BeeAlyzerInventory.SaveInv () [virtual]
```

This [Inventory](#) should not be saved

Reimplemented from [BeeGame.Inventory.Inventory](#).

Definition at line 43 of file [BeeAlyzerInventory.cs](#).

```

00044      {
00045          return;
00046      }

```

6.10.2.6 SetPlayerItems()

```
void BeeGame.Inventory.BeeAlyzerInventory.SetPlayerItems () [private]
```

Applies the players inventory to this inventory

Definition at line 166 of file [BeeAlyzerInventory.cs](#).

```

00167      {
00168          for (int i = 0; i < playerInventory.items.
itemsInInventory.Length; i++)
00169          {
00170              items.itemsInInventory[i + (slots.Length - 36)] =
playerInventory.items.itemsInInventory[i];
00171          }
00172      }

```

6.10.2.7 ToggleInventory()

```
override void BeeGame.Inventory.BeeAlyzerInventory.ToggleInventory (
    Inventory inv ) [virtual]
```

Opens and closes this inventory

Parameters

<i>inv</i>	
------------	--

Reimplemented from [BeeGame.Inventory.Inventory](#).

Definition at line 53 of file [BeeAlyzerInventory.cs](#).

```

00054      {
00055          thisInventoryOpen = !thisInventoryOpen;
00056
00057          isAnotherInventoryOpen = thisInventoryOpen;
00058
00059          if (this.gameObject.activeInHierarchy && !thisInventoryOpen)
00060          {
00061              chestOpen = false;
00062
00063              /* removes all of the items from thsi inventory
00064              DropItemsFromInventory();
00065
00066              /* tells item that inventory has been closed
00067              myItem.OpenItemInvnetory();
00068              Cursor.lockState = CursorLockMode.Locked;
00069              Cursor.visible = false;
00070              /* applies the chanegs to the players inventory
00071              ApplyPlayerItems();
00072              /* destroys this as it is not needed
00073              Destroy(this.gameObject);
00074          }
00075      else
00076      {
00077          chestOpen = true;
00078
00079          SetInventorySize(slots.Length);
00080
00081          playerInventory = inv;
00082
00083          SetPlayerItems();
00084
00085          PutItemsInSlots();
00086          /* hides and locks the cursor
00087          Cursor.lockState = CursorLockMode.None;
00088          Cursor.visible = true;
00089      }
00090  }
```

6.10.2.8 Update()

```
new void BeeGame.Inventory.BeeAlyzerInventory.Update () [protected]
```

Definition at line 30 of file [BeeAlyzerInventory.cs](#).

```

00031      {
00032          if (GetButtonDown("Close Menu/Inventory"))
00033              ToggleInventory(playerInventory);
00034          UpdateBase();
00035          PutItemsInSlots();
00036
00037          CheckForBeeAndHoney();
00038      }
```

6.10.3 Member Data Documentation

6.10.3.1 infoText

`Text BeeGame.Inventory.BeeAlyzerInventory.infoText`

Text box that shows the bee data

Definition at line 19 of file [BeeAlyzerInventory.cs](#).

6.10.3.2 myItem

`BeeAlyzer BeeGame.Inventory.BeeAlyzerInventory.myItem [package]`

Item that this is attached to

Definition at line 27 of file [BeeAlyzerInventory.cs](#).

6.10.3.3 playerInventory

`Inventory BeeGame.Inventory.BeeAlyzerInventory.playerInventory [private]`

The players inventory

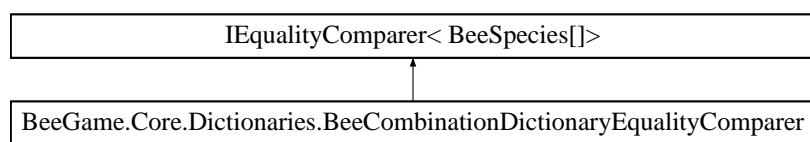
Definition at line 23 of file [BeeAlyzerInventory.cs](#).

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/ItemInventory/BeeAlyzerInventory.cs

6.11 BeeGame.Core.Dictionaries.BeeCombinationDictionaryEqualityComparer Class Reference

Inheritance diagram for BeeGame.Core.Dictionaries.BeeCombinationDictionaryEqualityComparer:



Public Member Functions

- bool [Equals \(BeeSpecies\[\] x, BeeSpecies\[\] y\)](#)
- int [GetHashCode \(BeeSpecies\[\] obj\)](#)

6.11.1 Detailed Description

Definition at line 9 of file [EqualityComperors.cs](#).

6.11.2 Member Function Documentation

6.11.2.1 Equals()

```
bool BeeGame.Core.Dictionaries.BeeCombinationDictionaryEqualityComparer.Equals (
    BeeSpecies [] x,
    BeeSpecies [] y )
```

Definition at line 11 of file [EqualityComperors.cs](#).

```
00012      {
00013          if (x.Contains(y[0]) && x.Contains(y[1]))
00014          {
00015              /* if the x length is greater than 2 this means that the combination can have duplicate
00016              bees for a product
00017              if (x.Length > 2)
00018                  return true;
00019
00020              /* if 1 means both y elements are the same so no combination has been found
00021              if(y.Intersect(x).Count() <= 1)
00022                  return false;
00023
00024          }
00025
00026          return false;
00027      }
```

6.11.2.2 GetHashCode()

```
int BeeGame.Core.Dictionaries.BeeCombinationDictionaryEqualityComparer.GetHashCode (
    BeeSpecies [] obj )
```

Definition at line 29 of file [EqualityComperors.cs](#).

```
00030      {
00031          unchecked
00032          {
00033              int hashcode = 13;
00034
00035              for (int i = 0; i < obj.Length; i++)
00036              {
00037                  hashcode += (int)obj[i];
00038              }
00039
00040              return hashcode;
00041          }
00042      }
```

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionaries/[EqualityComperors.cs](#)

6.12 BeeGame.Core.Dictionaries.BeeDictionaries Class Reference

Static Public Member Functions

- static float [] GetWeights (BeeSpecies[] species)
- static BeeSpecies [] GetCombinations (BeeSpecies s1, BeeSpecies s2)
- static Items.Item [] GetBeeProduce (BeeSpecies species)
- static Color GetBeeColour (BeeSpecies species)
- static Color GetCombColour (HoneyCombType type)

Returns colour if the given honey comb

Static Public Attributes

- static Dictionary< BeeSpecies[], BeeSpecies[]> beeCombinations

Static Private Member Functions

- static Color CombColour (float r, float g, float b, float a=255f)

Makes a new colour given Red, r , Green, g , Blue, b , optionaly an Alpha, a . Ranging from 0f-255f

Static Private Attributes

- static Dictionary< BeeSpecies, float > beeCombinationWeights
- static Dictionary< BeeSpecies, Items.Item[]> beeProduce
- static Dictionary< BeeSpecies, Color > beeColour
- static Dictionary< HoneyCombType, Color > honeyCombColour

The colour of the BeeGame.Items.HoneyComb for each of the HoneyCombTypes

6.12.1 Detailed Description

Definition at line 8 of file BeeDictionaries.cs.

6.12.2 Member Function Documentation

6.12.2.1 CombColour()

```
static Color BeeGame.Core.Dictionaries.BeeDictionaries.CombColour (
    float r,
    float g,
    float b,
    float a = 255f) [static], [private]
```

Makes a new colour given Red, r , Green, g , Blue, b , optionaly an Alpha, a . Ranging from 0f-255f

Parameters

<i>r</i>	Red
<i>g</i>	Green
<i>b</i>	Blue
<i>a</i>	Alpha, Default no alpha

Returns

new Color made with the given r, g, b values

Definition at line 117 of file [BeeDictionaries.cs](#).

```
00118      {
00119          return new Color(r / 255f, g / 255f, b / 255f);
00120      }
```

6.12.2.2 GetBeeColour()

```
static Color BeeGame.Core.Dictionaries.BeeDictionaries.GetBeeColour (
    BeeSpecies species ) [static]
```

Definition at line 90 of file [BeeDictionaries.cs](#).

```
00091      {
00092          beeColour.TryGetValue(species, out Color colour);
00093          return colour != null ? colour : new Color();
00094      }
```

6.12.2.3 GetBeeProduce()

```
static Items.Item [ ] BeeGame.Core.Dictionaries.BeeDictionaries.GetBeeProduce (
    BeeSpecies species ) [static]
```

Definition at line 74 of file [BeeDictionaries.cs](#).

```
00075      {
00076          beeProduce.TryGetValue(species, out Items.Item[] produce);
00077          /* if the produce can't be found, then return a honey comb as it is probably a bug
00078          return produce ?? new Items.Item[1] { new Items.HoneyComb(
00079              HoneyCombType.HONEY) };
00080      }
```

6.12.2.4 GetCombColour()

```
static Color BeeGame.Core.Dictionaries.BeeDictionaries.GetCombColour (
    HoneyCombType type ) [static]
```

Returns colour if the given honey coumb

Parameters

<i>type</i>	Type of the comb
-------------	------------------

Returns

The Color of the comb and a new Color.red if the given HoneyCombType does not exists as a key in the [honeyCombColour](#) dictionary

Definition at line 127 of file [BeeDictionaries.cs](#).

```
00128      {
00129          honeyCombColour.TryGetValue(type, out var temp);
00130
00131          if (temp == null)
00132              return new Color(1, 0, 0);
00133
00134          return temp;
00135      }
```

6.12.2.5 GetCombinations()

```
static BeeSpecies [] BeeGame.Core.Dictionaries.GetCombinations (
    BeeSpecies s1,
    BeeSpecies s2 ) [static]
```

Definition at line 39 of file [BeeDictionaries.cs](#).

```
00040      {
00041          var beeSpecies = new BeeSpecies[2] { s1, s2 };
00042          var returnBeeList = new List<BeeSpecies>();
00043
00044          var keys = beeCombinations.Keys.ToArray();
00045          var comparor = new BeeCombinationDictionaryEqualityComparer();
00046
00047          for (int i = 0; i < keys.Length; i++)
00048          {
00049              if (comparor.Equals(keys[i], beeSpecies))
00050              {
00051                  var temp = beeCombinations[keys[i]];
00052
00053                  for (int j = 0; j < temp.Length; j++)
00054                  {
00055                      returnBeeList.Add(temp[i]);
00056                  }
00057              }
00058          }
00059
00060          returnBeeList.Add(s1);
00061          returnBeeList.Add(s2);
00062
00063          return returnBeeList.ToArray();
00064      }
```

6.12.2.6 GetWeights()

```
static float [] BeeGame.Core.Dictionaries.BeeDictionaries.GetWeights (
    BeeSpecies [] species) [static]
```

Definition at line 17 of file [BeeDictionaries.cs](#).

```
00018      {
00019          var returnArray = new float[species.Length];
00020
00021          for (int i = 0; i < species.Length; i++)
00022          {
00023              if(beeCombinationWeights.ContainsKey(species[i]))
00024                  returnArray[i] = beeCombinationWeights[species[i]];
00025              else
00026                  returnArray[i] = 0.5f;
00027          }
00028
00029          return returnArray;
00030      }
```

6.12.3 Member Data Documentation

6.12.3.1 beeColour

```
Dictionary<BeeSpecies, Color> BeeGame.Core.Dictionaries.beeColour [static],
[private]
```

Initial value:

```
= new Dictionary<BeeSpecies, Color>()
{
    {BeeSpecies.FOREST, CombColour(0, 255, 0)},
    {BeeSpecies.COMMON, CombColour(255, 0, 0)}
}
```

Definition at line 84 of file [BeeDictionaries.cs](#).

6.12.3.2 beeCombinations

```
Dictionary<BeeSpecies[], BeeSpecies[]> BeeGame.Core.Dictionaries.beeCombinations
[static]
```

Initial value:

```
= new Dictionary<BeeSpecies[], BeeSpecies[]>(new BeeCombinationDictionaryEqualityComparer())
{
    { new BeeSpecies[6] { BeeSpecies.FOREST,
        BeeSpecies.MEADOWS, BeeSpecies.TROPICAL, BeeSpecies.WINTRY,
        BeeSpecies.MODEST, BeeSpecies.MARSHY }, new BeeSpecies[1] {
        BeeSpecies.COMMON } }
}
```

Definition at line 34 of file [BeeDictionaries.cs](#).

6.12.3.3 beeCombinationWeights

```
Dictionary<BeeSpecies, float> BeeGame.Core.Dictionaries.BeeDictionaries.beeCombinationWeights
[static], [private]
```

Initial value:

```
= new Dictionary<BeeSpecies, float>()
{
    {BeeSpecies.COMMON, 0.15f },
    {BeeSpecies.HEROIC, 0.06f }
}
```

Definition at line 11 of file [BeeDictionaries.cs](#).

6.12.3.4 beeProduce

```
Dictionary<BeeSpecies, Items.Item[]> BeeGame.Core.Dictionaries.BeeDictionaries.beeProduce
[static], [private]
```

Initial value:

```
= new Dictionary<BeeSpecies, Items.Item[]>()
{
    {BeeSpecies.FOREST, new Items.Item[]{new Items.HoneyComb(
        HoneyCombType.HONEY) } },
    {BeeSpecies.COMMON, new Items.Item[]{new Items.HoneyComb(
        HoneyCombType.HONEY) } }
}
```

Definition at line 68 of file [BeeDictionaries.cs](#).

6.12.3.5 honeyCoubmColour

```
Dictionary<HoneyCombType, Color> BeeGame.Core.Dictionaries.BeeDictionaries.honeyCoubmColour
[static], [private]
```

Initial value:

```
= new Dictionary<HoneyCombType, Color>()
{
    {HoneyCombType.HONEY, CombColour(255, 164, 56) },
    {HoneyCombType.ICEY, CombColour(78, 231, 231) }
}
```

The colour of the [BeeGame.Items.HoneyComb](#) for each of the HoneyCombTypes

Definition at line 102 of file [BeeDictionaries.cs](#).

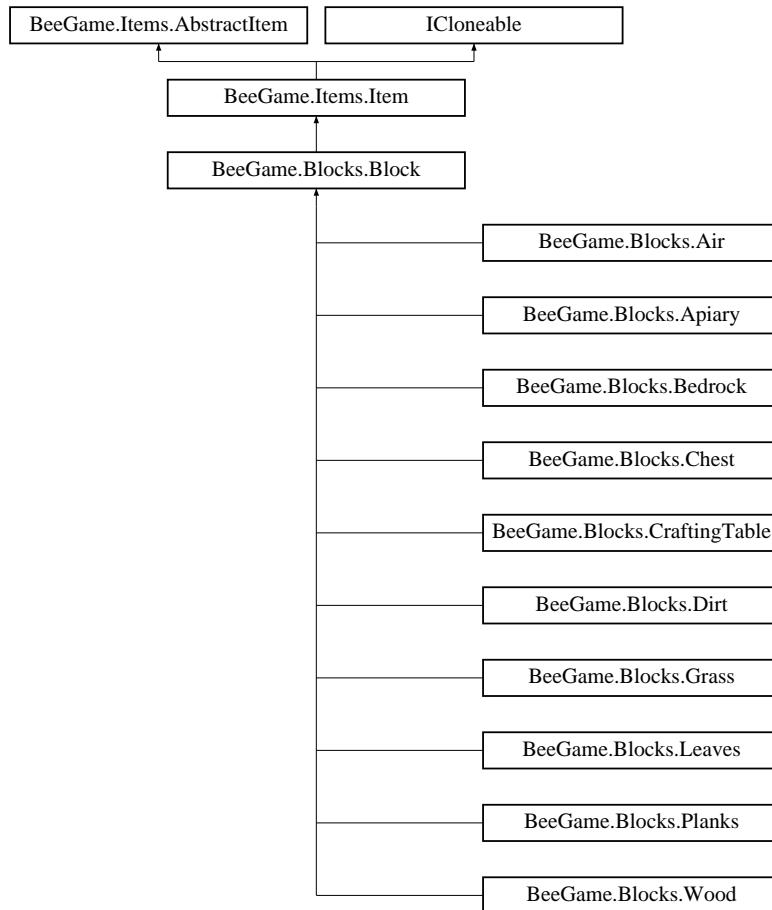
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/[BeeDictionaries.cs](#)

6.13 BeeGame.Blocks.Block Class Reference

Base class for blocks

Inheritance diagram for BeeGame.Blocks.Block:



Public Member Functions

- **Block ()**
Constructor sets the Item.placeable to true
- **Block (string name)**
Sets placeable to true and sets name of the block/item
- override Sprite **GetItemSprite ()**
Returns the sprite for the item
- virtual void **BreakBlock (THVector3 pos)**
Spawns an item with the same texture as the broken block
- virtual void **UpdateBlock (int x, int y, int z, Chunk chunk)**
Should this Block be updated when the mesh is made?
- virtual bool **InteractWithBlock (BeeGame.Inventory.Inventory inv)**
Can this block be interacted with?
- virtual **MeshData BlockData (Chunk chunk, int x, int y, int z, MeshData meshData, bool addToRender
Mesh=true)**
The data that this block adds to the mesh
- virtual bool **IsSolid (Direction direction)**

What Directions is this [Block](#) solid in?

- `override int GetHashCode ()`
Has code for the [Block](#)
- `override string ToString ()`
Returns the [Block](#) name and Id formatted nicely

Public Attributes

- `bool breakable = true`
Can this [Block](#) be broken?
- `bool changed = true`
Has this block been placed by the player?
- `override bool placeable => true`
Sets so that blocks can be placed?

Static Public Attributes

- `static new int ID = 1`

Additional Inherited Members

6.13.1 Detailed Description

Base class for blocks

Definition at line 14 of file [Block.cs](#).

6.13.2 Constructor & Destructor Documentation

6.13.2.1 Block() [1/2]

`BeeGame.Blocks.Block.Block ()`

Constructor sets the `Item.placeable` to true

Definition at line 36 of file [Block.cs](#).

```
00036             : base ()
00037         {
00038             itemName = "Stone";
00039         }
```

6.13.2.2 Block() [2/2]

```
BeeGame.Blocks.Block.Block (
    string name )
```

Sets `placeable` to true and sets name of the block/item

Parameters

<i>name</i>	Name of the block/item
-------------	------------------------

Definition at line 45 of file [Block.cs](#).

```
00045           : base(name)
00046     {
00047 }
```

6.13.3 Member Function Documentation

6.13.3.1 BlockData()

```
virtual MeshData BeeGame.Blocks.Block.BlockData (
    Chunk chunk,
    int x,
    int y,
    int z,
    MeshData meshData,
    bool addToRenderMesh = true ) [virtual]
```

The data that this block adds to the mesh

Parameters

<i>chunk</i>	Chunk the block is in
<i>x</i>	X pos of the block
<i>y</i>	Y pos of the block
<i>z</i>	Z pos of the block
<i>meshData</i>	meshdata to add to
<i>addToRenderMesh</i>	should the block also be added to the render mesh not just the collision mesh?

Returns

Given *meshData* with this blocks data added to it

If no data of either collider or render should be added override to return the given mesh
If only collision data should be added override to say render mesh false

Reimplemented in [BeeGame.Blocks.CraftingTable](#), [BeeGame.Blocks.Chest](#), [BeeGame.Blocks.Apiary](#), and [BeeGame.Blocks.Air](#).

Definition at line 102 of file [Block.cs](#).

```
00103     {
00104         /* Adds the Top face of the block
00105         if (!chunk.GetBlock(x, y + 1, z, false).IsSolid(Direction.DOWN))
00106         {
00107             meshData = FaceDataUp(x, y, z, meshData, addToRenderMesh);
00108         }
00109     }
```

```

00109
00110     /* Adds the Bottom face of the block
00111     if (!chunk.GetBlock(x, y - 1, z, false).IsSolid(Direction.UP))
00112     {
00113         meshData = FaceDataDown(x, y, z, meshData, addToRenderMesh);
00114     }
00115
00116     /* Adds the North face of the block
00117     if (!chunk.GetBlock(x, y, z + 1, false).IsSolid(Direction.SOUTH))
00118     {
00119         meshData = FaceDataNorth(x, y, z, meshData, addToRenderMesh);
00120     }
00121
00122     /* Adds the South face of the block
00123     if (!chunk.GetBlock(x, y, z - 1, false).IsSolid(Direction.NORTH))
00124     {
00125         meshData = FaceDataSouth(x, y, z, meshData, addToRenderMesh);
00126     }
00127
00128     /* Adds the East face of the block
00129     if (!chunk.GetBlock(x + 1, y, z, false).IsSolid(Direction.WEST))
00130     {
00131         meshData = FaceDataEast(x, y, z, meshData, addToRenderMesh);
00132     }
00133
00134     /* Adds the West face of the block
00135     if (!chunk.GetBlock(x - 1, y, z, false).IsSolid(Direction.EAST))
00136     {
00137         meshData = FaceDataWest(x, y, z, meshData, addToRenderMesh);
00138     }
00139
00140     return meshData;
00141 }
```

6.13.3.2 BreakBlock()

```
virtual void BeeGame.Blocks.Block.BreakBlock (
    THVector3 pos ) [virtual]
```

Spawns an item with the same texture as the broken block

Parameters

<i>pos</i>	position to spawn the Item
------------	----------------------------

Reimplemented in [BeeGame.Blocks.CraftingTable](#), [BeeGame.Blocks.Chest](#), [BeeGame.Blocks.Apiary](#), [BeeGame.Blocks.Bedrock](#), and [BeeGame.Blocks.Air](#).

Definition at line 62 of file [Block.cs](#).

```

00063     {
00064         GameObject go = Object.Instantiate(UnityEngine.Resources.Load("
00065             Prefabs/ItemGameObject") as GameObject, pos, Quaternion.identity) as GameObject;
00066         go.GetComponent<ItemGameObject>().item = this;
00067     }
```

6.13.3.3 GetHashCode()

```
override int BeeGame.Blocks.Block.GetHashCode () [virtual]
```

Has code for the Block

Returns

1

Implements [BeeGame.Items.AbstractItem](#).

Reimplemented in [BeeGame.Blocks.CraftingTable](#), [BeeGame.Blocks.Chest](#), [BeeGame.Blocks.Grass](#), [BeeGame.Blocks.Dirt](#), [BeeGame.Blocks.Planks](#), [BeeGame.Blocks.Leaves](#), and [BeeGame.Blocks.Wood](#).

Definition at line 159 of file [Block.cs](#).

```
00160     {
00161         return ID;
00162     }
```

6.13.3.4 GetItemSprite()

```
override Sprite BeeGame.Blocks.Block.GetItemSprite ( ) [virtual]
```

Returns the sprite for the item

Returns

Sprite for this item

Reimplemented from [BeeGame.Items.Item](#).

Reimplemented in [BeeGame.Blocks.CraftingTable](#), [BeeGame.Blocks.Chest](#), [BeeGame.Blocks.Grass](#), [BeeGame.Blocks.Dirt](#), [BeeGame.Blocks.Planks](#), [BeeGame.Blocks.Wood](#), and [BeeGame.Blocks.Leaves](#).

Definition at line 51 of file [Block.cs](#).

```
00052     {
00053         return SpriteDictionary.GetSprite("Stone");
00054     }
```

6.13.3.5 InteractWithBlock()

```
virtual bool BeeGame.Blocks.Block.InteractWithBlock (
    BeeGame.Inventory.Inventory inv) [virtual]
```

Can this block be interacted with?

Returns

False by default

Definition at line 81 of file [Block.cs](#).

```
00082     {
00083         return false;
00084     }
```

6.13.3.6 IsSolid()

```
virtual bool BeeGame.Blocks.Block.IsSolid (
    Direction direction) [virtual]
```

What Directions is this [Block](#) solid in?

Parameters

<i>direction</i>	Direction to check
------------------	--------------------

Returns

Default returns true for all sides

Reimplemented in [BeeGame.Blocks.CraftingTable](#), [BeeGame.Blocks.Air](#), and [BeeGame.Blocks.Leaves](#).

Definition at line 148 of file [Block.cs](#).

```
00149     {
00150         return true;
00151     }
```

6.13.3.7 ToString()

```
override string BeeGame.Blocks.Block.ToString ()
```

Returns the [Block](#) name and Id formatted nicely

Returns

Definition at line 168 of file [Block.cs](#).

```
00169     {
00170         return $"{itemName} \nID: {GetHashCode()}";
00171     }
```

6.13.3.8 UpdateBlock()

```
virtual void BeeGame.Blocks.Block.UpdateBlock (
    int x,
    int y,
    int z,
    Chunk chunk ) [virtual]
```

Should this [Block](#) be updated when the mesh is made?

Parameters

<i>x</i>	X pos if the block
<i>y</i>	Y pos of the block
<i>z</i>	Z pos of the block
<i>chunk</i>	Chunk that the block is in

Reimplemented in [BeeGame.Blocks.Grass](#).

Definition at line [75](#) of file [Block.cs](#).

```
00075 { }
```

6.13.4 Member Data Documentation

6.13.4.1 breakable

```
bool BeeGame.Blocks.Block.breakable = true
```

Can this [Block](#) be broken?

Definition at line [21](#) of file [Block.cs](#).

6.13.4.2 changed

```
bool BeeGame.Blocks.Block.changed = true
```

Has this block been placed by the player?

Definition at line [25](#) of file [Block.cs](#).

6.13.4.3 ID

```
new int BeeGame.Blocks.Block.ID = 1 [static]
```

Definition at line [17](#) of file [Block.cs](#).

6.13.4.4 placeable

```
override bool BeeGame.Blocks.Block.placeable => true
```

Sets so that blocks can be placed?

Definition at line [29](#) of file [Block.cs](#).

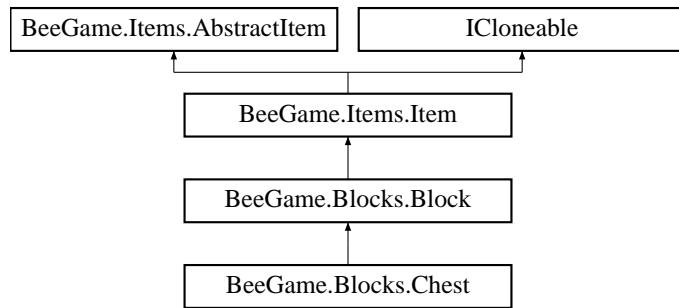
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/[Block.cs](#)

6.14 BeeGame.Blocks.Chest Class Reference

Chest Block

Inheritance diagram for BeeGame.Blocks.Chest:



Public Member Functions

- **Chest ()**
Makes a new chest from a parameterless constructor
- override GameObject **GetGameObject ()**
Gets the gme object for this chest
- override Tile TexturePosition (**Direction** direction)
Returns the texture for the chest Block
- override MeshData **BlockData** (Chunk chunk, int x, int y, int z, MeshData meshData, bool addToRender ← Mesh=true)
The data that this block adds to the mesh
- override void **BreakBlock** (THVector3 pos)
Breaks the block
- override Sprite **GetItemSprite ()**
Gets the Chest Sprite
- override bool **InteractWithBlock** (BeeGame.Inventory.Inventory inv)
Opens the ChestInventory when clicked on
- override int **GetHashCode ()**
Gets the ID of the Block
- override string **ToString ()**
Returns the Block name and ID formatted nicely

Static Public Attributes

- static new int **ID** => 8

Private Attributes

- GameObject **myGameobject**
Chest model for when it is placed

Additional Inherited Members

6.14.1 Detailed Description

Chest Block

Definition at line 16 of file [Chest.cs](#).

6.14.2 Constructor & Destructor Documentation

6.14.2.1 Chest()

```
BeeGame.Blocks.Chest.Chest ( )
```

Makes a new chest from a parameterless constructor

Definition at line 32 of file [Chest.cs](#).

```
00032             : base ("Chest")
00033     {
00034         usesGameObject = true;
00035     }
```

6.14.3 Member Function Documentation

6.14.3.1 BlockData()

```
override MeshData BeeGame.Blocks.Chest.BlockData (
    Chunk chunk,
    int x,
    int y,
    int z,
    MeshData meshData,
    bool addToRenderMesh = true ) [virtual]
```

The data that this block adds to the mesh

Parameters

<i>chunk</i>	Chunk the block is in
<i>x</i>	X pos of the block
<i>y</i>	Y pos of the block
<i>z</i>	Z pos of the block
<i>meshData</i>	meshdata to add to chunk
<i>addToRenderMesh</i>	should the block also be added to the render mesh not just the collision mesh

Returns

Given *meshData* with this blocks data added to it

Only adds to the collision mesh as the model is handled by the unity prefab system

Reimplemented from [BeeGame.Blocks.Block](#).

Definition at line 74 of file [Chest.cs](#).

```
00075      {
00076          if (myGameObject == null)
00077          {
00078              myGameObject = UnityEngine.Object.Instantiate(
00079                  PrefabDictionary.GetPrefab("Chest"), new THVector3(x, y, z) + chunk.
00080                  chunkWorldPos, Quaternion.identity, chunk.transform);
00081              myGameObject.GetComponent<ChestInventory>().inventoryPosition =
00082                  new THVector3(x, y, z) + chunk.chunkWorldPos;
00083              myGameObject.GetComponent<ChestInventory>().SetChestInventory();
00084          }
00085      }
00086      return base.BlockData(chunk, x, y, z, meshData, true);
00087  }
```

6.14.3.2 BreakBlock()

```
override void BeeGame.Blocks.Chest.BreakBlock (
    THVector3 pos ) [virtual]
```

Breaks the block

Parameters

<i>pos</i>	Position of the block
------------	-----------------------

Reimplemented from [BeeGame.Blocks.Block](#).

Definition at line 89 of file [Chest.cs](#).

```
00090      {
00091          /* removes the blocks blocks inventory save file and destroys the game object
00092          Serialization.Serialization.DeleteFile(myGameObject.GetComponent<
00093              ChestInventory>().inventoryName);
00094          UnityEngine.Object.Destroy(myGameObject);
00095          /* removes the collision mesh from the chunk
00096          base.BreakBlock(pos);
00097      }
```

6.14.3.3 GetGameObject()

```
override GameObject BeeGame.Blocks.Chest.GetGameObject ( ) [virtual]
```

Gets the gme object for this chest

Returns

The chest game object

Reimplemented from [BeeGame.Items.Item](#).

Definition at line 43 of file [Chest.cs](#).

```
00044      {
00045          return PrefabDictionary.GetPrefab("Chest");
00046      }
```

6.14.3.4 GetHashCode()

```
override int BeeGame.Blocks.Chest.GetHashCode () [virtual]
```

Gets the ID of the [Block](#)

Returns

8

Reimplemented from [BeeGame.Blocks.Block](#).

Definition at line 126 of file [Chest.cs](#).

```
00127      {
00128          return ID;
00129      }
```

6.14.3.5 GetItemSprite()

```
override Sprite BeeGame.Blocks.Chest.GetItemSprite () [virtual]
```

Gets the [Chest](#) Sprite

Returns

The [Chest](#) Sprite

Reimplemented from [BeeGame.Blocks.Block](#).

Definition at line 102 of file [Chest.cs](#).

```
00103      {
00104          return SpriteDictionary.GetSprite("Chest");
00105      }
```

6.14.3.6 InteractWithBlock()

```
override bool BeeGame.Blocks.Chest.InteractWithBlock (
    BeeGame.Inventory.Inventory inv)
```

Opens the ChestInventory when clicked on

Parameters

<i>inv</i>	Inventory that the chest is interacting with
------------	--

Returns

true

Definition at line 114 of file [Chest.cs](#).

```
00115      {
00116          myGameObject.GetComponent<ChestInventory>().ToggleInventory(inv);
00117          return true;
00118      }
```

6.14.3.7 TexturePosition()

```
override Tile BeeGame.Blocks.Chest.TexturePosition (
    Direction direction ) [virtual]
```

Returns the texture for the chest [Block](#)**Parameters**

<i>direction</i>	Direction of the desired face
------------------	-------------------------------

Returns[Tile](#) with the texture coordinates of the [Block](#) texture

Returns a transparent texture as the chest model already has a texture applied

Reimplemented from [BeeGame.Items.Item](#).Definition at line 56 of file [Chest.cs](#).

```
00057      {
00058          return new Tile() { x = 0, y = 9 };
00059      }
```

6.14.3.8 ToString()

```
override string BeeGame.Blocks.Chest.ToString ( )
```

Returns the [Block](#) name and ID formatted nicely**Returns**Definition at line 135 of file [Chest.cs](#).

```
00136      {
00137          return $"{itemName}\nID{GetItemID()}";
00138      }
```

6.14.4 Member Data Documentation

6.14.4.1 ID

```
new int BeeGame.Blocks.Chest.ID => 8 [static]
```

Definition at line 25 of file [Chest.cs](#).

6.14.4.2 myGameObject

```
GameObject BeeGame.Blocks.Chest.myGameObject [private]
```

[Chest](#) model for when it is placed

Definition at line 23 of file [Chest.cs](#).

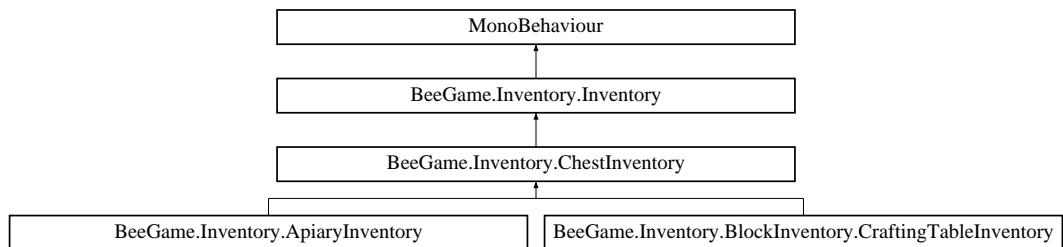
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/[Chest.cs](#)

6.15 BeeGame.Inventory.ChestInventory Class Reference

Incentory for the chests

Inheritance diagram for BeeGame.Inventory.ChestInventory:



Public Member Functions

- void [UpdateChestInventory \(\)](#)
The unity Update method is not called if the class is is child...annoyingly
- virtual void [SetChestInventory \(string invName="Chest"\)](#)
Sets the Size and name of this [Inventory](#)
- override void [ToggleInventory \(\[Inventory\]\(#\) inv\)](#)
Opens and closes the inventory

Public Attributes

- **THVector3 inventoryPosition**
Position in worldspace of the chest
- **Inventory playerinventory**
Reference to the players [Inventory](#) so that it can be updated when chest is closed
- **GameObject inventory**
The inventory gameobject that will be displayed
- **int inventorySize**
How many slots are in this [Inventory](#)

Private Member Functions

- **void Update ()**
Updates the slots and checks if the inventory should be closed
- **void SetPlayerItems ()**
Puts the player items into the chest
- **void ApplyPlayerItems ()**
Applies the changes made to the [playerinventory](#) in this

Additional Inherited Members

6.15.1 Detailed Description

Incentory for the chests

Definition at line 11 of file [ChestInventory.cs](#).

6.15.2 Member Function Documentation

6.15.2.1 ApplyPlayerItems()

```
void BeeGame.Inventory.ChestInventory.ApplyPlayerItems ( ) [private]
```

Applies the changes made to the [playerinventory](#) in this

Definition at line 88 of file [ChestInventory.cs](#).

```
00089      {
00090          for (int i = 0; i < playerinventory.items.
00091              itemsInInventory.Length; i++)
00092          {
00093              playerinventory.items.itemsInInventory[i] =
00094                  items.itemsInInventory[i + (inventorySize - 36)];
00095          }
00096      playerinventory.SaveInv();
```

6.15.2.2 SetChestInventory()

```
virtual void BeeGame.Inventory.ChestInventory.SetChestInventory (
    string invName = "Chest" ) [virtual]
```

Sets the Size and name of this [Inventory](#)

Reimplemented in [BeeGame.Inventory.BlockInventory.CraftingTableInventory](#), and [BeeGame.Inventory.ApiaryInventory](#).

Definition at line 60 of file [ChestInventory.cs](#).

```
00061      {
00062          SetInventorySize(inventorySize);
00063          /* sets the UI to not be seen as inventories cannot start open
00064          inventory.SetActive(false);
00065
00066          /* sets the name and position if this inventory used during serialization and deserialization
00067          inventoryName = $"{invName} @ {(ChunkWorldPos)inventoryPosition}";
00068
00069          /* loads the inventory if it had had items put in it last time it existed
00070          Serialization.DeserializeInventory(this, inventoryName);
00071      }
```

6.15.2.3 SetPlayerItems()

```
void BeeGame.Inventory.ChestInventory.SetPlayerItems ( ) [private]
```

Puts the player items into the chest

Definition at line 77 of file [ChestInventory.cs](#).

```
00078      {
00079          for (int i = 0; i < playerinventory.items.
00080              itemsInInventory.Length; i++)
00081          {
00082              items.itemsInInventory[i + (inventorySize - 36)] =
00083                  playerinventory.items.itemsInInventory[i];
00084          }
00085      }
```

6.15.2.4 ToggleInventory()

```
override void BeeGame.Inventory.ChestInventory.ToggleInventory (
    Inventory inv ) [virtual]
```

Opens and closes the inventory

Parameters

<i>inv</i>	
------------	--

Reimplemented from [BeeGame.Inventory.Inventory](#).

Reimplemented in [BeeGame.Inventory.BlockInventory.CraftingTableInventory](#).

Definition at line 103 of file [ChestInventory.cs](#).

```

00104      {
00105          /* sets the player inventory
00106          playerinventory = inv;
00107
00108          thisInventoryOpen = !thisInventoryOpen;
00109
00110          isAnotherInventoryOpen = thisInventoryOpen;
00111
00112          inventory.SetActive(!inventory.activeInHierarchy);
00113
00114          if (inventory.activeInHierarchy)
00115          {
00116              chestOpen = true;
00117
00118              /* stops the player inventory from being opened immidiately after this is closed
00119              blockInventoryJustClosed = true;
00120              SetPlayerItems();
00121              /* hides and locks the cursor
00122              Cursor.lockState = CursorLockMode.None;
00123              Cursor.visible = true;
00124          }
00125          else
00126          {
00127              chestOpen = false;
00128
00129              /* puts the items into the chest
00130              /* shows and unlocks the cursor
00131              ApplyPlayerItems();
00132              Cursor.lockState = CursorLockMode.Locked;
00133              Cursor.visible = false;
00134          }
00135      }

```

6.15.2.5 Update()

```
void BeeGame.Inventory.ChestInventory.Update () [private]
```

Updates the slots and checks if the inventory should be closed

Definition at line 37 of file [ChestInventory.cs](#).

```

00038      {
00039          UpdateChestInventory();
00040      }

```

6.15.2.6 UpdateChestInventory()

```
void BeeGame.Inventory.ChestInventory.UpdateChestInventory ()
```

The unity Update method is not called if the class is is child...annoyingly

Definition at line 45 of file [ChestInventory.cs](#).

```

00046      {
00047          /* the chest should always have a player inventory when it does this but checks just in case
00048          if (playerinventory != null)
00049              UpdateBase();
00050
00051          /* checks if the inventory should be closed
00052          if (GetButtonDown("Player Inventory") && thisInventoryOpen &&
00053              floatingItem == null)
00054              ToggleInventory(playerinventory);
00055      }

```

6.15.3 Member Data Documentation

6.15.3.1 inventory

`GameObject BeeGame.Inventory.ChestInventory.inventory`

The inventory gameobject that will be displayed

Definition at line 25 of file [ChestInventory.cs](#).

6.15.3.2 inventoryPosition

`THVector3 BeeGame.Inventory.ChestInventory.inventoryPosition`

Position in worldspace of the chest

Definition at line 17 of file [ChestInventory.cs](#).

6.15.3.3 inventorySize

`int BeeGame.Inventory.ChestInventory.inventorySize`

How many slots are in this [Inventory](#)

Definition at line 30 of file [ChestInventory.cs](#).

6.15.3.4 playerinventory

`Inventory BeeGame.Inventory.ChestInventory.playerinventory`

Refernce to the players [Inventory](#) so that it can be updated when chest is closed

Definition at line 21 of file [ChestInventory.cs](#).

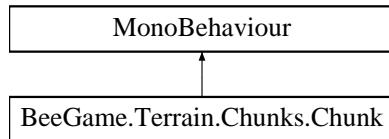
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/BlockInventory/[ChestInventory.cs](#)

6.16 BeeGame.Terrain.Chunks.Chunk Class Reference

A section of land for the game, used so that land can be generated in parts and not all at once

Inheritance diagram for BeeGame.Terrain.Chunks.Chunk:



Public Member Functions

- **Block GetBlock** (int x, int y, int z, bool checkNebouringChunks=true)
Returns the Block in the given x, y, z
- void **SetBlock** (int x, int y, int z, **Block** block, bool checkNebouringChunks=true)
Sets a Block in the given position
- void **SetBlocksUnmodified** ()
Sets all of the Blocks in the blocks array to unmodified so that the whole chunk is not saved when it does not need to be

Static Public Member Functions

- static bool **InRange** (int i)
Checks that a given value is within the Chunk

Public Attributes

- **Block [,,] blocks** = new **Block**[chunkSize, chunkSize, chunkSize]
All of the Blocks in the Chunk
- bool **update** = true
Should the Chunk be updated?
- bool **rendered**
Is the Chunk rendered?
- bool **updateCollisionMesh** = false
Should the chunks collision mesh be updated?
- bool **applyCollisionMesh** = false
Should the collision mesh be applied
- **World world**
World that this chunk is in as MonoBehaviours cannot be static this is for convenience
- **ChunkWorldPos chunkWorldPos**
Chunks position in the world as a ChunkWorldPos (int verson of Core.THVector3)

Static Public Attributes

- static int **chunkSize** = 16
Size of the Chunk

Private Member Functions

- void [Start \(\)](#)
Sets the `meshCollider` and `filter` variables
- void [Update \(\)](#)
Checks if the `Chunk` should be updated
- void [UpdateChunk \(\)](#)
Updates the `mesh` for the `Chunk`
- void [RenderMesh \(MeshData meshData\)](#)
Renders the given `MeshData` into a unity Mesh
- void [ColliderMesh \(\)](#)
Makes a collision mesh from the `mesh`

Private Attributes

- `MeshData mesh = new MeshData()`
`MeshData` of this chunk
- `MeshFilter filter`
This `Chunks` mesh filter
- `MeshCollider meshCollider`
This `Chunks` mesh colldier

6.16.1 Detailed Description

A section of land for the game, used so that land can be generated in parts and not all at once

Definition at line [14](#) of file [Chunk.cs](#).

6.16.2 Member Function Documentation

6.16.2.1 ColliderMesh()

```
void BeeGame.Terrain.Chunks.Chunk.ColliderMesh ( ) [private]
```

Makes a collision mesh from the `mesh`

Definition at line [237](#) of file [Chunk.cs](#).

```
00238         {
00239             /* if the chunk has been told to update the collisions but the chunk has ne verts dont do it as
00240             * their is no point
00241             if (this.mesh.verts.Count == 0)
00242                 return;
00243
00244             /* if the render and collision meshes should be shared set the render mesh to the collision
00245             * mesh otherwise make a collision mesh
00246             if (this.mesh.shareMeshes)
00247             {
00248                 world.chunkHasMadeCollisionMesh = true;
00249                 applyCollisionMesh = false;
00250                 meshCollider.sharedMesh = filter.mesh;
00251                 return;
00252             }
00253         }
```

```

00251         world.chunkHasMadeCollisionMesh = true;
00252         /* Applying the mesh takes the longest but nothing can be done with the mesh class in a
00253            secondary thread...thanks Unity
00254
00255         /* makes a new mesh setting the name for convenience
00256         Mesh mesh = new Mesh()
00257     {
00258         name = "Collider Mesh",
00259         vertices = this.mesh.colVerts.ToArray(),
00260         triangles = this.mesh.colTris.ToArray()
00261     };
00262
00263     /* recalcs the normals and applies the mesh
00264     mesh.RecalculateNormals();
00265
00266     meshCollider.sharedMesh = mesh;
00267
00268     applyCollisionMesh = false;
00269 }
```

6.16.2.2 GetBlock()

```
Block BeeGame.Terrain.Chunks.Chunk.GetBlock (
    int x,
    int y,
    int z,
    bool checkNebouringChunks = true )
```

Returns the Block in the given x, y, z

Parameters

x	X pos if the Block
y	Z pos if the Block
z	Y pos if the Block
checkNebouringChunks	Should this check neighbouring chunks? Only set to false when chunk <code>mesh</code> is being built for performance

Returns

Block at given x, y, z

Definition at line 123 of file [Chunk.cs](#).

```

00124     {
00125         /* checks that block is in the chunk
00126         if (InRange(x) && InRange(y) && InRange(z))
00127             return blocks[x, y, z];
00128
00129         /* if the block is not in the chunk and we should check other chunks do that, otherwise return
00130            an air block (empty block)
00131         //if(checkNebouringChunks)
00132             return world.GetBlock(chunkWorldPos.x + x,
00133             chunkWorldPos.y + y, chunkWorldPos.z + z);
00132
00133         //return new Air();
00134     }
```

6.16.2.3 InRange()

```
static bool BeeGame.Terrain.Chunks.Chunk.InRange (
    int i ) [static]
```

Checks that a given value is within the [Chunk](#)

Parameters

<i>i</i>	Value to check
----------	----------------

Returns

true if the value is in the [Chunk](#)

Definition at line 162 of file [Chunk.cs](#).

```
00163      {
00164          /* if the value is less than 0 or greater than 16 the value is outside the chunk
00165          if (i < 0 || i >= chunkSize)
00166              return false;
00167          return true;
00168      }
```

6.16.2.4 RenderMesh()

```
void BeeGame.Terrain.Chunks.Chunk.RenderMesh (
    MeshData meshData ) [private]
```

Renders the given [MeshData](#) into a unity Mesh

Parameters

<i>meshData</i>	Mesh data to render
-----------------	---------------------

Definition at line 213 of file [Chunk.cs](#).

```
00214      {
00215          /* Applying the mesh takes the longest but nothing can be done with the mesh class in a
00216          secondary thread...thanks unity
00217          mesh.done = false;
00218          /* clears the current chunk mesh
00219          filter.mesh.Clear();
00220          /* name for convenience
00221          filter.mesh.name = "Render Mesh";
00222          /* puts the tris and verts from the meshdata into the chunk mesh
00223          filter.mesh.vertices = meshData_verts.ToArray();
00224          filter.mesh.triangles = meshData_tris.ToArray();
00225
00226          /* sets the uvs
00227          filter.mesh.uv = meshData_uv.ToArray();
00228
00229          /* redoing the normals incase they got messed up
00230          filter.mesh.RecalculateNormals();
00231          /* is this necessary as it causes a lot of lag?
00232      }
```

6.16.2.5 SetBlock()

```
void BeeGame.Terrain.Chunks.Chunk.SetBlock (
    int x,
    int y,
    int z,
    Block block,
    bool checkNebouringChunks = true )
```

Sets a Block in the given position

Parameters

<i>x</i>	X pos of the Block
<i>y</i>	Y pos of the Block
<i>z</i>	Z pos of the Block
<i>block</i>	Block to set

Definition at line 143 of file [Chunk.cs](#).

```
00144     {
00145         /* sets the block in the position if it is in the chunk, then return early
00146         if (InRange(x) && InRange(y) && InRange(z))
00147         {
00148             blocks[x, y, z] = block;
00149             return;
00150         }
00151
00152         if (checkNebouringChunks)
00153             /* if the block is not in the chunk find its chunk and set it their
00154             world.SetBlock(chunkWorldPos.x + x,
00155             chunkWorldPos.y + y, chunkWorldPos.z + z, block);
00155     }
```

6.16.2.6 SetBlocksUnmodified()

```
void BeeGame.Terrain.Chunks.Chunk.SetBlocksUnmodified ( )
```

Sets all of the Blocks in the `blocks` array to unmodified so that the whole chunk is not saved when it does not need to be

A modified Block is a Block removed or added by the player

Definition at line 178 of file [Chunk.cs](#).

```
00179     {
00180         foreach (var block in blocks)
00181         {
00182             block.changed = false;
00183         }
00184     }
```

6.16.2.7 Start()

```
void BeeGame.Terrain.Chunks.Chunk.Start ( ) [private]
```

Sets the `meshCollider` and `filter` variables

Definition at line 77 of file [Chunk.cs](#).

```
00078      {
00079          filter = GetComponent<MeshFilter>();
00080          meshCollider = GetComponent<MeshCollider>();
00081      }
```

6.16.2.8 Update()

```
void BeeGame.Terrain.Chunks.Chunk.Update ( ) [private]
```

Checks if the `Chunk` should be updated

Definition at line 86 of file [Chunk.cs](#).

```
00087      {
00088          lock(mesh)
00089          {
00090              if (update)
00091              {
00092                  update = false;
00093                  updateCollisionMesh = true;
00094                  mesh = new MeshData();
00095                  /* Enabling threading here works in editor but not in build?
00096                  /* ok whatever...
00097                  /* Thread thread = new Thread(UpdateChunk);
00098
00099                  /* thread.Start();
00100                  UpdateChunk();
00101          }
00102
00103          if (mesh.done && mesh != new MeshData())
00104          {
00105              RenderMesh(mesh);
00106          }
00107
00108          if (applyCollisionMesh)
00109              ColliderMesh();
00110      }
00111  }
```

6.16.2.9 UpdateChunk()

```
void BeeGame.Terrain.Chunks.Chunk.UpdateChunk ( ) [private]
```

Updates the `mesh` for the `Chunk`

Definition at line 189 of file [Chunk.cs](#).

```
00190      {
00191          /* says that this chunk is rendered and initializes the mesh
00192          rendered = true;
00193
00194          /* goes through every block in the blocks array getting their mesh data
00195          for (int x = 0; x < chunkSize; x++)
00196          {
00197              for (int z = 0; z < chunkSize; z++)
00198              {
00199                  for (int y = 0; y < chunkSize; y++)
00200                  {
00201                      blocks[x, y, z]?.UpdateBlock(x, y, z, this);
00202                      mesh = blocks[x, y, z]?.BlockData(this, x, y, z,
00203                      mesh) ?? mesh;
00204                  }
00205              }
00206              mesh.done = true;
00207      }
```

6.16.3 Member Data Documentation

6.16.3.1 applyCollisionMesh

```
bool BeeGame.Terrain.Chunks.Chunk.applyCollisionMesh = false
```

Should the collision mesh be applied

Definition at line 47 of file [Chunk.cs](#).

6.16.3.2 blocks

```
Block[,] BeeGame.Terrain.Chunks.Chunk.blocks = new Block[chunkSize, chunkSize, chunkSize]
```

All of the Blocks in the [Chunk](#)

Definition at line 29 of file [Chunk.cs](#).

6.16.3.3 chunkSize

```
int BeeGame.Terrain.Chunks.Chunk.chunkSize = 16 [static]
```

Size of the [Chunk](#)

Same size for x, y, z

Possibly some place has 16 hard coded as reducing the number breaks things TODO: find

Definition at line 24 of file [Chunk.cs](#).

6.16.3.4 chunkWorldPos

```
ChunkWorldPos BeeGame.Terrain.Chunks.Chunk.chunkWorldPos
```

[Chunks](#) position in the world as a [ChunkWorldPos](#) (int version of [Core.THVector3](#))

Definition at line 56 of file [Chunk.cs](#).

6.16.3.5 filter

```
MeshFilter BeeGame.Terrain.Chunks.Chunk.filter [private]
```

This [Chunks](#) mesh filter

Definition at line 66 of file [Chunk.cs](#).

6.16.3.6 mesh

```
MeshData BeeGame.Terrain.Chunks.Chunk.mesh = new MeshData() [private]
```

MeshData of this chunk

Definition at line 61 of file [Chunk.cs](#).

6.16.3.7 meshCollider

```
MeshCollider BeeGame.Terrain.Chunks.Chunk.meshCollider [private]
```

This [Chunks](#) mesh colldier

Definition at line 70 of file [Chunk.cs](#).

6.16.3.8 rendered

```
bool BeeGame.Terrain.Chunks.Chunk.rendered
```

Is the [Chunk](#) rendered?

Definition at line 38 of file [Chunk.cs](#).

6.16.3.9 update

```
bool BeeGame.Terrain.Chunks.Chunk.update = true
```

Should the [Chunk](#) be updated?

Definition at line 34 of file [Chunk.cs](#).

6.16.3.10 updateCollisionMesh

```
bool BeeGame.Terrain.Chunks.Chunk.updateCollsionMesh = false
```

Should the chunks collision mesh be updated?

Definition at line 43 of file [Chunk.cs](#).

6.16.3.11 world

`World` BeeGame.Terrain.Chunks.Chunk.world

World that this chunk is in as MonoBehaviours cannot be static this is for convenience

Definition at line 52 of file [Chunk.cs](#).

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/[Chunk.cs](#)

6.17 BeeGame.Terrain.ChunkWorldPos Struct Reference

Serializable int version of THVector3

Public Member Functions

- `ChunkWorldPos (int x, int y, int z)`
Constructor so that values can be input on creation of the vector
- `override string ToString ()`
Formats the values nicely incase it is needed
- `override bool Equals (object obj)`
- `override int GetHashCode ()`
Makes a unique hascode for the vector

Static Public Member Functions

- `static implicit operator THVector3 (ChunkWorldPos pos)`
Converts a `ChunkWorldPos` to a `THVector3` without the need for an explicit cast as no data will be lost
- `static operator ChunkWorldPos (THVector3 pos)`
Converts a `ChunkWorldPos` to a `THVector3`

Public Attributes

- `int x`
x, y, z values for the vector
- `int y`
- `int z`

6.17.1 Detailed Description

Serializable int version of THVector3

Definition at line 10 of file [ChunkWorldPos.cs](#).

6.17.2 Constructor & Destructor Documentation

6.17.2.1 ChunkWorldPos()

```
BeeGame.Terrain.ChunkWorldPos.ChunkWorldPos (
    int x,
    int y,
    int z )
```

Constructor so that values can be input on creation of the vector

Parameters

x	X Value
y	Y Value
z	Z Value

Definition at line 23 of file [ChunkWorldPos.cs](#).

```
00024     {
00025         this.x = x;
00026         this.y = y;
00027         this.z = z;
00028     }
```

6.17.3 Member Function Documentation

6.17.3.1 Equals()

```
override bool BeeGame.Terrain.ChunkWorldPos.Equals (
    object obj )
```

Definition at line 41 of file [ChunkWorldPos.cs](#).

```
00042     {
00043         /* possibly remove and just check if obj is null
00044         if (!(obj is ChunkWorldPos))
00045             return false;
00046
00047         ChunkWorldPos temp = (ChunkWorldPos)obj;
00048
00049         /* possibly change to hashCode checking
00050         if (temp.x == x && temp.y == y && temp.z == z)
00051             return true;
00052
00053         return false;
00054     }
```

6.17.3.2 GetHashCode()

```
override int BeeGame.Terrain.ChunkWorldPos.GetHashCode ( )
```

Makes a unique hascode for the vector

Returns

unique int value for the vector

Possible that 2 different values can give the same hashCode but chance of that happening and the vectors needing to be checked against each other is low

Definition at line 63 of file [ChunkWorldPos.cs](#).

```
00064     {
00065         unchecked
00066         {
00067             int hashCode = 47;
00068
00069             hashCode *= 227 + x.GetHashCode();
00070             hashCode *= 227 + y.GetHashCode();
00071             hashCode *= 227 + z.GetHashCode();
00072
00073             return hashCode;
00074         }
00075     }
```

6.17.3.3 operator ChunkWorldPos()

```
static BeeGame.Terrain.ChunkWorldPos.operator ChunkWorldPos (
    THVector3 pos ) [explicit], [static]
```

Converts a [ChunkWorldPos](#) to a [THVector3](#)

Parameters

<i>pos</i>	A THVector3
------------	-----------------------------

Operator is explicit as data could be lost, [THVector3](#) is a float and [ChunkWorldPos](#) is a int

Definition at line 93 of file [ChunkWorldPos.cs](#).

```
00094      {
00095          return new ChunkWorldPos((int)pos.x, (int)pos.y, (int)pos.
00096          z);
00096      }
```

6.17.3.4 operator THVector3()

```
static implicit BeeGame.Terrain.ChunkWorldPos.operator THVector3 (
    ChunkWorldPos pos ) [static]
```

Converts a [ChunkWorldPos](#) to a [THVector3](#) without the need for an explicit cast as no data will be lost

Parameters

<i>pos</i>	this ChunkWorldPos
------------	------------------------------------

Definition at line 81 of file [ChunkWorldPos.cs](#).

```
00082      {
00083          return new THVector3(pos.x, pos.y, pos.z);
00084      }
```

6.17.3.5 ToString()

```
override string BeeGame.Terrain.ChunkWorldPos.ToString ( )
```

Formats the values nicely incase it is needed

Returns

Definition at line 34 of file [ChunkWorldPos.cs](#).

```
00035      {
00036          return ${"({x}, {y}, {z})"};
00037      }
```

6.17.4 Member Data Documentation

6.17.4.1 x

`int BeeGame.Terrain.ChunkWorldPos.x`

x, y, z values for the vector

Definition at line 15 of file [ChunkWorldPos.cs](#).

6.17.4.2 y

`int BeeGame.Terrain.ChunkWorldPos.y`

Definition at line 15 of file [ChunkWorldPos.cs](#).

6.17.4.3 z

`int BeeGame.Terrain.ChunkWorldPos.z`

Definition at line 15 of file [ChunkWorldPos.cs](#).

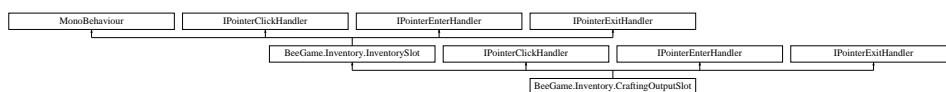
The documentation for this struct was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/[ChunkWorldPos.cs](#)

6.18 BeeGame.Inventory.CraftingOutputSlot Class Reference

Overrides the

Inheritance diagram for BeeGame.Inventory.CraftingOutputSlot:



Public Member Functions

- override void [OnPointerClick](#) (PointerEventData eventData)
Gives extra functionality to the base slot

Protected Member Functions

- new void [Update](#) ()
Updates the base slot things

Additional Inherited Members

6.18.1 Detailed Description

Overrides the

Definition at line 12 of file [CraftingOutputSlot.cs](#).

6.18.2 Member Function Documentation

6.18.2.1 OnPointerClick()

```
override void BeeGame.Inventory.CraftingOutputSlot.OnPointerClick (
    PointerEventData eventData) [virtual]
```

Gives extra functionality to the base slot

Parameters

<i>eventData</i>	
------------------	--

Reimplemented from [BeeGame.Inventory.InventorySlot](#).

Definition at line 27 of file [CraftingOutputSlot.cs](#).

```
00028      {
00029          /* recorded what item was in the slot before it is moved
00030          Item before = item;
00031
00032          base.OnPointerClick(eventData);
00033
00034          /* if the item is different now then the crafting result must have been removed so call the
00035          event
00036          if (before != item && before != null)
00037              ((CraftingTableInventory)myInventory).
00038          craftingResultRemoved.Invoke();
00039      }
```

6.18.2.2 Update()

```
new void BeeGame.Inventory.CraftingOutputSlot.Update () [protected]
```

Updates the base slot things

Definition at line 17 of file [CraftingOutputSlot.cs](#).

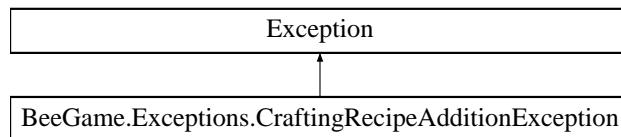
```
00018      {
00019          CheckItem();
00020          UpdateIcon();
00021      }
```

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/[CraftingOutputSlot.cs](#)

6.19 BeeGame.Exceptions.CraftingRecipeAdditionException Class Reference

Inheritance diagram for BeeGame.Exceptions.CraftingRecipeAdditionException:



Public Member Functions

- [CraftingRecipeAdditionException \(\)](#)
- [CraftingRecipeAdditionException \(string message\)](#)
- [CraftingRecipeAdditionException \(string message, Exception innerException\)](#)

6.19.1 Detailed Description

Definition at line 8 of file [CraftingRecipeAdditionException.cs](#).

6.19.2 Constructor & Destructor Documentation

6.19.2.1 CraftingRecipeAdditionException() [1/3]

`BeeGame.Exceptions.CraftingRecipeAdditionException.CraftingRecipeAdditionException ()`

Definition at line 10 of file [CraftingRecipeAdditionException.cs](#).

```

00010          : base()
00011      {
00012
00013      }
  
```

6.19.2.2 CraftingRecipeAdditionException() [2/3]

`BeeGame.Exceptions.CraftingRecipeAdditionException.CraftingRecipeAdditionException (
 string message)`

Definition at line 15 of file [CraftingRecipeAdditionException.cs](#).

```

00015          : base(message)
00016      {
00017
00018      }
  
```

6.19.2.3 CraftingRecipeAdditionException() [3 / 3]

```
BeeGame.Exceptions.CraftingRecipeAdditionException.CraftingRecipeAdditionException (
    string message,
    Exception innerException )
```

Definition at line 20 of file [CraftingRecipeAdditionException.cs](#).

```
00020     innerException)
00021     {
00022     }
: base (message,
```

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Exceptions/[CraftingRecipeAdditionException.cs](#)

6.20 BeeGame.Core.Dictionaries.CraftingRecipes Class Reference

Static Public Member Functions

- static void [AddShapedRecipie](#) (object[] reicpe, [Item](#) result)
Will add a shaped crafting recipe to the game
- static [Item](#) [GetShapedRecipieItem](#) (string recipe)
Returns an Item from the [shapedCraftingRecipes](#) dictionary
- static void [AddShapelessRecipie](#) (object[] recipe, [Item](#) result)
Adds a Shapeless recipe to the dictionary
- static string [GetShapelessRecipieString](#) ([Item](#)[] recipe)
Gets a shapeless recipe string from a given recipe
- static [Item](#) [GetShapelessRecipieResult](#) (int[] recipe)
Tries to get a shapeless recipe
- static [Item](#) [GetShapelessRecipieResult](#) (string recipe)
Tries to get a shapeless recipe
- static [Item](#) [GetShapelessRecipieResult](#) ([Item](#)[] recipe)
Tries to get a shapeless recipe

Static Private Attributes

- static Dictionary< string, [Item](#) > [shapedCraftingRecipes](#) = new Dictionary<string, [Item](#)>()
Contains all crafting recipes that require a certain layout in the crafting grid ([Blocks.CraftingTable](#))
- static Dictionary< string, [Item](#) > [shapelessRecipes](#)
All shapeless recipes

6.20.1 Detailed Description

Definition at line 10 of file [CraftingRecipes.cs](#).

6.20.2 Member Function Documentation

6.20.2.1 AddShapedRecipie()

```
static void BeeGame.Core.Dictionaries.CraftingRecipes.AddShapedRecipie (
    object [ ] reicpe,
    Item result ) [static]
```

Will add a shaped crafting recipe to the game

Parameters

<i>reicpe</i>	The desired recipe. Layout is {"XXX", "XXX", "XXX", "X", ItemID} where each X is a slot in the crafting grid, Each group of 3 is a row, and a "X", ItemID is the Item ID X represents (for each new item a new symbol is required), a Sapce is no item required in that slot
<i>result</i>	The Item that the recipe will produce

This example shows how to call AddShapedRecipie(object[], Item)

```
void Main()
{
    CraftingRecipies.AddShapedRecipie(new object[] { " X ", "X@X", " X ", "X", Wood.GetItemID(), "@", Stone
        .GetItemID() }, new Chest());
}
```

Definition at line 32 of file [CraftingRecipies.cs](#).

```
00033      {
00034          /* converts the given blocks of 3 characters to a 9 character string
00035          var stringRecipie = "";
00036
00037          for (int i = 0; i < 3; i++)
00038          {
00039              stringRecipie += reicpe[i] as string;
00040          }
00041
00042          /* gets what character represents which item
00043          for (int i = 3; i < reicpe.Length; i += 2)
00044          {
00045              var character = (string)reicpe[i];
00046              var itemID = (int)reicpe[i + 1];
00047
00048              /* replaces the character with the items id
00049              stringRecipie = stringRecipie.Replace(character, $"{itemID.ToString()}:");
00050          }
00051
00052          /* converts empty sets " " into "0:"
00053          stringRecipie = stringRecipie.Replace(" ", "0:");
00054
00055          /* if the recipe exists an exception is thrown as two recipies cannot be the same
00056          if (shapedCraftingRecipies.ContainsKey(stringRecipie))
00057              throw new CraftingRecipeAdditionException($"Shaped Recipe
already exists: {stringRecipie}");
00058
00059          result.itemStackCount = 1;
00060
00061          /* adds the recipe to the dictionary
00062          shapedCraftingRecipies.Add(stringRecipie, result);
00063      }
```

6.20.2.2 AddShaplessRecipie()

```
static void BeeGame.Core.Dictionaries.CraftingRecipies.AddShaplessRecipie (
    object [] recipe,
    Item result ) [static]
```

Adds a Shapeless recipe to the dictionary

Parameters

<i>recipe</i>	Recipe to add. Format as { Item, Number of items }
<i>result</i>	Result of the crafting recipe

2 Examples of adding a shapeless recipe

```
void Main()
{
    CraftingRecipes.AddShaplessRecipie(new object[] { new Dirt(), 2 }, new Grass());
}

void Main()
{
    CraftingRecipes.AddShaplessRecipie(new object[] { new Stone(), 3, new Wood(), 3 }, new Apiary());
}
```

Definition at line 111 of file [CraftingRecipes.cs](#).

```
00112     {
00113         var itemList = new List<int>();
00114         var stringRecpie = "";
00115
00116         for (int i = 0; i < recipe.Length; i+=2)
00117         {
00118             for (int j = 0; j < (int)recipe[i+1]; j++)
00119             {
00120                 itemList.Add(int.Parse(((Item)recipe[i]).GetItemID()));
00121             }
00122         }
00123
00124         itemList.Sort();
00125
00126         for (int i = 0; i < itemList.Count; i++)
00127         {
00128             stringRecpie += $"{itemList[i]}:";
00129         }
00130
00131         if (shaplessRecipes.ContainsKey(stringRecpie))
00132             throw new CraftingRecipeAdditionException($"Shaped Recipe
already exists: {stringRecpie}");
00133
00134         result.itemStackCount = 1;
00135         shaplessRecipes.Add(stringRecpie, result);
00136     }
```

6.20.2.3 GetShapedRecipeItem()

```
static Item BeeGame.Core.Dictionaries.CraftingRecipes.GetShapedRecipeItem (
    string recipe) [static]
```

Returns an Item from the `shapedCraftingRecipes` dictionary

Parameters

<code>recipe</code>	Recipe for Item
---------------------	-----------------

Returns

An Item or null if recipe was not found

Definition at line 70 of file [CraftingRecipes.cs](#).

```
00071     {
00072         shapedCraftingRecipes.TryGetValue(recipe, out var item);
```

```
00073         if (item != null)
00074             item.itemStackCount = 1;
00075
00076             return item;
00077     }
```

6.20.2.4 GetShaplessRecipieResult() [1/3]

```
static Item BeeGame.Core.Dictionaries.CraftingRecipies.GetShaplessRecipieResult (
    int [] recipe ) [static]
```

Tries to get a shapeless recipe

Parameters

<i>recipe</i>	Recipe to get
---------------	---------------

Returns

Item for the recipe, null if recipe does not exist

Definition at line 172 of file [CraftingRecipies.cs](#).

```
00173     {
00174         var list = recipe.ToList();
00175         list.Sort();
00176
00177         var stringRecipe = "";
00178
00179         for (int i = 0; i < list.Count; i++)
00180         {
00181             stringRecipe += $"{list[i]}:";
00182         }
00183
00184         return GetShaplessRecipieResult(stringRecipe);
00185     }
```

6.20.2.5 GetShaplessRecipieResult() [2/3]

```
static Item BeeGame.Core.Dictionaries.CraftingRecipies.GetShaplessRecipieResult (
    string recipe ) [static]
```

Tries to get a shapeless recipe

Parameters

<i>recipe</i>	Recipe to get
---------------	---------------

Returns

Item for the recipe, null if recipe does not exist

Definition at line 192 of file [CraftingRecipies.cs](#).

```
00193     {
00194         shaplessRecipies.TryGetValue(recipe, out var item);
00195         return item;
00196     }
00197 }
```

6.20.2.6 GetShaplessRecipieResult() [3/3]

```
static Item BeeGame.Core.Dictionaries.CraftingRecipies.GetShaplessRecipieResult (
    Item [] recipe ) [static]
```

Tries to get a shapeless recipe

Parameters

<i>recipe</i>	Recipe to get
---------------	---------------

Returns

Item for the recipe, null if does not exist

Definition at line 205 of file [CraftingRecipies.cs](#).

```
00206     {
00207         shaplessRecipies.TryGetValue(
00208             GetShaplessRecipieString(recipe), out var item);
00209         if (item != null)
00210             item.itemStackCount = 1;
00211         return item;
00212     }
00213 }
```

6.20.2.7 GetShaplessRecipieString()

```
static string BeeGame.Core.Dictionaries.CraftingRecipies.GetShaplessRecipieString (
    Item [] recipe ) [static]
```

Gets a shapeless recipe string from a given recipe

Parameters

<i>recipe</i>	Recipe for string
---------------	-------------------

Returns

A string of the given shapeless recipe

Definition at line 143 of file [CraftingRecipies.cs](#).

```

00144         {
00145             var IDList = new List<int>();
00146             var stringRecipe = "";
00147
00148             /* converts the given item list to an ID list so it can be sorted
00149             for (int i = 0; i < recipe.Length; i++)
00150             {
00151                 if(recipe[i] != null)
00152                     IDList.Add(recipe[i].GetHashCode());
00153             }
00154
00155             IDList.Sort();
00156
00157             /* converts the sorted ID list to a string so it can be used as a dictionary key
00158             for (int i = 0; i < IDList.Count; i++)
00159             {
00160                 /* : after each ID as it is possible for ID clashes without eg ID: 11 can be seen as 2 *
ID: 1
00161                 stringRecipe += $"{IDList[i]}:";
00162             }
00163
00164             return stringRecipe;
00165         }

```

6.20.3 Member Data Documentation

6.20.3.1 shapedCraftingRecipies

```
Dictionary<string, Item> BeeGame.Core.Dictionaries.CraftingRecipies.shapedCraftingRecipies =
new Dictionary<string, Item>() [static], [private]
```

Contains all crafting recipes that require a certain layout in the crafting grid ([Blocks.CraftingTable](#))

Definition at line 16 of file [CraftingRecipies.cs](#).

6.20.3.2 shaplessRecipes

```
Dictionary<string, Item> BeeGame.Core.Dictionaries.CraftingRecipies.shaplessRecipes [static],
[private]
```

Initial value:

```
= new Dictionary<string, Item>()
{
    {"5:", new Blocks.Planks() }
}
```

All shapeless recipes

Definition at line 85 of file [CraftingRecipies.cs](#).

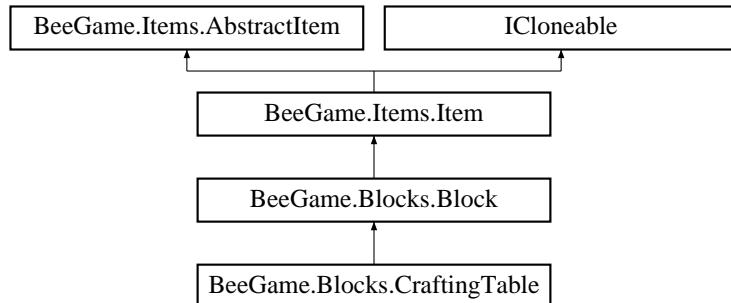
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionaries/[CraftingRecipies.cs](#)

6.21 BeeGame.Blocks.CraftingTable Class Reference

The Workbench [Block](#) class

Inheritance diagram for BeeGame.Blocks.CraftingTable:



Public Member Functions

- [CraftingTable \(\)](#)
Constructor
- [Item ReturnShapedRecipientItem \(Item\[\] items\)](#)
Makes a shaped crafting recipe from the given items and return if it is a recipe
- [virtual Item ReturnShapelessRecipientItem \(Item\[\] items\)](#)
- [virtual Item ReturnShapedRecipientItem \(string recipe\)](#)
Returns a crafting recipe from a given recipe
- [override bool InteractWithBlock \(Inventory.Inventory inv\)](#)
Toggles the CraftingTableInventory for the block
- [override GameObject GetGameObject \(\)](#)
Returns this [Blocks](#) game object
- [override MeshData BlockData \(Chunk chunk, int x, int y, int z, MeshData meshData, bool addToRender ← Mesh=true\)](#)
The data that this block adds to the mesh
- [override bool IsSolid \(Direction direction\)](#)
So that the game knows to render the faces of the blocks around the crafting table as they can be seen
- [override void BreakBlock \(THVector3 pos\)](#)
Breaks the [Block](#)
- [override Sprite GetItemSprite \(\)](#)
Returns the sprite for the [Item](#)
- [override Tile TexturePosition \(Direction direction\)](#)
Returns the texture for the crafting table [Block](#)
- [override int GetHashCode \(\)](#)
Returns the ID of the [Item](#)

Static Public Attributes

- static new int [ID => 9](#)
This block's ID

Private Attributes

- `GameObject myGameObject`
The GameObject for this block

Additional Inherited Members

6.21.1 Detailed Description

The Workbench [Block](#) class

Definition at line 15 of file [CraftingTable.cs](#).

6.21.2 Constructor & Destructor Documentation

6.21.2.1 CraftingTable()

`BeeGame.Blocks.CraftingTable.CraftingTable ()`

Constructor

Definition at line 34 of file [CraftingTable.cs](#).

```
00034             : base("Workbench")
00035     {
00036         usesGameObject = true;
00037     }
```

6.21.3 Member Function Documentation

6.21.3.1 BlockData()

```
override MeshData BeeGame.Blocks.CraftingTable.BlockData (
    Chunk chunk,
    int x,
    int y,
    int z,
    MeshData meshData,
    bool addToRenderMesh = true ) [virtual]
```

The data that this block adds to the mesh

Parameters

<code>chunk</code>	Chunk the block is in
<code>x</code>	X pos of the block
<code>y</code>	Y pos of the block
<code>z</code>	Z pos of the block
<small>Generated by Doxygen</small> <code>meshData</code>	meshdata to add to
<code>addToRenderMesh</code>	should the block also be added to the render mesh not just the collision mesh

Returns

Given *meshData* with this blocks data added to it

Only adds to the colision mesh as the model is handlled by the unity prefab system

Reimplemented from [BeeGame.Blocks.Block](#).

Definition at line 118 of file [CraftingTable.cs](#).

```
00119      {
00120          if (myGameObject == null)
00121          {
00122              myGameObject = UnityEngine.Object.Instantiate(
00123                  PrefabDictionary.GetPrefab("CraftingTable"), new
00124                  THVector3(x, y, z) + chunk.chunkWorldPos, Quaternion.identity, chunk.transform);
00125          }
00126          return base.BlockData(chunk, x, y, z, meshData, true);
00127      }
```

6.21.3.2 BreakBlock()

```
override void BeeGame.Blocks.CraftingTable.BreakBlock (
    THVector3 pos) [virtual]
```

Breaks the [Block](#)

Parameters

<i>pos</i>	Positon of the Block
------------	--------------------------------------

Reimplemented from [BeeGame.Blocks.Block](#).

Definition at line 141 of file [CraftingTable.cs](#).

```
00142      {
00143          /* removes the game object
00144          UnityEngine.Object.Destroy(myGameObject);
00145          /* removes the collision mesh from the chunk
00146          base.BreakBlock(pos);
00147      }
```

6.21.3.3 GetGameObject()

```
override GameObject BeeGame.Blocks.CraftingTable.GetGameObject () [virtual]
```

Returns this [Blocks](#) game object

Returns

Reimplemented from [BeeGame.Items.Item](#).

Definition at line 100 of file [CraftingTable.cs](#).

```
00101      {
00102          return PrefabDictionary.GetPrefab("CraftingTable");
00103      }
```

6.21.3.4 GetHashCode()

```
override int BeeGame.Blocks.CraftingTable.GetHashCode ( ) [virtual]
```

Returns the ID of the Item

Returns

ID

Reimplemented from [BeeGame.Blocks.Block](#).

Definition at line 177 of file [CraftingTable.cs](#).

```
00178     {
00179         return ID;
00180     }
```

6.21.3.5 GetItemSprite()

```
override Sprite BeeGame.Blocks.CraftingTable.GetItemSprite ( ) [virtual]
```

Returns the sprite for the [Item](#)

Returns

Sprite for this [Item](#)

Reimplemented from [BeeGame.Blocks.Block](#).

Definition at line 153 of file [CraftingTable.cs](#).

```
00154     {
00155         return SpriteDictionary.GetSprite ("CraftingTable");
00156     }
```

6.21.3.6 InteractWithBlock()

```
override bool BeeGame.Blocks.CraftingTable.InteractWithBlock (
    Inventory.Inventory inv )
```

Toggles the CraftingTableInventory for the block

Parameters

<i>inv</i>	
------------	--

Returns

Definition at line 89 of file [CraftingTable.cs](#).

```
00090         {
00091             myGameObject.GetComponent<Inventory.BlockInventory.CraftingTableInventory>().
00092             myblock = this;
00093             myGameObject.GetComponent<Inventory.BlockInventory.CraftingTableInventory>().
00094             ToggleInventory(inv);
00095             return true;
00096         }
```

6.21.3.7 IsSolid()

```
override bool BeeGame.Blocks.CraftingTable.IsSolid (
    Direction direction) [virtual]
```

So that the game knows to render the faces of the blocks around the crafting table as they can be seen

Parameters

<i>direction</i>	Direction
------------------	-----------

Returns

false

Reimplemented from [BeeGame.Blocks.Block](#).

Definition at line 132 of file [CraftingTable.cs](#).

```
00133         {
00134             return false;
00135         }
```

6.21.3.8 ReturnShapedRecipientItem() [1/2]

```
Item BeeGame.Blocks.CraftingTable.ReturnShapedRecipientItem (
    Item[] items)
```

Makes a shaped crafting recipe from the given items and return if it is a recipe

Parameters

<i>items</i>	Items to make the recipe from
--------------	-------------------------------

Returns

A [Item](#) if the recipe exists

Definition at line 46 of file [CraftingTable.cs](#).

```
00047      {
00048          var recipe = "";
00049
00050          for (int i = 0; i < items.Length; i++)
00051          {
00052              if (items[i] == null)
00053              {
00054                  recipe += "0:";
00055                  continue;
00056              }
00057
00058              recipe += $"{items[i].GetItemID()}:";
00059          }
00060
00061          return ReturnShapedRecipieItem(recipe);
00062      }
```

6.21.3.9 ReturnShapedRecipieItem() [2/2]

```
virtual Item BeeGame.Blocks.CraftingTable.ReturnShapedRecipieItem (
    string recipe) [virtual]
```

Returns a crafting recipe from a given recipe

Parameters

<i>recipe</i>	
---------------	--

Returns

A [Item](#) if the recipe exists

Virtual incase needs to be overriden by a different crafting system

Definition at line 77 of file [CraftingTable.cs](#).

```
00078      {
00079          return CraftingRecipes.GetShapedRecipeItem(recipe);
00080      }
```

6.21.3.10 ReturnShapelessRecipieItem()

```
virtual Item BeeGame.Blocks.CraftingTable.ReturnShapelessRecipieItem (
    Item [] items) [virtual]
```

Definition at line 64 of file [CraftingTable.cs](#).

```
00065      {
00066          return CraftingRecipes.GetShaplessRecipieResult(items)
00067      ;}
```

6.21.3.11 TexturePosition()

```
override Tile BeeGame.Blocks.CraftingTable.TexturePosition (
    Direction direction ) [virtual]
```

Returns the texture for the crafting table [Block](#)

Parameters

<i>direction</i>	Direction of the desired face
------------------	-------------------------------

Returns

Tile with the texture coordinates of the [Block](#) texture

Returns a transparent texture as the chest model already has a texture applied

Reimplemented from [BeeGame.Items.Item](#).

Definition at line 166 of file [CraftingTable.cs](#).

```
00167      {
00168          return new Tile() { x = 0, y = 9 };
00169      }
```

6.21.4 Member Data Documentation

6.21.4.1 ID

```
new int BeeGame.Blocks.CraftingTable.ID => 9 [static]
```

This block's ID

Definition at line 27 of file [CraftingTable.cs](#).

6.21.4.2 myGameobject

```
GameObject BeeGame.Blocks.CraftingTable.myGameobject [private]
```

The GameObject for this block

Definition at line 22 of file [CraftingTable.cs](#).

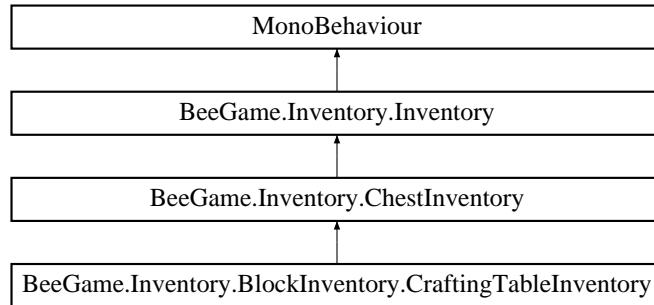
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/[CraftingTable.cs](#)

6.22 BeeGame.Inventory.BlockInventory.CraftingTableInventory Class Reference

Inventory for the CraftingTable Block

Inheritance diagram for BeeGame.Inventory.BlockInventory.CraftingTableInventory:



Public Member Functions

- delegate void [ItemRemovedFromResult \(\)](#)
Makes the delegate
- virtual [Item CheckShapedRecipie \(\)](#)
Check in the recipie in the grid for a shaped crafting recipe
- virtual [Item CheckShapelessRecipie \(\)](#)
Check in the recipe grid for a shapless crafting recipe
- void [CraftedItemRemoved \(\)](#)
Removes the items form the crafting grid one an item has been removed from the crafting result slot. Called via the `craftingResultRemoved` delegate from `InventorySlot.OnPointerClick(UnityEngine.EventSystems.PointerEventData)`
- virtual void [DropItemsFromInventory \(\)](#)
Removes all Items from the inventory when it is closed
- override void [ToggleInventory \(Inventory inv\)](#)
Opens/Closes the inventory
- override void [SetChestInventory \(string invName="Workbench"\)](#)
Set the size of the `Inventory`
- override void [AddItemToSlots \(int slotIndex, Item item\)](#)
Adds an item to a `InventorySlot`
- override void [SaveInv \(\)](#)
Oerriden so the inventory is not saved in any way

Public Attributes

- [ItemRemovedFromResult craftingResultRemoved](#)
Holds the method for the delegate to call

Protected Member Functions

- void [Start \(\)](#)
Sets the size of the inventory
- void [Update \(\)](#)
Updates the base and checks crafting recipies
- void [OnDestroy \(\)](#)
Ensuring no memory leaks occur due to the delegate

Additional Inherited Members

6.22.1 Detailed Description

Invnetry for the CraftingTable Block

Definition at line 15 of file [CraftingTableInventory.cs](#).

6.22.2 Member Function Documentation

6.22.2.1 AddItemToSlots()

```
override void BeeGame.Inventory.BlockInventory.CraftingTableInventory.AddItemToSlots (
    int slotIndex,
    Item item ) [virtual]
```

Adds an item to a [InventorySlot](#)

Parameters

<i>slotIndex</i>	InventorySlot.slotIndex to add the items to
<i>item</i>	Item to add

Overriden so serialization does not occur

Reimplemented from [BeeGame.Inventory.Inventory](#).

Definition at line 187 of file [CraftingTableInventory.cs](#).

```
00188     {
00189         items.AddItem(slotIndex, item);
00190     }
```

6.22.2.2 CheckShapedRecipie()

```
virtual Item BeeGame.Inventory.BlockInventory.CraftingTableInventory.CheckShapedRecipie ( )
[virtual]
```

Check in the recipie in the grid for a shaped crafting recipe

Definition at line 81 of file [CraftingTableInventory.cs](#).

```
00082     {
00083         var items = new Item[9];
00084
00085         for (int i = 0; i < items.Length; i++)
00086         {
00087             items[i] = base.items.itemsInInventory[i];
00088         }
00089
00090         /* if it is a recipe put the result into the crafting result slot
00091         return ((CraftingTable)myblock).ReturnShapedRecipieItem(items);
00092     }
```

6.22.2.3 CheckShapelessRecipie()

```
virtual Item BeeGame.Inventory.BlockInventory.CraftingTableInventory.CheckShapelessRecipie ( )
[virtual]
```

Check in the recipe grid for a shapless crafting recipe

Definition at line 97 of file [CraftingTableInventory.cs](#).

```
00098      {
00099          var items = new Item[9];
00100
00101          for (int i = 0; i < items.Length; i++)
00102          {
00103              items[i] = base.items.itemsInInventory[i];
00104          }
00105
00106          return ((CraftingTable)myblock).ReturnShapelessRecipieItem(items);
00107      }
```

6.22.2.4 CraftedItemRemoved()

```
void BeeGame.Inventory.BlockInventory.CraftingTableInventory.CraftedItemRemoved ( )
```

Removes the items form the crafting grid one an item has been removed from the crafting result slot, Called via the `craftingResultRemoved` delegate from `InventorySlot.OnPointerClick(UnityEngine.EventSystems.PointerEventData)`

Definition at line 112 of file [CraftingTableInventory.cs](#).

```
00113      {
00114          if (items.itemsInInventory[9] != null)
00115          {
00116              QuestEvents.CallItemCraftedEvent(
00117                  items.itemsInInventory[9].GetHashCode());
00118              for (int i = 0; i < 9; i++)
00119              {
00120                  if (items.itemsInInventory[i] != null)
00121                      items.itemsInInventory[i].
00122                      itemStackCount -= 1;
00123              }
00124          }
00125      }
```

6.22.2.5 DropItemsFromInventory()

```
virtual void BeeGame.Inventory.BlockInventory.CraftingTableInventory.DropItemsFromInventory ( )
) [virtual]
```

Removes all Items from the inventory when it is closed

Called by the output invenotry slot as it is a button

Definition at line 133 of file [CraftingTableInventory.cs](#).

```
00134      {
00135          /* looks at every item in the crafting grid
00136          for (int i = 0; i < 9; i++)
00137          {
00138              if (items.itemsInInventory[i] != null)
00139              {
00140                  /* spawns it and removes it from the inventory if an items exists within
00141                  for (int j = 0; j < items.itemsInInventory[i].
00142                  itemStackCount; j++)
00143                  {
00144                      items.itemsInInventory[i].SpawnItem(
00145                          THVector3)this.transform.position + new THVector3(0, 1, 0));
00146                  }
00147              }
00148          }
```

6.22.2.6 ItemRemovedFromResult()

```
delegate void BeeGame.Inventory.BlockInventory.CraftingTableInventory.ItemRemovedFromResult ( )
```

Makes the delegate

6.22.2.7 OnDestroy()

```
void BeeGame.Inventory.BlockInventory.CraftingTableInventory.OnDestroy ( ) [protected]
```

Ensuring no memory leaks occur due to the delegate

Definition at line 70 of file [CraftingTableInventory.cs](#).

```
00071      {
00072          /* just ensures no memory leaks occur
00073          craftingResultRemoved -= CraftedItemRemoved;
00074      }
```

6.22.2.8 SaveInv()

```
override void BeeGame.Inventory.BlockInventory.CraftingTableInventory.SaveInv ( ) [virtual]
```

OVERRIDDEN so the inventory is not saved in any way

Reimplemented from [BeeGame.Inventory.Inventory](#).

Definition at line 195 of file [CraftingTableInventory.cs](#).

```
00196      {
00197          /* does not need to be saved so overridden to do nothing
00198      }
```

6.22.2.9 SetChestInventory()

```
override void BeeGame.Inventory.BlockInventory.CraftingTableInventory.SetChestInventory (
    string invName = "Workbench" ) [virtual]
```

Set the size of the [Inventory](#)

Parameters

<i>invName</i>	Workbench
----------------	-----------

overridden here so that no attempt is made to deserialize the inventory helping with performance

Reimplemented from [BeeGame.Inventory.ChestInventory](#).

Definition at line 172 of file [CraftingTableInventory.cs](#).

```
00173     {
00174         SetInventorySize(inventorySize);
00175         /* sets the UI to not be seen as inventories cannot start open
00176         inventory.SetActive(false);
00177     }
```

6.22.2.10 Start()

```
void BeeGame.Inventory.BlockInventory.CraftingTableInventory.Start ( ) [protected]
```

Sets the size of the inventory

Definition at line 32 of file [CraftingTableInventory.cs](#).

```
00033     {
00034         SetChestInventory ();
00035         craftingResultRemoved = CraftedItemRemoved;
00036     }
```

6.22.2.11 ToggleInventory()

```
override void BeeGame.Inventory.BlockInventory.CraftingTableInventory.ToggleInventory (
    Inventory inv ) [virtual]
```

Opens/Closes the inventory

Parameters

<i>inv</i>	The inventory to toggle
------------	-------------------------

Reimplemented from [BeeGame.Inventory.ChestInventory](#).

Definition at line 156 of file [CraftingTableInventory.cs](#).

```
00157     {
00158         base.ToggleInventory(inv);
00159
00160         /* if the inventory was closed drop the items within
00161         if (!inventory.activeInHierarchy)
00162             DropItemsFromInventory ();
00163     }
```

6.22.2.12 Update()

```
void BeeGame.Inventory.BlockInventory.CraftingTableInventory.Update () [protected]
```

Updates the base and checks crafting recipies

Definition at line 42 of file [CraftingTableInventory.cs](#).

```
00043      {
00044          UpdateChestInventory();
00045
00046          if (inventory.activeInHierarchy)
00047          {
00048              var shaped = CheckShapedRecipie();
00049              var shapless = CheckShapelessRecipie();
00050
00051              if (shaped != null)
00052              {
00053                  items.itemsInInventory[9] = shaped;
00054                  return;
00055              }
00056              /* checks for shapless recipies second
00057              else if(shapless != null)
00058              {
00059                  items.itemsInInventory[9] = shapless;
00060                  return;
00061              }
00062
00063          items.itemsInInventory[9] = null;
00064      }
00065 }
```

6.22.3 Member Data Documentation

6.22.3.1 craftingResultRemoved

```
ItemRemovedFromResult BeeGame.Inventory.BlockInventory.CraftingTableInventory.craftingResult←
Removed
```

Holds the method for the delegate to call

Definition at line 25 of file [CraftingTableInventory.cs](#).

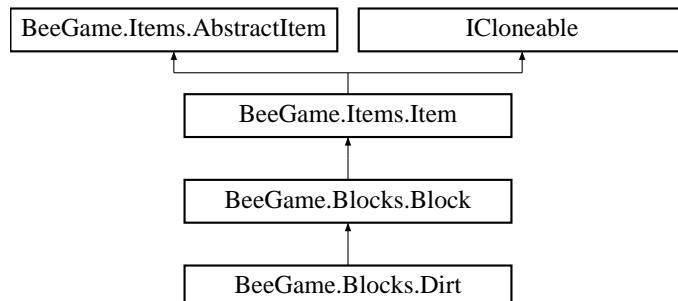
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/BlockInventory/[CraftingTableInventory.cs](#)

6.23 BeeGame.Blocks.Dirt Class Reference

Dirt Block

Inheritance diagram for BeeGame.Blocks.Dirt:



Public Member Functions

- [Dirt \(\)](#)
Constructor
- [override Sprite GetItemSprite \(\)](#)
Returns the sprite for the item
- [override Tile TexturePosition \(Direction direction\)](#)
Position of the dirt texture in the atlas
- [override int GetHashCode \(\)](#)
Base ID of the block
- [override string ToString \(\)](#)
Returns the name and ID of the block as a string

Static Public Attributes

- static new int [ID => 3](#)

Additional Inherited Members

6.23.1 Detailed Description

[Dirt Block](#)

Definition at line [13](#) of file [Dirt.cs](#).

6.23.2 Constructor & Destructor Documentation

6.23.2.1 [Dirt\(\)](#)

`BeeGame.Blocks.Dirt.Dirt ()`

Constructor

Definition at line [21](#) of file [Dirt.cs](#).

```
00021 : base("Dirt"){}  
}
```

6.23.3 Member Function Documentation

6.23.3.1 GetHashCode()

```
override int BeeGame.Blocks.Dirt.GetHashCode () [virtual]
```

Base ID of the block

Returns

5

Reimplemented from [BeeGame.Blocks.Block](#).

Definition at line [48](#) of file [Dirt.cs](#).

```
00049     {  
00050         return ID;  
00051     }
```

6.23.3.2 GetItemSprite()

```
override Sprite BeeGame.Blocks.Dirt.GetItemSprite () [virtual]
```

Returns the sprite for the item

Returns

Sprite for this item

Reimplemented from [BeeGame.Blocks.Block](#).

Definition at line [25](#) of file [Dirt.cs](#).

```
00026     {  
00027         return SpriteDictionary.GetSprite("Dirt");  
00028     }
```

6.23.3.3 TexturePosition()

```
override Tile BeeGame.Blocks.Dirt.TexturePosition (  
    Direction direction) [virtual]
```

Position of the dirt texture in the atlas

Parameters

<i>direction</i>	<input type="text"/>
------------------	----------------------

Returns

Reimplemented from [BeeGame.Items.Item](#).

Definition at line 37 of file [Dirt.cs](#).

```
00038     {
00039         return new Tile { x = 2, y = 9 };
00040     }
```

6.23.3.4 ToString()

```
override string BeeGame.Blocks.Dirt.ToString ()
```

Returns the name and ID of the block as a string

Returns

A nicely formatted string

Definition at line 57 of file [Dirt.cs](#).

```
00058     {
00059         return $"{itemName} \nID: {GetItemID()}";
00060     }
```

6.23.4 Member Data Documentation**6.23.4.1 ID**

```
new int BeeGame.Blocks.Dirt.ID => 3 [static]
```

Definition at line 15 of file [Dirt.cs](#).

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/[Dirt.cs](#)

6.24 BeeGame.Core.Extensions Class Reference**Static Public Member Functions**

- static T [CloneObject< T >](#) (this T obj)
Allows the copying of a class by value using reflection
- static Sprite [ColourSprite](#) (this Sprite sprite, Color colour, Color[] coloursToAvoid=null, bool setTransparent← ToWhite=false)
Will colour the sprite given a colour and optionaly colours to avoid
- static void [SpawnItem](#) (this Item item, THVector3 position, Quaternion rotation=new Quaternion())

6.24.1 Detailed Description

Definition at line 8 of file [Extensions.cs](#).

6.24.2 Member Function Documentation

6.24.2.1 `CloneObject< T >()`

```
static T BeeGame.Core.Extensions.CloneObject< T > (
    this T obj) [static]
```

Allows the copying of a class by value using reflection

Parameters

<code>obj</code>	Object to copy
------------------	----------------

Returns

a new object with all values copied

Much faster than the serialize method however a lot more complicated

Definition at line 18 of file [Extensions.cs](#).

```
00019      {
00020          /* gets the type of the given object
00021          Type typeSource = obj.GetType();
00022
00023          /* makes a new object of type T
00024          T objTarget = (T)Activator.CreateInstance(typeSource);
00025
00026          /* gets the properties in T
00027          PropertyInfo[] propertyInfo = typeSource.GetProperties(BindingFlags.Public | BindingFlags.
NonPublic | BindingFlags.Instance);
00028
00029          /* applies the properties in T to the new type T object
00030          foreach (var property in propertyInfo)
00031          {
00032              if (property.CanWrite)
00033              {
00034                  /* if the property is a value just set it
00035                  if (property.PropertyType.IsValueType || property.PropertyType.IsEnum || property.
.PropertyType.Equals(typeof(string)))
00036                  {
00037                      property.SetValue(objTarget, property.GetValue(obj, null), null);
00038                  }
00039                  else
00040                  {
00041                      /* if the property is not a value type this function will need to be called
recursively as it could also have non value type variables
00042                      object PropertyValue = property.GetValue(obj, null);
00043
00044                      if (PropertyValue == null)
00045                      {
00046                          property.SetValue(objTarget, null, null);
00047                      }
00048                      else
00049                      {
00050                          property.SetValue(objTarget, PropertyValue.CloneObject(), null);
00051                      }
00052                  }
00053              }
```

```

00054         }
00055
00056     /* gets all of the field in T
00057     FieldInfo[] fieldInfo = typeSource.GetFields();
00058
00059     /* applies all of the fields of T to the new object if type T in the same manor that the
00060     properties are applied
00061     foreach (var field in fieldInfo)
00062     {
00063         if(field.FieldType.IsValueType || field.FieldType.IsEnum || field.FieldType.Equals(typeof(
00064             string)))
00065         {
00066             if(field.SetValue(objTarget, field.GetValue(obj)));
00067         }
00068         else
00069         {
00070             object fieldValue = field.GetValue(obj);
00071             if(fieldValue == null)
00072             {
00073                 field.SetValue(objTarget, null);
00074             }
00075             else
00076             {
00077                 field.SetValue(objTarget, field.CloneObject());
00078             }
00079         }
00080     }
00081     return objTarget;
00082 }
```

6.24.2.2 ColourSprite()

```
static Sprite BeeGame.Core.Extensions.ColourSprite (
    this Sprite sprite,
    Color colour,
    Color [] coloursToAvoid = null,
    bool setTransparentToWhite = false ) [static]
```

Will colour the sprite given a colour and optionaly colours to avoid

Parameters

<i>sprite</i>	Sprite to colour
<i>colour</i>	Colour to set the sprite to
<i>coloursToAvoid</i>	Colours to avoid, Optional
<i>setTransparentToWhite</i>	Should transparent value to set wo white, Default true

Returns

Definition at line 92 of file [Extensions.cs](#).

```

00093     {
00094         Texture2D tex = new Texture2D((int)sprite.rect.width, (int)sprite.rect.height)
00095         {
00096             filterMode = FilterMode.Point,
00097             wrapMode = TextureWrapMode.Clamp
00098         };
00099
00100         /* sets the texture pixels to the pixels of teh sprite so the original sprite is not modified
00101         tex.SetPixels(sprite.texture.GetPixels());
```

```

00102         for (int x = 0; x < tex.width; x++)
00103     {
00104         for (int y = 0; y < tex.height; y++)
00105     {
00106         /* if we dont have to avoid any colours set the pixel
00107         if (coloursToAvoid == null)
00108         {
00109             tex.SetPixel(x, y, tex.GetPixel(x, y) * colour);
00110         }
00111         else
00112         {
00113             for (int i = 0; i < coloursToAvoid.Length; i++)
00114             {
00115                 /* if this colour should be avoided skip this iteration of the loop and move
00116                 on
00117                 if (tex.GetPixel(x, y) == coloursToAvoid[i])
00118                     goto Skip;
00119                 }
00120
00121                 tex.SetPixel(x, y, tex.GetPixel(x, y) * colour);
00122             }
00123
00124             /* if transparent pixels should be set to white do that
00125             if (setTransparentToWhite && tex.GetPixel(x, y).a == 0)
00126                 tex.SetPixel(x, y, Color.white);
00127
00128             Skip:
00129             continue;
00130         }
00131     }
00132
00133     /* apply the new texture with its colours
00134     tex.Apply();
00135
00136     /* return the Texture2D as a sprite
00137     return Sprite.Create(tex, new Rect(0, 0, tex.width, tex.height), new THVector2(0.5f, 0.5f));
00138 }
```

6.24.2.3 SpawnItem()

```

static void BeeGame.Core.Extensions.SpawnItem (
    this Item item,
    THVector3 position,
    Quaternion rotation = new Quaternion() ) [static]
```

Definition at line 140 of file [Extensions.cs](#).

```

00141         {
00142             GameObject go = MonoBehaviour.Instantiate(UnityEngine.Resources.Load("
00143             Prefabs/ItemGameObject") as GameObject, position, rotation) as GameObject;
00144             go.GetComponent<ItemGameObject>().item = item;
00145         }
```

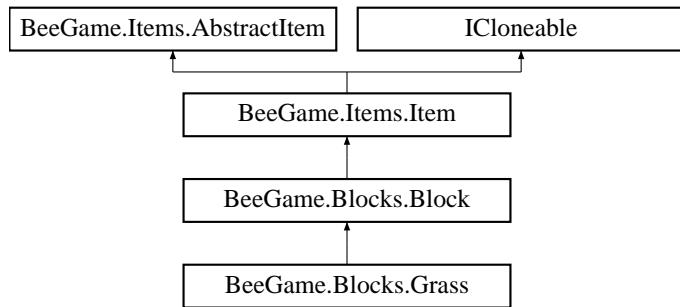
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/[Extensions.cs](#)

6.25 BeeGame.Blocks.Grass Class Reference

[Grass Block](#)

Inheritance diagram for BeeGame.Blocks.Grass:



Public Member Functions

- `Grass ()`
Constructor also sets the items name
- `override Sprite GetItemSprite ()`
Returns the sprite for the item
- `override void UpdateBlock (int x, int y, int z, Chunk chunk)`
Will turn this `Block` into a `Dirt` block if another block is above it
- `override Tile TexturePosition (Direction direction)`
Texture position of the `Block` face
- `override int GetHashCode ()`
The Base ID for the block
- `override string ToString ()`
Returns the name and value for the block as a string

Static Public Attributes

- `static new int ID => 4`

Additional Inherited Members

6.25.1 Detailed Description

Grass Block

Definition at line 14 of file [Grass.cs](#).

6.25.2 Constructor & Destructor Documentation

6.25.2.1 Grass()

`BeeGame.Blocks.Grass.Grass ()`

Constructor also sets the items name

Definition at line 22 of file [Grass.cs](#).

```
00022 : base("Grass") {}
```

6.25.3 Member Function Documentation

6.25.3.1 GetHashCode()

```
override int BeeGame.Blocks.Grass.GetHashCode ( ) [virtual]
```

The Base ID for the block

Returns

4

Reimplemented from [BeeGame.Blocks.Block](#).

Definition at line 82 of file [Grass.cs](#).

```
00083     {
00084         return ID;
00085     }
```

6.25.3.2 GetItemSprite()

```
override Sprite BeeGame.Blocks.Grass.GetItemSprite ( ) [virtual]
```

Returns the sprite for the item

Returns

Sprite for this item

Reimplemented from [BeeGame.Blocks.Block](#).

Definition at line 26 of file [Grass.cs](#).

```
00027     {
00028         return SpriteDictionary.GetSprite("Grass");
00029     }
```

6.25.3.3 TexturePosition()

```
override Tile BeeGame.Blocks.Grass.TexturePosition (
    Direction direction ) [virtual]
```

Texture position of the [Block](#) face

Parameters

<i>direction</i>	Direction of the block face
------------------	-----------------------------

Returns

Texture positon as a Tile

Reimplemented from [BeeGame.Items.Item](#).

Definition at line 51 of file [Grass.cs](#).

```
00052      {
00053          //All textures are on the same Y value for the texture atlas so Y can be set
00054          Tile tile = new Tile()
00055          {
00056              y = 9
00057          };
00058
00059          switch (direction)
00060          {
00061              //if we want the top face return the full grass texture
00062              case Direction.UP:
00063                  tile.x = 3;
00064                  return tile;
00065              //if we want the bottom face return the dirt texture
00066              case Direction.DOWN:
00067                  tile.x = 2;
00068                  return tile;
00069              //return the 1/2 grass texture if a side face is wanted
00070              default:
00071                  tile.x = 4;
00072                  return tile;
00073          }
00074      }
```

6.25.3.4 ToString()

```
override string BeeGame.Blocks.Grass.ToString ( )
```

Returns the name and value for the block as a string

Returns

A nicely formatted string

Definition at line 91 of file [Grass.cs](#).

```
00092      {
00093          return $"{itemName} \nID: {GetItemID()}";
00094      }
```

6.25.3.5 UpdateBlock()

```
override void BeeGame.Blocks.Grass.UpdateBlock (
    int x,
    int y,
    int z,
    Chunk chunk ) [virtual]
```

Will turn this [Block](#) into a [Dirt](#) block if another block is above it

Parameters

<i>x</i>	X pos if the block
<i>y</i>	Y pos if the block
<i>z</i>	Z pos if the block
<i>chunk</i>	Chunk that this block is in

Reimplemented from [BeeGame.Blocks.Block](#).

Definition at line [40](#) of file [Grass.cs](#).

```
00041      {
00042          if (chunk.GetBlock(x, y + 1, z, false).IsSolid(Direction.DOWN))
00043              chunk.blocks[x, y, z] = new Dirt() { changed =
00044      changed };
00045 }
```

6.25.4 Member Data Documentation

6.25.4.1 ID

```
new int BeeGame.Blocks.Grass.ID => 4 [static]
```

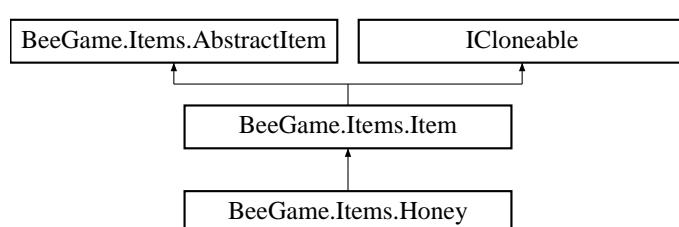
Definition at line [16](#) of file [Grass.cs](#).

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/[Grass.cs](#)

6.26 BeeGame.Items.Honey Class Reference

Inheritance diagram for BeeGame.Items.Honey:



Public Member Functions

- override string [ToString \(\)](#)
- override string [GetItemID \(\)](#)
- override int [GetHashCode \(\)](#)

Static Public Attributes

- static new int ID => 14

Additional Inherited Members

6.26.1 Detailed Description

Definition at line 8 of file [Honey.cs](#).

6.26.2 Member Function Documentation

6.26.2.1 GetHashCode()

```
override int BeeGame.Items.Honey.GetHashCode () [virtual]
```

Implements [BeeGame.Items.AbstractItem](#).

Definition at line 22 of file [Honey.cs](#).

```
00023     {
00024         return ID;
00025     }
```

6.26.2.2 GetItemID()

```
override string BeeGame.Items.Honey.GetItemID () [virtual]
```

Implements [BeeGame.Items.AbstractItem](#).

Definition at line 17 of file [Honey.cs](#).

```
00018     {
00019         return GetHashCode().ToString(); ;
00020     }
```

6.26.2.3 ToString()

```
override string BeeGame.Items.Honey.ToString ()
```

Definition at line 12 of file [Honey.cs](#).

```
00013     {
00014         return $"{itemName} \\\ {GetItemID()}";
00015     }
```

6.26.3 Member Data Documentation

6.26.3.1 ID

```
new int BeeGame.Items.Honey.ID => 14 [static]
```

Definition at line 10 of file [Honey.cs](#).

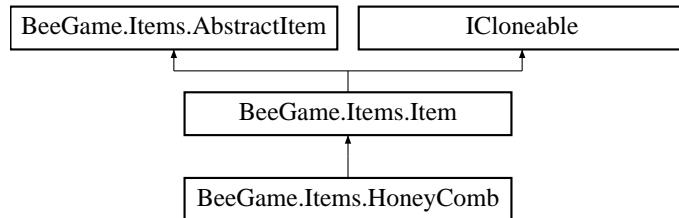
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/[Honey.cs](#)

6.27 BeeGame.Items.HoneyComb Class Reference

[Honey](#) comb item produced by bees

Inheritance diagram for BeeGame.Items.HoneyComb:



Public Member Functions

- [HoneyComb \(\)](#)
Make the [Item](#) from no arguments giving it the default honey comb value `HoneyCombType.HONEY`
- [HoneyComb \(HoneyCombType type\)](#)
Makes a [HoneyComb](#) for the given `HoneyCombType`
- override Sprite [GetItemSprite \(\)](#)
Returns the sprite for this of the correct colour
- override GameObject [GetGameObject \(\)](#)
Returns the game object for this and gives the object the correct colouring
- override string [GetItemID \(\)](#)
Makes the item ID. For this it is the Normal ID | the int value of the [type](#) this comb is
- override int [GetHashCode \(\)](#)
Returns the hashcode for this [Item](#)

Static Public Attributes

- static new int [ID => 12](#)

Properties

- **HoneyCombType type** [get, set]
The type of comb this is, HoneyCombType
- **Color CombColour** [get]
The colour if this comb, BeeDictionaries.GetCombColour(HoneyCombType)

Private Attributes

- **Sprite itemSprite**
The Sprite for this honey comb

Additional Inherited Members

6.27.1 Detailed Description

Honey comb item produced by bees

Definition at line 14 of file [HoneyComb.cs](#).

6.27.2 Constructor & Destructor Documentation

6.27.2.1 HoneyComb() [1/2]

`BeeGame.Items.HoneyComb.HoneyComb()`

Make the **Item** from no arguments giving it the default honey comb value HoneyCombType.HONEY

Definition at line 46 of file [HoneyComb.cs](#).

```
00046             : base(new CultureInfo("en-US", false).TextInfo.ToTitleCase($"  
00047     {HoneyCombType.HONEY} Comb".ToLower()))  
00048     {  
00049         usesGameObject = true;  
00049         type = HoneyCombType.HONEY;  
00050     }
```

6.27.2.2 HoneyComb() [2/2]

`BeeGame.Items.HoneyComb.HoneyComb(
 HoneyCombType type)`

Makes a **HoneyComb** for the given HoneyCombType

Parameters

<code>type</code>	that this comb is
-------------------	-------------------

Definition at line 56 of file [HoneyComb.cs](#).

```
00056             : base(new CultureInfo("en-US", false).TextInfo.ToTitleCase($""
00057     {type.ToString()} Comb".ToLower()))
00058     {
00059         usesGameObject = true;
00060         this.type = type;
00060     }
```

6.27.3 Member Function Documentation

6.27.3.1 GetGameObject()

```
override GameObject BeeGame.Items.HoneyComb.GetGameObject () [virtual]
```

Returns the game object for this and gives the object the correct colouring

Returns

GameObject for this

Reimplemented from [BeeGame.Items.Item](#).

Definition at line 77 of file [HoneyComb.cs](#).

```
00078     {
00079         GameObject obj = PrefabDictionary.GetPrefab("HoneyComb");
00080         /* cannot access the instance material from here have to do it on the object
00081         obj.GetComponent<ApplyColour>().colour = CombColour;
00082         return obj;
00083     }
```

6.27.3.2 GetHashCode()

```
override int BeeGame.Items.HoneyComb.GetHashCode () [virtual]
```

Returns the hashcode for this [Item](#)

Returns

8

Implements [BeeGame.Items.AbstractItem](#).

Definition at line 100 of file [HoneyComb.cs](#).

```
00101     {
00102         return ID;
00103     }
```

6.27.3.3 GetItemID()

```
override string BeeGame.Items.HoneyComb.GetItemID () [virtual]
```

Makes the item ID. For this it is the Normal ID \ the int value of the [type](#) this comb is

Returns

[Item](#) ID as a string

Implements [BeeGame.Items.AbstractItem](#).

Definition at line 89 of file [HoneyComb.cs](#).

```
00090      {
00091          return ${GetHashCode ()} \\ { (int) type } ;
00092      }
```

6.27.3.4 GetItemSprite()

```
override Sprite BeeGame.Items.HoneyComb.GetItemSprite () [virtual]
```

Retuens the sprite for the this of the correct colour

Returns

[Sprite](#)

Reimplemented from [BeeGame.Items.Item](#).

Definition at line 68 of file [HoneyComb.cs](#).

```
00069      {
00070          return itemSprite ?? (itemSprite =
00071              SpriteDictionary.GetSprite ("HoneyComb").ColourSprite (
00071                  CombColour));
00071      }
```

6.27.4 Member Data Documentation

6.27.4.1 ID

```
new int BeeGame.Items.HoneyComb.ID => 12 [static]
```

Definition at line 39 of file [HoneyComb.cs](#).

6.27.4.2 itemSprite

Sprite BeeGame.Items.HoneyComb.itemSprite [private]

The Sprite for this honey comb

Definition at line 37 of file [HoneyComb.cs](#).

6.27.5 Property Documentation

6.27.5.1 CombColour

Color BeeGame.Items.HoneyComb.CombColour [get]

The colour if this coumb, BeeDictionaries.GetCombColour(HoneyCombType)

Definition at line 26 of file [HoneyComb.cs](#).

6.27.5.2 type

[HoneyCombType](#) BeeGame.Items.HoneyComb.type [get], [set]

The type of comb this is, HoneyCombType

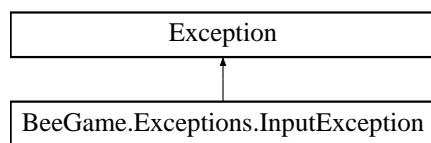
Definition at line 20 of file [HoneyComb.cs](#).

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/[HoneyComb.cs](#)

6.28 BeeGame.Exceptions.InputException Class Reference

Inheritance diagram for BeeGame.Exceptions.InputException:



Public Member Functions

- [InputException \(\)](#)
- [InputException \(string message\)](#)
- [InputException \(string message, Exception innerException\)](#)

6.28.1 Detailed Description

Definition at line [8](#) of file [InputException.cs](#).

6.28.2 Constructor & Destructor Documentation

6.28.2.1 InputException() [1/3]

```
BeeGame.Exceptions.InputException.InputException ( )
```

Definition at line [10](#) of file [InputException.cs](#).

```
00010          : base ()  
00011          {  
00012          }  
00013          }
```

6.28.2.2 InputException() [2/3]

```
BeeGame.Exceptions.InputException.InputException (   
    string message )
```

Definition at line [15](#) of file [InputException.cs](#).

```
00015          : base (message)  
00016          {  
00017          }  
00018          }
```

6.28.2.3 InputException() [3/3]

```
BeeGame.Exceptions.InputException.InputException (   
    string message,  
    Exception innerException )
```

Definition at line [20](#) of file [InputException.cs](#).

```
00020          : base (message, innerException)  
00021          {  
00022          }  
00023          }
```

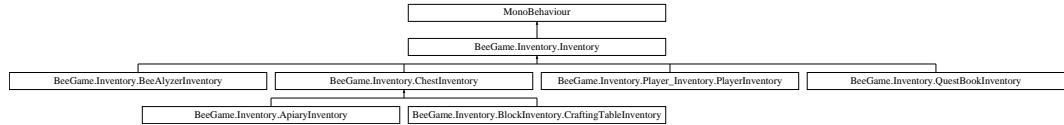
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Exceptions/[InputException.cs](#)

6.29 BeeGame.Inventory.Inventory Class Reference

Base class for all inventories in the game

Inheritance diagram for BeeGame.Inventory.Inventory:



Public Member Functions

- bool [InventorySet \(\)](#)
Is the inventory set?
- void [SetInventorySize \(int inventorySize\)](#)
Sets the inventory size to the number of slots in the inventory
- void [SetAllItems \(ItemsInInventory items\)](#)
Sets the items to the given ItemsInInventory
- virtual void [ToggleInventory \(Inventory inv\)](#)
- virtual void [SaveInv \(\)](#)
Saves the inventory
- virtual void [SetItemInventory \(Item\[\] items\)](#)
- virtual void [PutItemsInSlots \(\)](#)
Sets an Item in the ItemsInInventory.itemsInInventory array to a InventorySlot.item
- [ItemsInInventory GetAllItems \(\)](#)
Gets all of the items in the inventory
- virtual void [AddItemToSlots \(int slotIndex, Item item\)](#)
Adds the given item to the inventory in the given slotIndex
- bool [AddItemToInventory \(Item item\)](#)
Add an item to the inventory

Public Attributes

- [ItemsInInventory items](#)
Items in the inventory
- [InventorySlot \[\] slots](#)
Slots in the inventory
- string [inventoryName = ""](#)
Name of this inventory
- [Blocks.Block myblock](#)
The block class that this inventory is part of

Protected Member Functions

- void [UpdateBase \(\)](#)
Things in the inventory that should be updated

Protected Attributes

- bool `thisInventoryOpen` = false
is this inventory open?

Package Attributes

- `Item floatingItem`
Item that is currently being moved

Private Member Functions

- void `DrawItemAtCursor()`
Draws the `floatingItem`'s `Item.GetItemSprite()` at the mouse position

Private Attributes

- `GameObject spriteAtCursor`
The sprite at the cursor

6.29.1 Detailed Description

Base class for all inventories in the game

Definition at line 11 of file [Inventory.cs](#).

6.29.2 Member Function Documentation

6.29.2.1 AddItemToInventory()

```
bool BeeGame.Inventory.Inventory.AddItemToInventory (
    Item item )
```

Add an item to the inventory

Parameters

<code>item</code>	Item to add
-------------------	-------------

Returns

true if item was added

Definition at line 181 of file [Inventory.cs](#).

```

00182     {
00183         return items.AddItem(item);
00184     }

```

6.29.2.2 AddItemToSlots()

```

virtual void BeeGame.Inventory.Inventory.AddItemToSlots (
    int slotIndex,
    Item item ) [virtual]

```

Adds the given *item* to the inventory in the given *slotIndex*

Parameters

<i>slotIndex</i>	Slot to add item to
<i>item</i>	Item to add

Reimplemented in [BeeGame.Inventory.BlockInventory.CraftingTableInventory](#).

Definition at line 169 of file [Inventory.cs](#).

```

00170     {
00171         items.AddItem(slotIndex, item);
00172         /* saves the inventory changes
00173         Serialization.Serialization.SerializeInventory(this, inventoryName);
00174     }

```

6.29.2.3 DrawItemAtCursor()

```
void BeeGame.Inventory.Inventory.DrawItemAtCursor ( ) [private]
```

Draws the [floatingItem](#)s [Item.GetItemSprite\(\)](#) at the mouse position

Definition at line 95 of file [Inventory.cs](#).

```

00096     {
00097         if(floatingItem != null)
00098         {
00099             if (spriteAtCursor == null)
00100             {
00101                 spriteAtCursor = Instantiate(PrefabDictionary.
GetPrefab("ItemIcon"));
00102                 spriteAtCursor.GetComponentInChildren<
UnityEngine.UI.Image>().sprite = floatingItem.
GetItemSprite();
00103             }
00104             /* will update a the sprite of in item is swapped between a slot and teh floating item if
the previous item wasnt put into a slot first
00105             else if(spriteAtCursor != null)
00106             {
00107                 spriteAtCursor.GetComponentInChildren<
UnityEngine.UI.Image>().sprite = floatingItem.
GetItemSprite();
00108             }
00109             spriteAtCursor.transform.GetChild(0).position = Input.mousePosition;
00110         }
00111     }
00112     else
00113     {
00114         Destroy(spriteAtCursor);
00115     }
00116 }

```

6.29.2.4 GetAllItems()

```
ItemsInInventory BeeGame.Inventory.Inventory.GetAllItems ( )
```

Gets all of the items in the inventory

Returns

All of the items in the inventory as `ItemsInInventory`

Definition at line 159 of file `Inventory.cs`.

```
00160     {
00161         return items;
00162     }
```

6.29.2.5 InventorySet()

```
bool BeeGame.Inventory.Inventory.InventorySet ( )
```

Is the inventory set?

Returns

true if `items` == null

Definition at line 52 of file `Inventory.cs`.

```
00053     {
00054         if (items == null)
00055             return true;
00056         return false;
00058     }
```

6.29.2.6 PutItemsInSlots()

```
virtual void BeeGame.Inventory.Inventory.PutItemsInSlots ( ) [virtual]
```

Sets an Item in the `ItemsInInventory.itemsInInventory` array to a `InventorySlot.item`

Definition at line 144 of file `Inventory.cs`.

```
00145     {
00146         /* goes through all of the items in the array setting them all to a slot
00147         for (int i = 0; i < slots.Length; i++)
00148         {
00149             slots[i].slotIndex = i;
00150             slots[i].myInventory = this;
00151             slots[i].item = items.itemsInInventory[i];
00152         }
00153     }
```

6.29.2.7 SaveInv()

```
virtual void BeeGame.Inventory.Inventory.SaveInv ( ) [virtual]
```

Saves the inventory

Used when closing a chest so the changes to the player inventory are saved

Reimplemented in [BeeGame.Inventory.BlockInventory.CraftingTableInventory](#), and [BeeGame.Inventory.BeeAlyzerInventory](#).

Definition at line 131 of file [Inventory.cs](#).

```
00132     {
00133         Serialization.Serialization.SerializeInventory(this, inventoryName);
00134     }
```

6.29.2.8 SetAllItems()

```
void BeeGame.Inventory.Inventory.SetAllItems (
    ItemsInInventory items )
```

Sets the `items` to the given `ItemsInInventory`

Parameters

<code>items</code>	<code>Items</code> to set this inventory to
--------------------	---

remarks> Used during deserialization to restore the inventory /remarks>

Definition at line 76 of file [Inventory.cs](#).

```
00077     {
00078         this.items = items;
00079     }
```

6.29.2.9 SetInventorySize()

```
void BeeGame.Inventory.Inventory.SetInventorySize (
    int inventorySize )
```

Sets the inventory size to the number of slots in the inventory

Parameters

<code>inventorySize</code>	
----------------------------	--

Definition at line 64 of file [Inventory.cs](#).

```
00065      {
00066          items = new ItemsInInventory(slots.Length);
00067      }
```

6.29.2.10 SetItemInventory()

```
virtual void BeeGame.Inventory.Inventory.SetItemInventory (
    Item [] items) [virtual]
```

Definition at line 136 of file [Inventory.cs](#).

```
00137      {
00138
00139      }
```

6.29.2.11 ToggleInventory()

```
virtual void BeeGame.Inventory.Inventory.ToggleInventory (
    Inventory inv) [virtual]
```

Reimplemented in [BeeGame.Inventory.BlockInventory.CraftingTableInventory](#), [BeeGame.Inventory.ChestInventory](#), [BeeGame.Inventory.QuestBookInventory](#), and [BeeGame.Inventory.BeeAlyzerInventory](#).

Definition at line 120 of file [Inventory.cs](#).

```
00121      {
00122          throw new NotImplementedException();
00123      }
```

6.29.2.12 UpdateBase()

```
void BeeGame.Inventory.Inventory.UpdateBase () [protected]
```

Things in the inventory that should be updated

Definition at line 86 of file [Inventory.cs](#).

```
00087      {
00088          PutItemsInSlots ();
00089          DrawItemAtCursor ();
00090      }
```

6.29.3 Member Data Documentation

6.29.3.1 floatingItem

`Item BeeGame.Inventory.Inventory.floatingItem [package]`

Item that is currently being moved

Definition at line 25 of file [Inventory.cs](#).

6.29.3.2 inventoryName

`string BeeGame.Inventory.Inventory.inventoryName = ""`

Name of this inventory

Definition at line 29 of file [Inventory.cs](#).

6.29.3.3 items

`ItemsInInventory BeeGame.Inventory.Inventory.items`

Items in the inventory

Definition at line 17 of file [Inventory.cs](#).

6.29.3.4 myblock

`Blocks.Block BeeGame.Inventory.Inventory.myblock`

The block class that this inventory is part of

currently only used for the [Blocks.Apiary](#) but could be used so that block inventories are stored in the chunk and not in a separate file

Definition at line 44 of file [Inventory.cs](#).

6.29.3.5 slots

`InventorySlot [] BeeGame.Inventory.Inventory.slots`

Slots in the inventory

Definition at line 21 of file [Inventory.cs](#).

6.29.3.6 spriteAtCursor

```
GameObject BeeGame.Inventory.Inventory.spriteAtCursor [private]
```

The sprite at the cursor

Definition at line 37 of file [Inventory.cs](#).

6.29.3.7 thisInventoryOpen

```
bool BeeGame.Inventory.Inventory.thisInventoryOpen = false [protected]
```

is this inventory open?

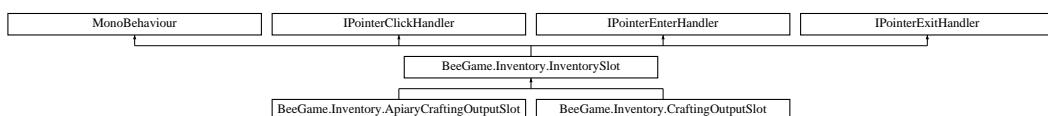
Definition at line 33 of file [Inventory.cs](#).

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/[Inventory.cs](#)

6.30 BeeGame.Inventory.InventorySlot Class Reference

Inheritance diagram for BeeGame.Inventory.InventorySlot:



Public Member Functions

- virtual void [OnPointerClick](#) (PointerEventData eventData)
Allows the player to interact with the item slot
- void [OnPointerEnter](#) (PointerEventData eventData)
Makes the text object when the cursor is over the slot
- void [OnPointerExit](#) (PointerEventData eventData)
Destroys the text object when the cursor is not over the slot anymore

Public Attributes

- [Item item](#)
The item this slot has in it
- [Inventory myInventory](#)
The Inventory this slot is in
- [GameObject itemText](#)
If the slot currently has the item text object made this will be not null otherwise it is null
- bool [selectedSlot](#) = false
Is this slot currently the selected slot in the hotbar?
- bool [itemsCanBeInserted](#) = true
Can items be inserted into this slot by the player

Protected Member Functions

- void [Update \(\)](#)

Updates the slot

Package Functions

- void [UpdateIcon \(\)](#)

Applies the correct icon to the slot depending on what is in the slot

- void [CheckItem \(\)](#)

checks that the item is valid

Package Attributes

- int [slotIndex](#)

The slot in the inventory this is

Private Member Functions

- void [AddToFloatingItem \(\)](#)

Add items from the slot to the [Inventory.floatingItem](#)

- void [AddToSlot \(int numerToAdd\)](#)

Adds a number to items into the slot

- void [SplitStack \(\)](#)

Halves a [Item.itemStackCount](#) between the slot and the [Inventory.floatingItem](#)

- void [SwapItems \(\)](#)

Swaps the [Item](#) in the [Inventory.floatingItem](#) with the slots [item](#)

- void [CheckFloatingItem \(\)](#)

Checks if the [Inventory.floatingItem](#) should be null

- void [OnDisable \(\)](#)

Destroys the [item](#) text when the inventory is closed

6.30.1 Detailed Description

Definition at line 10 of file [InventorySlot.cs](#).

6.30.2 Member Function Documentation

6.30.2.1 AddToFloatingItem()

```
void BeeGame.Inventory.InventorySlot.AddToFloatingItem ( ) [private]
```

Add items from the slot to the [Inventory.floatingItem](#)

Definition at line 173 of file [InventorySlot.cs](#).

```
00174         {
00175             /* if the whole stack can be added do it and move on
00176             if(myInventory.floatingItem.itemStackCount +
00177                 item.itemStackCount <= item.maxStackCount)
00178             {
00179                 myInventory.floatingItem.itemStackCount +=
00180                     item.itemStackCount;
00181                 item = null;
00182                 myInventory.AddItemToSlots(slotIndex,
00183                     item);
00184             }
00185         }
00186
00187         /* if the whole stack cannot be added calculate how many need to be removed from the slots
00188         item.stack
00189             item.itemStackCount -= (item.maxStackCount -
00190                 myInventory.floatingItem.itemStackCount);
00191             /* set the floating item to the max stack count
00192                 myInventory.floatingItem.itemStackCount =
00193                     item.maxStackCount;
00194                 myInventory.AddItemToSlots(slotIndex,
00195                     item);
00196         }
```

6.30.2.2 AddToSlot()

```
void BeeGame.Inventory.InventorySlot.AddToSlot (
    int numerToAdd ) [private]
```

Adds a number to items into the slot

Parameters

<i>numerToAdd</i>	Numebr or items to add to the slot
-------------------	------------------------------------

Definition at line 199 of file [InventorySlot.cs](#).

```
00200         {
00201             /* if the item in the slot is null create it
00202             if (item == null)
00203             {
00204                 item = myInventory.floatingItem.CloneObject();
00205                 item.itemStackCount = 0;
00206             }
00207
00208             /* add to number to add to the stack count
00209             item.itemStackCount += numerToAdd;
00210
00211             /* if the stack count is now larger than it should be dont let it be
00212             if (item.itemStackCount > item.maxStackCount)
00213             {
00214                 item.itemStackCount = item.maxStackCount;
00215             }
00216 }
```

```

00217      /* remove the number if items from the floating item then check the floating item is not null
00218      myInventory.floatingItem.itemStackCount -= numerToAdd;
00219      CheckFloatingItem();
00220      /* save the inventory changes
00221      myInventory.AddItemToSlots(slotIndex,
00222          item);
00223      }

```

6.30.2.3 CheckFloatingItem()

void BeeGame.Inventory.InventorySlot.CheckFloatingItem () [private]

Checks if the `Inventory.floatingItem` should be null

Definition at line 264 of file [InventorySlot.cs](#).

```

00265      {
00266          if(myInventory.floatingItem.itemStackCount <= 0)
00267          {
00268              myInventory.floatingItem = null;
00269          }
00270      }

```

6.30.2.4 CheckItem()

void BeeGame.Inventory.InventorySlot.CheckItem () [package]

checks that the item is valid

Definition at line 276 of file [InventorySlot.cs](#).

```

00277      {
00278          if (item != null && myInventory != null)
00279          {
00280              if (item.itemStackCount == 0 || item.
00281      itemName == "TestItem")
00282              {
00283                  myInventory.items.itemsInInventory[
00284                      slotIndex] = null;
00285              }
00286          }

```

6.30.2.5 OnDisable()

void BeeGame.Inventory.InventorySlot.OnDisable () [private]

Destroys the item text when the inventory is closed

Definition at line 318 of file [InventorySlot.cs](#).

```

00319      {
00320          Destroy(itemText);
00321      }

```

6.30.2.6 OnPointerClick()

```

virtual void BeeGame.Inventory.InventorySlot.OnPointerClick (
    PointerEventData eventData) [virtual]

```

Allows the player to interact with the item slot

Parameters

<i>eventData</i>	Right or Left click
------------------	---------------------

Called by the unity event handler when the slot is clicked on

Reimplemented in [BeeGame.Inventory.ApiaryCraftingOutputSlot](#), and [BeeGame.Inventory.CraftingOutputSlot](#).

Definition at line 82 of file [InventorySlot.cs](#).

```

00083      {
00084          if (myInventory.floatingItem != null)
00085          {
00086              /* Left click moves whole stacks of items
00087              if (eventData.button == PointerEventData.InputButton.Left)
00088              {
00089                  /* If the item in the slot is empty put the floating item into it then clear it and
00090                  /* the slot can have items inserted
00091                  if (item == null && itemsCanBeInserted)
00092                  {
00093                      item = myInventory.floatingItem;
00094                      myInventory.floatingItem = null;
00095                      myInventory.AddItemToSlots(
00096                          slotIndex, item);
00097                      return;
00098                  }
00099                  /* if the items are the same
00100                  if(myInventory.floatingItem == item &&
00101                      itemsCanBeInserted)
00102                  {
00103                      /* if the item in the inventoys stack count + the floating items stack count is
00104                      /* less than the max stack count
00105                      if (myInventory.floatingItem.
00106                          itemStackCount + item.itemStackCount <= item.
00107                          maxStackCount)
00108                      {
00109                          AddToSlot (myInventory.
00110                          floatingItem.itemStackCount);
00111                          return;
00112                      }
00113                  }
00114                  /* if the tiems are the same but items cannot be inserted into the slot add as many
00115                  /* items as you
00116                  /* can from the slot to the floating item
00117                  else if(myInventory.floatingItem ==
00118                      item && !itemsCanBeInserted)
00119                  {
00120                      AddToFloatingItem();
00121                      return;
00122                  }
00123                  /* only if items can be inserted into the slot
00124                  if(itemsCanBeInserted)
00125                      SwapItems();
00126                  return;
00127              }
00128          }
00129          else if(eventData.button == PointerEventData.InputButton.Right)
00130          {
00131              /* if the item in slot is null add 1 from the floating item to it
00132              if(item == null && itemsCanBeInserted)
00133              {
00134                  AddToSlot (1);
00135                  return;
00136              }
00137              /* if the items are the same add 1 from the floating item to this item
00138              else if(item == myInventory.floatingItem &&
00139                      itemsCanBeInserted)
00140              {
                  AddToSlot (1);

```

```

00141             return;
00142         }
00143     }
00144 }
00145 /* if the floating item is null
00146 else
00147 {
00148     /* add 1/2 of the stack into the floating item if right click was pressed
00149     if(eventData.button == PointerEventData.InputButton.Right)
00150     {
00151         SplitStack();
00152
00153         return;
00154     }
00155
00156
00157     if (item == null)
00158         return;
00159
00160     /* otherwise add the items into the floating item slot
00161     myInventory.floatingItem = item.CloneObject();
00162
00163     item.itemStackCount -= item.itemStackCount;
00164
00165     return;
00166 }
00167
00168 }
```

6.30.2.7 OnPointerEnter()

```
void BeeGame.Inventory.InventorySlot.OnPointerEnter (
    PointerEventData eventData )
```

Makes the text object when the cursor is over the slot

Parameters

<code>eventData</code>	Not used but required for the interface
------------------------	---

Definition at line 293 of file [InventorySlot.cs](#).

```

00294     {
00295         /* if the item is null or the floating item has something in it dont display the item text as
00296         it is not necessary
00297         if (item != null && myInventory.floatingItem == null)
00298         {
00299             itemText = Instantiate(PrefabDictionary.
00300                 GetPrefab("ItemDetails"));
00301             /* sets the text to the correct position
00302             itemText.transform.GetChild(0).position = Input.mousePosition;
00303             /* puts the correct text in the box
00304             itemText.transform.GetChild(0).GetChild(0).GetComponent<Text>().text = $""
00305             {item.GetItemName()}\nStack: {item.itemStackCount}";
00306         }
00307     }
```

6.30.2.8 OnPointerExit()

```
void BeeGame.Inventory.InventorySlot.OnPointerExit (
    PointerEventData eventData )
```

Destroys the text object when the cursor is not over the slot anymore

Parameters

<code>eventData</code>	Not used but required for the interface
------------------------	---

Definition at line 310 of file [InventorySlot.cs](#).

```
00311     {
00312         Destroy(itemText);
00313     }
```

6.30.2.9 SplitStack()

```
void BeeGame.Inventory.InventorySlot.SplitStack ( ) [private]
```

Halves a Item.itemStackCount between the slot and the [Inventory.floatingItem](#)

If the stack count is the slot is not an even number more items go to the floating item than go to the slot. This is so that right clicking on a slot when there is only 1 item in it actually make the item in that slot go into the floating item

Definition at line 230 of file [InventorySlot.cs](#).

```
00231     {
00232         myInventory.floatingItem = item.CloneObject();
00233         int give = (item.itemStackCount + 1) / 2;
00234         myInventory.floatingItem.itemStackCount = give;
00235         item.itemStackCount -= give;
00236
00237         if (item.itemStackCount <= 0)
00238             item = null;
00239
00240         myInventory.AddItemToSlots(slotIndex,
00241             item);
00242         Destroy(itemText);
00243     }
```

6.30.2.10 SwapItems()

```
void BeeGame.Inventory.InventorySlot.SwapItems ( ) [private]
```

Swaps the Item in the [Inventory.floatingItem](#) with the slots `item`

Definition at line 247 of file [InventorySlot.cs](#).

```
00248     {
00249         /* temp copy of the item
00250         Item temp = myInventory.floatingItem;
00251         /* sets the floating item
00252         myInventory.floatingItem = item;
00253         /* sets the item that was in the floating item to the item in the slot
00254         item = temp;
00255         /* Saves the changes to the inventory
00256         myInventory.AddItemToSlots(slotIndex,
00257             item);
00258         /* destroys the text as it is not needed anymore
00259         Destroy(itemText);
00260     }
```

6.30.2.11 Update()

```
void BeeGame.Inventory.InventorySlot.Update ( ) [protected]
```

Updates the slot

Definition at line 42 of file [InventorySlot.cs](#).

```
00043     {
00044         CheckItem();
00045         UpdateIcon();
00046     }
```

6.30.2.12 UpdateIcon()

```
void BeeGame.Inventory.InventorySlot.UpdateIcon ( ) [package]
```

Applies the correct icon to the slot depending on what is in the slot

Definition at line 51 of file [InventorySlot.cs](#).

```
00052     {
00053         if(item == null)
00054         {
00055             GetComponent<Image>().sprite = null;
00056         }
00057         else
00058         {
00059             if(!item.Equals(new Item()))
00060                 GetComponent<Image>().sprite = item.GetItemSprite();
00061         }
00062
00063         /* if the slot is selected in the hotbar give the player some indication by colouring it grey
00064         if (selectedSlot)
00065         {
00066             GetComponent<Image>().color = Color.gray;
00067         }
00068         else
00069         {
00070             GetComponent<Image>().color = Color.white;
00071         }
00072     }
```

6.30.3 Member Data Documentation

6.30.3.1 item

`Item` BeeGame.Inventory.InventorySlot.item

The item this slot has in it

Definition at line 20 of file [InventorySlot.cs](#).

6.30.3.2 itemsCanBeInserted

```
bool BeeGame.Inventory.InventorySlot.itemsCanBeInserted = true
```

Can items be inserted into this slot by the player

Definition at line 36 of file [InventorySlot.cs](#).

6.30.3.3 itemText

```
GameObject BeeGame.Inventory.InventorySlot.itemText
```

If the slot currently has the item text object made this will be not null otherwise it is null

Definition at line 28 of file [InventorySlot.cs](#).

6.30.3.4 myInventory

```
Inventory BeeGame.Inventory.InventorySlot.myInventory
```

The [Inventory](#) this slot is in

Definition at line 24 of file [InventorySlot.cs](#).

6.30.3.5 selectedSlot

```
bool BeeGame.Inventory.InventorySlot.selectedSlot = false
```

Is this slot currently the selected slot in the hotbar?

Definition at line 32 of file [InventorySlot.cs](#).

6.30.3.6 slotIndex

```
int BeeGame.Inventory.InventorySlot.slotIndex [package]
```

The slot in the inventory this is

Definition at line 16 of file [InventorySlot.cs](#).

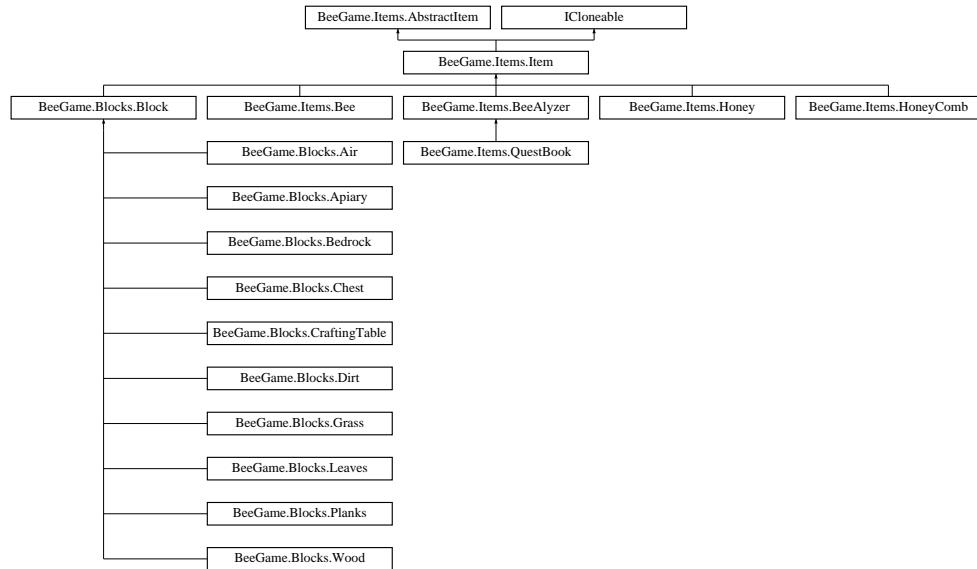
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/[InventorySlot.cs](#)
-

6.31 BeeGame.Items.Item Class Reference

Base class for all [Items](#) and [Blocks](#) in the game

Inheritance diagram for BeeGame.Items.Item:



Public Member Functions

- [`Item \(\)`](#)
- [`Item \(string name\)`](#)
- [`virtual bool InteractWithObject \(\)`](#)
- [`virtual void InteractWithItem \(Inventory.Inventory playerInventory\)`](#)
- [`virtual GameObject GetGameObject \(\)`](#)

Returns the GameObject for the item of it has one

- [`override string GetItemID \(\)`](#)

Returns the id for the item as a string

- [`virtual Sprite GetItemSprite \(\)`](#)

Returns the sprite for the item

- [`override string GetItemName \(\)`](#)

Returns the items name

- [`virtual Tile TexturePosition \(Direction direction\)`](#)

Texture postion of the items texture

- [`virtual MeshData ItemMesh \(int x, int y, int z, MeshData meshData\)`](#)

Returns the mesh for the item

- [`virtual Vector2 \[\] FaceUVs \(Direction direction\)`](#)

Sets the UVs for the given Direction

- [`object Clone \(\)`](#)

Slow try no to use. Instead use Extensions.CloneObject<T>(T)

- [`override string ToString \(\)`](#)

Returns the item name an id formatted nicely

- [`override int GetHashCode \(\)`](#)

Returns the hashcode for the item

- [`override bool Equals \(object obj\)`](#)

Checks if the item is equal to another

Static Public Member Functions

- static bool `operator==` (`Item` a, `Item` b)
Overrides the default == operator as different things need to be checked
- static bool `operator!=` (`Item` a, `Item` b)
Inverse of ==

Public Attributes

- virtual bool `placeable` => false
Is this item placeable. Saves checking if the item is a block type
- int `itemStackCount` = 1
Number of items in the stack
- virtual int `maxStackCount` => 64
Max number of items in a stack

Static Public Attributes

- static int `ID` => 0

Protected Member Functions

- virtual `MeshData FaceDataUp` (int x, int y, int z, `MeshData` meshData, bool addToRenderMesh=true, float blockSize=0.5f)
Adds the Upwards face to the given MeshData
- virtual `MeshData FaceDataDown` (int x, int y, int z, `MeshData` meshData, bool addToRenderMesh=true, float blockSize=0.5f)
Adds the Bottom face to the given MeshData
- virtual `MeshData FaceDataNorth` (int x, int y, int z, `MeshData` meshData, bool addToRenderMesh=true, float blockSize=0.5f)
Adds the North face to the given MeshData
- virtual `MeshData FaceDataEast` (int x, int y, int z, `MeshData` meshData, bool addToRenderMesh=true, float blockSize=0.5f)
Adds the East face to the given MeshData
- virtual `MeshData FaceDataSouth` (int x, int y, int z, `MeshData` meshData, bool addToRenderMesh=true, float blockSize=0.5f)
Adds the South face to the given MeshData
- virtual `MeshData FaceDataWest` (int x, int y, int z, `MeshData` meshData, bool addToRenderMesh=true, float blockSize=0.5f)
Adds the West face to the given MeshData

Properties

- string `itemName` [get, set]
Name of the item
- bool `usesGameObject` [get, set]
Does the item use a gameobject

Private Attributes

- const float tileSize = 0.1f

How big are the texture tiles in the texture map (1/tile number x)

6.31.1 Detailed Description

Base class for all [Items](#) and [Blocks](#) in the game

Definition at line [16](#) of file [Item.cs](#).

6.31.2 Constructor & Destructor Documentation

6.31.2.1 Item() [1/2]

```
BeeGame.Items.Item()
```

Definition at line [50](#) of file [Item.cs](#).

```
00051     {
00052         itemName = "TestItem";
00053     }
```

6.31.2.2 Item() [2/2]

```
BeeGame.Items.Item(
    string name)
```

Definition at line [55](#) of file [Item.cs](#).

```
00056     {
00057         itemName = name;
00058     }
```

6.31.3 Member Function Documentation

6.31.3.1 Clone()

```
object BeeGame.Items.Item.Clone ( )
```

Slow try no to use. Instead use Extensions.CloneObject<T>(T)

Returns

A deep copy of this

Definition at line 334 of file [Item.cs](#).

```
00335      {
00336          /* Saves this to a file then reads it back so that a copy and not a reference is passed
00337          BinaryFormatter bf = new BinaryFormatter();
00338          MemoryStream ms = new MemoryStream();
00339
00340          bf.Serialize(ms, this);
00341          ms.Seek(0, SeekOrigin.Begin);
00342
00343          return bf.Deserialize(ms);
00344      }
```

6.31.3.2 Equals()

```
override bool BeeGame.Items.Item.Equals (
    object obj )
```

Checks if the item is equal to another

Parameters

<i>obj</i>	object to check against
------------	-------------------------

Returns

true if items are the same

Definition at line 371 of file [Item.cs](#).

```
00372      {
00373          if (!(obj is Item))
00374              return false;
00375
00376          return this == (obj as Item);
00377      }
```

6.31.3.3 FaceDataDown()

```
virtual MeshData BeeGame.Items.Item.FaceDataDown (
    int x,
    int y,
    int z,
    MeshData meshData,
    bool addToRenderMesh = true,
    float blockSize = 0.5f ) [protected], [virtual]
```

Adds the Bottom face to the given MeshData

Parameters

<i>x</i>	X pos of the item
<i>y</i>	Y pos of the item
<i>z</i>	Z pos of the item
<i>meshData</i>	MeshData to add the face to
<i>addToRenderMesh</i>	Should the mesh be added to the render mesh (default true)
<i>blockSize</i>	how big is the item

Returns

Given MeshData with the face data added

Definition at line 198 of file [Item.cs](#).

```
00199      {
00200          /* Adds vertices in a anti-clockwise order
00201          meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z -
00202              blockSize), addToRenderMesh);
00203          meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z -
00204              blockSize), addToRenderMesh);
00205          meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z +
00206              blockSize), addToRenderMesh);
00207          meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z +
00208              blockSize), addToRenderMesh);
00209          /* adds teh tirs for the quad
00210          meshData.AddQuadTriangles(addToRenderMesh);
00211          /* if the data should be added to the render mesh also add the uvs to the mesh
00212          if (addToRenderMesh)
00213              meshData.uv.AddRange(FaceUVs.Direction.DOWN));
00214      return meshData;
00215 }
```

6.31.3.4 FaceDataEast()

```
virtual MeshData BeeGame.Items.Item.FaceDataEast (
    int x,
    int y,
    int z,
    MeshData meshData,
    bool addToRenderMesh = true,
    float blockSize = 0.5f ) [protected], [virtual]
```

Adds the East face to the given MeshData

Parameters

<i>x</i>	X pos of the item
<i>y</i>	Y pos of the item
<i>z</i>	Z pos of the item
<i>meshData</i>	MeshData to add the face to
<i>addToRenderMesh</i>	Should the mesh be added to the render mesh (default true)
<i>blockSize</i>	how big is the item

Returns

Given MeshData with the face data added

Definition at line 254 of file [Item.cs](#).

```

00255      {
00256          /* Adds vertices in a anti-clockwise order
00257          meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z -
00258              blockSize), addToRenderMesh);
00259          meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z -
00260              blockSize), addToRenderMesh);
00261          meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z +
00262              blockSize), addToRenderMesh);
00263          meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z +
00264              blockSize), addToRenderMesh);
00265          /* adds teh tirs for the quad
00266          meshData.AddQuadTriangles(addToRenderMesh);
00267          /* if the data should be added to the render mesh also add the uvs to the mesh
00268          if (addToRenderMesh)
00269              meshData.uv.AddRange(FaceUVs.Direction.EAST));
00270      }

```

6.31.3.5 FaceDataNorth()

```

virtual MeshData BeeGame.Items.Item.FaceDataNorth (
    int x,
    int y,
    int z,
    MeshData meshData,
    bool addToRenderMesh = true,
    float blockSize = 0.5f ) [protected], [virtual]

```

Adds the North face to the given MeshData

Parameters

<i>x</i>	X pos of the item
<i>y</i>	Y pos of the item
<i>z</i>	Z pos of the item
<i>meshData</i>	MeshData to add the face to
<i>addToRenderMesh</i>	Should the mesh be added to the render mesh (default true)
<i>blockSize</i>	how big is the item

Returns

Given MeshData with the face data added

Definition at line 226 of file [Item.cs](#).

```

00227      {
00228          /* Adds vertices in a anti-clockwise order
00229          meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z +
00230              blockSize), addToRenderMesh);
00230          meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z +
00231              blockSize), addToRenderMesh);
00231          meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z +
00232              blockSize), addToRenderMesh);
00232          meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z +
00233              blockSize), addToRenderMesh);
00233
00234          /* adds teh tirs for the quad
00235          meshData.AddQuadTriangles(addToRenderMesh);
00236
00237          /* if the data should be added to the render mesh also add the uvs to the mesh
00238          if (addToRenderMesh)
00239              meshData.uv.AddRange(FaceUVs.Direction.NORTH));
00240
00241      return meshData;
00242 }
```

6.31.3.6 FaceDataSouth()

```
virtual MeshData BeeGame.Items.Item.FaceDataSouth (
    int x,
    int y,
    int z,
    MeshData meshData,
    bool addToRenderMesh = true,
    float blockSize = 0.5f ) [protected], [virtual]
```

Adds the South face to the given MeshData

Parameters

<i>x</i>	X pos of the item
<i>y</i>	Y pos of the item
<i>z</i>	Z pos of the item
<i>meshData</i>	MeshData to add the face to
<i>addToRenderMesh</i>	Should the mesh be added to the render mesh (default true)
<i>blockSize</i>	how big is the item

Returns

Given MeshData with the face data added

Definition at line 282 of file [Item.cs](#).

```

00283      {
00284          /* Adds vertices in a anti-clockwise order
00285          meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z -
blockSize), addToRenderMesh);
```

```

00286         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z -
00287 blockSize), addToRenderMesh);
00288         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z -
00289 blockSize), addToRenderMesh);
00290         meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z -
00291 blockSize), addToRenderMesh);
00292
00293         /* adds teh tirs for the quad
00294         meshData.AddQuadTriangles(addToRenderMesh);
00295
00296         /* if the data should be added to the render mesh also add the uvs to the mesh
00297         if (addToRenderMesh)
00298             meshData.uv.AddRange(FaceUVs(Direction.SOUTH));
00299
00300         return meshData;
00301     }

```

6.31.3.7 FaceDataUp()

```

virtual MeshData BeeGame.Items.Item.FaceDataUp (
    int x,
    int y,
    int z,
    MeshData meshData,
    bool addToRenderMesh = true,
    float blockSize = 0.5f ) [protected], [virtual]

```

Adds the Upwards face to the given MeshData

Parameters

<i>x</i>	X pos of the item
<i>y</i>	Y pos of the item
<i>z</i>	Z pos of the item
<i>meshData</i>	MeshData to add the face to
<i>addToRenderMesh</i>	Should the mesh be added to the render mesh (default true)
<i>blockSize</i>	how big is the item

Returns

Given MeshData with the face data added

Definition at line 170 of file [Item.cs](#).

```

00171     {
00172         /* Adds vertices in a anti-clockwise order
00173         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z +
00174 blockSize), addToRenderMesh, Direction.UP);
00175         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z +
00176 blockSize), addToRenderMesh, Direction.UP);
00177         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z -
00178 blockSize), addToRenderMesh, Direction.UP);
00179         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z -
00180 blockSize), addToRenderMesh, Direction.UP);
00181
00182         /* adds teh tirs for the quad
00183         meshData.AddQuadTriangles(addToRenderMesh);
00184
00185         /* if the data should be added to the render mesh also add the uvs to the mesh
00186         if (addToRenderMesh)
00187             meshData.uv.AddRange(FaceUVs(Direction.UP));
00188
00189         return meshData;
00190     }

```

6.31.3.8 FaceDataWest()

```
virtual MeshData BeeGame.Items.Item.FaceDataWest (
    int x,
    int y,
    int z,
    MeshData meshData,
    bool addToRenderMesh = true,
    float blockSize = 0.5f ) [protected], [virtual]
```

Adds the West face to the given MeshData

Parameters

<i>x</i>	X pos of the item
<i>y</i>	Y pos of the item
<i>z</i>	Z pos of the item
<i>meshData</i>	MeshData to add the face to
<i>addToRenderMesh</i>	Should the mesh be added to the render mesh (default true)
<i>blockSize</i>	how big is the item

Returns

Given MeshData with the face data added

Definition at line 310 of file [Item.cs](#).

```
00311      {
00312          /* Adds vertices in a anti-clockwise order
00313          meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z +
00314              blockSize), addToRenderMesh);
00314          meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z +
00315              blockSize), addToRenderMesh);
00315          meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z -
00316              blockSize), addToRenderMesh);
00316          meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z -
00317              blockSize), addToRenderMesh);
00317
00318          /* adds teh tirs for the quad
00319          meshData.AddQuadTriangles(addToRenderMesh);
00320
00321          /* if the data should be added to the render mesh also add the uvs to the mesh
00322          if (addToRenderMesh)
00323              meshData.uv.AddRange(FaceUVs(Direction.WEST));
00324
00325      return meshData;
00326 }
```

6.31.3.9 FaceUVs()

```
virtual Vector2 [] BeeGame.Items.Item.FaceUVs (
    Direction direction ) [virtual]
```

Sets the UVs for the given Direction

Parameters

<i>direction</i>	Direction to add the texture
------------------	------------------------------

Returns

Array of Vector2 to add to the UVs

Definition at line 145 of file [Item.cs](#).

```
00146      {
00147          /* only 4 uvs per face
00148          Vector2[] UVs = new Vector2[4];
00149          Tile tilePos = TexturePosition(direction);
00150
00151          /* sets the UVs for each vertex
00152          UVs[0] = new THVector2(tileSize * tilePos.x +
00153          tileSize - 0.01f, tileSize * tilePos.y + 0.01f);
00154          UVs[1] = new THVector2(tileSize * tilePos.x +
00155          tileSize - 0.01f, tileSize * tilePos.y + tileSize - 0.01f);
00156          UVs[2] = new THVector2(tileSize * tilePos.x + 0.01f,
00157          tileSize * tilePos.y + tileSize - 0.01f);
00158          UVs[3] = new THVector2(tileSize * tilePos.x + 0.01f,
00159          tileSize * tilePos.y + 0.01f);
00160
00161          return UVs;
00162      }
```

6.31.3.10 GetGameObject()

```
virtual GameObject BeeGame.Items.Item.GetGameObject () [virtual]
```

Returns the GameObject for the item if it has one

Returns

GameObject for the item

Reimplemented in [BeeGame.Items.Bee](#), [BeeGame.Blocks.CraftingTable](#), [BeeGame.Items.HoneyComb](#), [BeeGame.Blocks.Chest](#), and [BeeGame.Blocks.Apiary](#).

Definition at line 78 of file [Item.cs](#).

```
00078 { return null; }
```

6.31.3.11 GetHashCode()

```
override int BeeGame.Items.Item.GetHashCode () [virtual]
```

Returns the hashcode for the item

Returns

1

Implements [BeeGame.Items.AbstractItem](#).

Reimplemented in [BeeGame.Items.QuestBook](#).

Definition at line 361 of file [Item.cs](#).

```
00362      {
00363          return ID;
00364      }
```

6.31.3.12 GetItemID()

```
override string BeeGame.Items.Item.GetItemID () [virtual]
```

Returns the id for the item as a string

Returns

Implements [BeeGame.Items.AbstractItem](#).

Definition at line 84 of file [Item.cs](#).

```
00085      {
00086          return $"{GetHashCode()}";
00087      }
```

6.31.3.13 GetItemName()

```
override string BeeGame.Items.Item.GetItemName () [virtual]
```

Returns the items name

Returns

Implements [BeeGame.Items.AbstractItem](#).

Definition at line 102 of file [Item.cs](#).

```
00103      {
00104          return $"{itemName}";
00105      }
```

6.31.3.14 GetItemSprite()

```
virtual Sprite BeeGame.Items.Item.GetItemSprite () [virtual]
```

Returns the sprite for the item

Returns

Sprite for this item

Reimplemented in [BeeGame.Blocks.CraftingTable](#), [BeeGame.Items.BeeAlyzer](#), [BeeGame.Blocks.Chest](#), [BeeGame.Items.Bee](#), [BeeGame.Blocks.Apiary](#), [BeeGame.Items.HoneyComb](#), [BeeGame.Items.QuestBook](#), [BeeGame.Blocks.Block](#), [BeeGame.Blocks.Grass](#), [BeeGame.Blocks.Dirt](#), [BeeGame.Blocks.Planks](#), [BeeGame.Blocks.Wood](#), and [BeeGame.Blocks.Leaves](#).

Definition at line 93 of file [Item.cs](#).

```
00094      {
00095          return SpriteDictionary.GetSprite("TestSprite");
00096      }
```

6.31.3.15 InteractWithItem()

```
virtual void BeeGame.Items.Item.InteractWithItem (
    Inventory.Inventory playerInventory ) [virtual]
```

Definition at line 67 of file [Item.cs](#).

```
00068      {
00069          return;
00070      }
```

6.31.3.16 InteractWithObject()

```
virtual bool BeeGame.Items.Item.InteractWithObject ( ) [virtual]
```

Reimplemented in [BeeGame.Items.BeeAlyzer](#).

Definition at line 62 of file [Item.cs](#).

```
00063      {
00064          return false;
00065      }
```

6.31.3.17 ItemMesh()

```
virtual MeshData BeeGame.Items.Item.ItemMesh (
    int x,
    int y,
    int z,
    MeshData meshData ) [virtual]
```

Returns the mesh for the item

Parameters

<i>x</i>	X pos if the item
<i>y</i>	Y pos if the item
<i>z</i>	Z pos if the item
<i>meshData</i>	data to add the mesh to

Returns

given MeshData with the items mesh added

Definition at line 127 of file [Item.cs](#).

```
00128      {
00129          /* adds all faces of the item to the mesh as all faces could be seen at any time
```

```

00130     meshData = FaceDataUp(x, y, z, meshData, true, 0.25f);
00131     meshData = FaceDataDown(x, y, z, meshData, true, 0.25f);
00132     meshData = FaceDataNorth(x, y, z, meshData, true, 0.25f);
00133     meshData = FaceDataEast(x, y, z, meshData, true, 0.25f);
00134     meshData = FaceDataSouth(x, y, z, meshData, true, 0.25f);
00135     meshData = FaceDataWest(x, y, z, meshData, true, 0.25f);
00136
00137     return meshData;
00138 }
```

6.31.3.18 operator"!=()

```
static bool BeeGame.Items.Item.operator!= (
    Item a,
    Item b ) [static]
```

Inverse of ==

Parameters

a	Item
b	Item

Returns

True if $a \neq b$

Definition at line 404 of file [Item.cs](#).

```

00405     {
00406         return !(a == b);
00407     }
```

6.31.3.19 operator==()

```
static bool BeeGame.Items.Item.operator== (
    Item a,
    Item b ) [static]
```

Overrides the default == operator as different things need to be checked

Parameters

a	Item
b	Item

Returns

true if $a == b$

Definition at line 385 of file [Item.cs](#).

```

00386      {
00387          if (ReferenceEquals(a, null) && ReferenceEquals(b, null))
00388              return true;
00389          if (ReferenceEquals(a, null) || ReferenceEquals(b, null))
00390              return false;
00391          if(a.GetItemID() == b.GetItemID())
00392              return true;
00393          return false;
00394      }

```

6.31.3.20 TexturePosition()

```
virtual Tile BeeGame.Items.Item.TexturePosition (
    Direction direction ) [virtual]
```

Texture postion of the items texture

Parameters

<i>direction</i>	Direction for the texture
------------------	---------------------------

Returns

Position of the texture

Reimplemented in [BeeGame.Blocks.CraftingTable](#), [BeeGame.Blocks.Chest](#), [BeeGame.Blocks.Apiary](#), [BeeGame.Blocks.Grass](#), [BeeGame.Blocks.Bedrock](#), [BeeGame.Blocks.Dirt](#), [BeeGame.Blocks.Planks](#), [BeeGame.Blocks.Wood](#), and [BeeGame.Blocks.Leaves](#).

Definition at line 114 of file [Item.cs](#).

```

00115      {
00116          return new Tile() { x = 1, y = 9 };
00117      }

```

6.31.3.21 ToString()

```
override string BeeGame.Items.Item.ToString ( )
```

Returns the item name an id formatted nicely

Returns

Definition at line 352 of file [Item.cs](#).

```

00353      {
00354          return $"{itemName} \nID: {GetItemID()}";
00355      }

```

6.31.4 Member Data Documentation

6.31.4.1 ID

```
int BeeGame.Items.Item.ID => 0 [static]
```

Definition at line [46](#) of file [Item.cs](#).

6.31.4.2 itemStackCount

```
int BeeGame.Items.Item.itemStackCount = 1
```

Number of items in the stack

Definition at line [39](#) of file [Item.cs](#).

6.31.4.3 maxStackCount

```
virtual int BeeGame.Items.Item.maxStackCount => 64
```

Max number of items in a stack

Definition at line [44](#) of file [Item.cs](#).

6.31.4.4 placeable

```
virtual bool BeeGame.Items.Item.placeable => false
```

Is this item placeable. Saves checking if the item is a block type

Definition at line [26](#) of file [Item.cs](#).

6.31.4.5 tileSize

```
const float BeeGame.Items.Item.tileSize = 0.1f [private]
```

How big are the texture tiles in the texture map (1/tile number x)

Definition at line [34](#) of file [Item.cs](#).

6.31.5 Property Documentation

6.31.5.1 itemName

```
string BeeGame.Items.Item.itemName [get], [set], [package]
```

Name of the item

Definition at line 22 of file [Item.cs](#).

6.31.5.2 usesGameObject

```
bool BeeGame.Items.Item.usesGameObject [get], [set]
```

Does the item use a gameobject

Definition at line 30 of file [Item.cs](#).

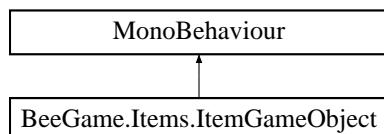
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/[Item.cs](#)

6.32 BeeGame.Items.ItemGameObject Class Reference

Interface between item and unity gameobjects

Inheritance diagram for BeeGame.Items.ItemGameObject:



Public Attributes

- [Item item](#)
Item that this gameobject repersents
- [GameObject go](#)
GameObject to make

Private Member Functions

- void [Start \(\)](#)
Makes the mesh or instantiates the items gameobject
- void [Update \(\)](#)
Destroys the game object if it falls to low
- void [MakeMesh \(\)](#)
Makes the items mesh

6.32.1 Detailed Description

Interface between item and unity gameobjects

Definition at line 18 of file [ItemGameObject.cs](#).

6.32.2 Member Function Documentation

6.32.2.1 MakeMesh()

```
void BeeGame.Items.ItemGameObject.MakeMesh () [private]
```

Makes the items mesh

Definition at line 58 of file [ItemGameObject.cs](#).

```
00059      {
00060          MeshData meshData = new MeshData();
00061          if(item != null)
00062              meshData = item.ItemMesh(0, 0, 0, meshData);
00063
00064          Mesh mesh = new Mesh()
00065          {
00066              vertices = meshData.verts.ToArray(),
00067              triangles = meshData.tris.ToArray(),
00068              uv = meshData.uv.ToArray()
00069          };
00070
00071          mesh.RecalculateNormals();
00072
00073          GetComponent<MeshFilter>().mesh = mesh;
00074      }
```

6.32.2.2 Start()

```
void BeeGame.Items.ItemGameObject.Start () [private]
```

Makes the mesh or instantiates the items gameobject

Definition at line 32 of file [ItemGameObject.cs](#).

```
00033      {
00034          if (!item.usesGameObject)
00035              MakeMesh();
00036
00037          if (item.usesGameObject)
00038          {
00039              Instantiate(item.GetGameObject(), transform, false);
00040              transform.localScale = new Vector3(0.5f, 0.5f, 0.5f);
00041          }
00042      }
```

6.32.2.3 Update()

```
void BeeGame.Items.ItemGameObject.Update ( ) [private]
```

Destroys the game object if it falls to low

Definition at line 47 of file [ItemGameObject.cs](#).

```
00048     {
00049         if (transform.position.y < -100)
00050     {
00051         Destroy(gameObject);
00052     }
00053 }
```

6.32.3 Member Data Documentation

6.32.3.1 go

```
GameObject BeeGame.Items.ItemGameObject.go
```

GameObject to make

Definition at line 27 of file [ItemGameObject.cs](#).

6.32.3.2 item

```
Item BeeGame.Items.ItemGameObject.item
```

Item that this gameobject repersents

Definition at line 23 of file [ItemGameObject.cs](#).

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/GameObjectStuff/[ItemGameObject.cs](#)

6.33 BeeGame.Inventory.ItemsInInventory Class Reference

Class that holds all of the items in the inventory. Can be serialized so inventory may be saved

Public Member Functions

- [ItemsInInventory](#) (int numberofInventorySlots)
Sets the size of the inventory
- void [AddItem](#) (int index, [Item](#) item)
Add an Item to a specific index in the inventory
- bool [AddItem](#) ([Item](#) item)
Adds a Item to the inventory

Public Attributes

- [Item \[\] itemsInInventory](#)

All of the items in the inventory

6.33.1 Detailed Description

Class that holds all of the items in the inventory. Can be serialized so inventory may be saved

Definition at line 10 of file [ItemsInInventory.cs](#).

6.33.2 Constructor & Destructor Documentation

6.33.2.1 ItemsInInventory()

```
BeeGame.Inventory.ItemsInInventory.ItemsInInventory (
    int numberOfInventorySlots )
```

Sets the size of the inventory

Parameters

<code>numberOfInventorySlots</code>	<input type="button" value=""/>
-------------------------------------	---------------------------------

Definition at line 21 of file [ItemsInInventory.cs](#).

```
00022     {
00023         itemsInInventory = new Item[numberOfInventorySlots];
00024     }
```

6.33.3 Member Function Documentation

6.33.3.1 AddItem() [1/2]

```
void BeeGame.Inventory.ItemsInInventory.AddItem (
    int index,
    Item item )
```

Add an Item to a specific index in the inventory

Parameters

<code>index</code>	Were to add the item
<code>item</code>	What Item to put in the inventory

Definition at line 31 of file [ItemsInInventory.cs](#).

```
00032     {
00033         itemsInInventory[index] = item;
00034     }
```

6.33.3.2 AddItem() [2/2]

```
bool BeeGame.Inventory.ItemsInInventory.AddItem (
    Item item )
```

Adds a Item to the inventory

Parameters

<i>item</i>	Item to add
-------------	-------------

Returns

true if *item* was added to the inventory

Definition at line 41 of file [ItemsInInventory.cs](#).

```
00042     {
00043         for (int i = 0; i < itemsInInventory.Length; i++)
00044     {
00045         if (itemsInInventory[i] == null)
00046         {
00047             itemsInInventory[i] = item;
00048             return true;
00049         }
00050         if (itemsInInventory[i] == item &&
    itemsInInventory[i].itemStackCount + 1 <= itemsInInventory[i].maxStackCount
    )
00051         {
00052             itemsInInventory[i].itemStackCount++;
00053             return true;
00054         }
00055     }
00056     return false;
00058 }
```

6.33.4 Member Data Documentation

6.33.4.1 itemsInInventory

`Item [] BeeGame.Inventory.ItemsInInventory.itemsInInventory`

All of the items in the inventory

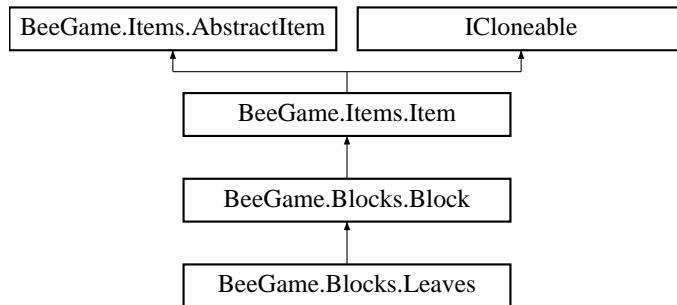
Definition at line 15 of file [ItemsInInventory.cs](#).

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/[ItemsInInventory.cs](#)

6.34 BeeGame.Blocks.Leaves Class Reference

Inheritance diagram for BeeGame.Blocks.Leaves:



Public Member Functions

- [Leaves \(\)](#)
- [override Sprite GetItemSprite \(\)](#)
Returns the sprite for the item
- [override Tile TexturePosition \(Direction direction\)](#)
Texture position of the items texture
- [override bool IsSolid \(Direction direction\)](#)
What Directions is this Block solid in?
- [override int GetHashCode \(\)](#)
Base ID of the block
- [override string ToString \(\)](#)
Returns the name and ID of the block as a string

Static Public Attributes

- [static new int ID => 6](#)

Additional Inherited Members

6.34.1 Detailed Description

Definition at line 10 of file [Leaves.cs](#).

6.34.2 Constructor & Destructor Documentation

6.34.2.1 Leaves()

`BeeGame.Blocks.Leaves.Leaves ()`

Definition at line 14 of file [Leaves.cs](#).

```

00014             : base("Leaves")
00015         {
00016     }
00017 }
```

6.34.3 Member Function Documentation

6.34.3.1 GetHashCode()

```
override int BeeGame.Blocks.Leaves.GetHashCode ( ) [virtual]
```

Base ID of the block

Returns

5

Reimplemented from [BeeGame.Blocks.Block](#).

Definition at line 41 of file [Leaves.cs](#).

```
00042     {
00043         return ID;
00044     }
```

6.34.3.2 GetItemSprite()

```
override Sprite BeeGame.Blocks.Leaves.GetItemSprite ( ) [virtual]
```

Returns the sprite for the item

Returns

Sprite for this item

Reimplemented from [BeeGame.Blocks.Block](#).

Definition at line 20 of file [Leaves.cs](#).

```
00021     {
00022         return SpriteDictionary.GetSprite("Leaves");
00023     }
```

6.34.3.3 IsSolid()

```
override bool BeeGame.Blocks.Leaves.IsSolid (
    Direction direction ) [virtual]
```

What Directions is this [Block](#) solid in?

Parameters

<i>direction</i>	Direction to check
------------------	--------------------

Returns

Default returns true for all sides

Reimplemented from [BeeGame.Blocks.Block](#).

Definition at line 31 of file [Leaves.cs](#).

```
00032     {
00033         return false;
00034     }
```

6.34.3.4 TexturePosition()

```
override Tile BeeGame.Blocks.Leaves.TexturePosition (
    Direction direction ) [virtual]
```

Texture position of the items texture

Parameters

<i>direction</i>	Direction for the texture
------------------	---------------------------

Returns

Position of the texture

Reimplemented from [BeeGame.Items.Item](#).

Definition at line 26 of file [Leaves.cs](#).

```
00027     {
00028         return new Tile() { x = 5, y = 9 };
00029     }
```

6.34.3.5 ToString()

```
override string BeeGame.Blocks.Leaves.ToString ( )
```

Returns the name and ID of the block as a string

Returns

A nicely formatted string

Definition at line 50 of file [Leaves.cs](#).

```
00051     {
00052         return $"{itemName} \nID: {GetItemID()}";
00053     }
```

6.34.4 Member Data Documentation

6.34.4.1 ID

```
new int BeeGame.Blocks.Leaves.ID => 6 [static]
```

Definition at line 12 of file [Leaves.cs](#).

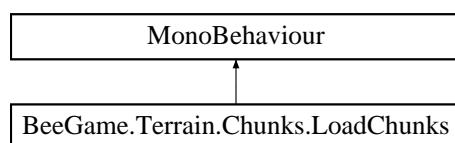
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/[Leaves.cs](#)

6.35 BeeGame.Terrain.Chunks.LoadChunks Class Reference

Loads the [Chunks](#) around the player

Inheritance diagram for BeeGame.Terrain.Chunks.LoadChunks:



Public Attributes

- [World world](#)

The world the player is in

Private Member Functions

- void [Start \(\)](#)
Sets the world
- void [Update \(\)](#)
Builds, Renders, and Removes Chunks
- void [ApplyCollisionMeshToNearbyChunks \(\)](#)
Makes a collision mesh for the Chunks nearest to the player to reduce lag created by PhysX mesh baking
- void [LoadAndRenderChunks \(\)](#)
Gets the chunks that should be built and renders them
- void [FindChunksToLoad \(\)](#)
Finds the Chunks that should be rendered
- void [BuildChunk \(ChunkWorldPos pos\)](#)
Makes a chunk in the given position if it does not already exist
- bool [DeleteChunks \(\)](#)
Destroys Chunks every 10 calls

Private Attributes

- List< ChunkWorldPos > buildList = new List<ChunkWorldPos>()
List if chunks to build

Static Private Attributes

- static ChunkWorldPos [] chunkPositions
Positions to make chunks around the player ///
- static ChunkWorldPos [] nearbyChunks
Chunks in a 3x3 radius around the player that should have a collision mesh
- static int timer = 0
Timer for chunk removal

6.35.1 Detailed Description

Loads the [Chunks](#) around the player

Definition at line 11 of file [LoadChunks.cs](#).

6.35.2 Member Function Documentation

6.35.2.1 ApplyCollisionMeshToNearbyChunks()

```
void BeeGame.Terrain.Chunks.LoadChunks.ApplyCollisionMeshToNearbyChunks ( ) [private]
```

Makes a collision mesh for the [Chunks](#) nearest to the player to reduce lag created by PhysX mesh baking

We don't need to worry about removing [Chunk](#) collision meshes as once PhysX has baked them they have minimal performance impact Doing things this way also spreads out the PhysX mesh baking

Definition at line 111 of file [LoadChunks.cs](#).

```
00112      {
00113          /* gets the player position in chunk coordinates
00114          ChunkWorldPos playerPos = new ChunkWorldPos(Mathf.FloorToInt(transform.position.x / Chunk.
00115          chunkSize) * Chunk.chunkSize, Mathf.FloorToInt(transform.position.y / Chunk.chunkSize) * Chunk.chunkSize, Mathf.
00116          FloorToInt(transform.position.z / Chunk.chunkSize) * Chunk.chunkSize);
00117          for (int i = 0; i < nearbyChunks.Length; i++)
00118          {
00119              ChunkWorldPos chunkPos = new ChunkWorldPos(nearbyChunks[i].x * Chunk.chunkSize
00120 + playerPos.x, 0, nearbyChunks[i].z * Chunk.chunkSize + playerPos.z);
00121              for (int j = -1; j < 2; j++)
00122              {
00123                  Chunk nearbyChunk = world.GetChunk(chunkPos.x, j * Chunk.chunkSize,
00124                  chunkPos.z);
00125                  if (nearbyChunk != null)
00126                      nearbyChunk.applyCollisionMesh = true;
00127              }
00128          }
```

6.35.2.2 BuildChunk()

```
void BeeGame.Terrain.Chunks.LoadChunks.BuildChunk (
    ChunkWorldPos pos) [private]
```

Makes a chunk in the given position if it does not already exist

Parameters

<i>pos</i>	the position of the new chunk
------------	-------------------------------

Definition at line 186 of file [LoadChunks.cs](#).

```
00187      {
00188          if (world.GetChunk(pos.x, pos.y, pos.z) == null)
00189              world.CreateChunk(pos.x, pos.y, pos.z);
00190      }
```

6.35.2.3 DeleteChunks()

```
bool BeeGame.Terrain.Chunks.LoadChunks.DeleteChunks () [private]
```

Destroys [Chunks](#) every 10 calls

Returns

true if [Chunks](#) were destroyed

Definition at line 196 of file [LoadChunks.cs](#).

```
00197      {
00198          /* destroys every 10 call to reduce load on CPU so that chunks are not destroyed and created
at the same time
00199          if(timer == 10)
00200          {
00201              timer = 0;
00202              var chunksToDelete = new List<ChunkWorldPos>();
00203
00204              // *go through all of the built chunks and if the chunk is 256 units away it is assumed to
be out of sight so is added to the destroy list
00205              foreach (var chunk in world.chunks)
00206              {
00207                  float distance = Vector3.Distance(chunk.Value.transform.position, transform.position);
00208
00209                  if (distance > 256)
00210                      chunksToDelete.Add(chunk.Key);
00211              }
00212
00213              foreach (var chunk in chunksToDelete)
00214              {
00215                  world.DestroyChunk(chunk.x, chunk.y, chunk.z);
00216              }
00217
00218              return true;
00219          }
00220
00221          timer++;
00222
00223          return false;
00224      }
```

6.35.2.4 FindChunksToLoad()

```
void BeeGame.Terrain.Chunks.FindChunksToLoad ( ) [private]
```

Finds the [Chunks](#) that should be rendered

Definition at line 150 of file [LoadChunks.cs](#).

```

00151      {
00152          if (buildList.Count == 0)
00153          {
00154              /* gets the player position in chunk coordinates
00155              ChunkWorldPos playerPos = new ChunkWorldPos(Mathf.FloorToInt(transform.position.x / Chunk.
00156              chunkSize) * Chunk.chunkSize, Mathf.FloorToInt(transform.position.y / Chunk.chunkSize) * Chunk.chunkSize,
00157              Mathf.FloorToInt(transform.position.z / Chunk.chunkSize) * Chunk.chunkSize);
00158
00159              /* check all of the chunk positions and if that position does not have a chunk in it make
00160              it
00161              for (int i = 0; i < chunkPositions.Length; i++)
00162              {
00163                  ChunkWorldPos newChunkPos = new ChunkWorldPos(chunkPositions[i].x * Chunk
00164                  .chunkSize + playerPos.x, 0, chunkPositions[i].z * Chunk.chunkSize + playerPos.z);
00165
00166                  Chunk newChunk = world.GetChunk(newChunkPos.x, newChunkPos.y, newChunkPos.
00167                  z);
00168
00169                  if (newChunk != null && (newChunk.rendered || buildList.Contains(newChunkPos)))
00170                      continue;
00171
00172                  for (int y = -1; y < 2; y++)
00173                  {
00174                      for (int x = newChunkPos.x - Chunk.chunkSize; x < newChunkPos.x + Chunk.chunkSize;
00175                      x += Chunk.chunkSize)
00176                      {
00177                          for (int z = newChunkPos.z - Chunk.chunkSize; z < newChunkPos.z + Chunk.
00178                          chunkSize; z += Chunk.chunkSize)
00179                          {
00180                              buildList.Add(new ChunkWorldPos(x, y * Chunk.chunkSize, z));
00181
00182                          }
00183
00184                      }
00185
00186                  }
00187
00188              }
00189
00190          }
00191
00192      }
00193
00194 }
```

6.35.2.5 LoadAndRenderChunks()

```
void BeeGame.Terrain.Chunks.LoadAndRenderChunks ( ) [private]
```

Gets the chunks that could be built and renders them

Definition at line 133 of file [LoadChunks.cs](#).

```

00134      {
00135          /* if there is something in the build list new chunks can be made
00136          if (buildList.Count != 0)
00137          {
00138              /* makes all of the chunks in the build list. Works backwards through the list so that no
00139              chunk is missed because chunks are removed from the list as they are made
00140              for (int i = buildList.Count - 1, j = 0; i >= 0 && j < 8; i--, j++)
00141              {
00142                  BuildChunk(buildList[0]);
00143                  buildList.RemoveAt(0);
00144              }
00145          }
00146
00147      }
00148
00149 }
```

6.35.2.6 Start()

```
void BeeGame.Terrain.Chunks.LoadChunks.Start () [private]
```

Sets the world

Definition at line 82 of file [LoadChunks.cs](#).

```
00083     {
00084         LandGeneration.Terrain.world = world;
00085     }
```

6.35.2.7 Update()

```
void BeeGame.Terrain.Chunks.LoadChunks.Update () [private]
```

Builds, Renders, and Remmoves Chunks

Definition at line 90 of file [LoadChunks.cs](#).

```
00091     {
00092         if (DeleteChunks ())
00093             return;
00094         if (!world.chunkHasMadeCollisionMesh)
00095         {
00096             FindChunksToLoad ();
00097             LoadAndRenderChunks ();
00098             ApplyCollisionMeshToNearbyChunks ();
00099         }
00100        /* stops chunks being made and collision meshes being made at the same time
00101        world.chunkHasMadeCollisionMesh = false;
00102    }
```

6.35.3 Member Data Documentation

6.35.3.1 buildList

```
List<ChunkWorldPos> BeeGame.Terrain.Chunks.LoadChunks.buildList = new List<ChunkWorldPos>()
[private]
```

List if chunks to build

Definition at line 22 of file [LoadChunks.cs](#).

6.35.3.2 chunkPositions

```
ChunkWorldPos [ ] BeeGame.Terrain.Chunks.LoadChunks.chunkPositions [static], [private]
```

Positions to make chunks aroud the player ///

Definition at line 27 of file [LoadChunks.cs](#).

6.35.3.3 nearbyChunks

`ChunkWorldPos [] BeeGame.Terrain.Chunks.LoadChunks.nearbyChunks [static], [private]`

Initial value:

```
= new ChunkWorldPos[] { new ChunkWorldPos(0, 0, 0), new ChunkWorldPos(1, 0, 0), new ChunkWorldPos(-1, 0, 0),
    , new ChunkWorldPos(0, 0, 1), new ChunkWorldPos(0, 0, -1),
                                new ChunkWorldPos(1, 0, 1), new
    ChunkWorldPos(1, 0, -1), new ChunkWorldPos(-1, 0, 1), new ChunkWorldPos(-1, 0, -1)}
```

Chunks in a 3x3 radius around the player that should have a collision mesh

Definition at line 70 of file [LoadChunks.cs](#).

6.35.3.4 timer

`int BeeGame.Terrain.Chunks.LoadChunks.timer = 0 [static], [private]`

Timer for chunk removal

Definition at line 76 of file [LoadChunks.cs](#).

6.35.3.5 world

`World BeeGame.Terrain.Chunks.LoadChunks.world`

The world the player is in

Definition at line 17 of file [LoadChunks.cs](#).

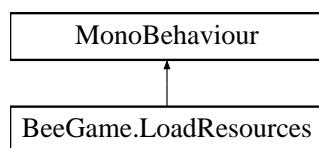
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/[LoadChunks.cs](#)

6.36 BeeGame.LoadResources Class Reference

Loads all of the resources in the game

Inheritance diagram for BeeGame.LoadResources:



Private Member Functions

- void [Awake \(\)](#)
Loads the sprites and prefab dictionarys
- void [Update \(\)](#)

Private Attributes

- int [delay](#) = 0

6.36.1 Detailed Description

Loads all of the resources in the game

Definition at line [9](#) of file [LoadResources.cs](#).

6.36.2 Member Function Documentation

6.36.2.1 Awake()

```
void BeeGame.LoadResources.Awake ( ) [private]
```

Loads the sprites and prefab dictionarys

Definition at line [14](#) of file [LoadResources.cs](#).

```
00015      {
00016          Serialization.Serialization.MakeDirectorys();
00017          Serialization.Serialization.LoadPlayerPosition(GameObject.Find("Player").GetComponent<Transform
00018 >());
00019          Serialization.Serialization.LoadQuests();
00020
00021          SpriteDictionary.LoadSprites();
00022          PrefabDictionary.LoadPrefabs();
00023      }
```

6.36.2.2 Update()

```
void BeeGame.LoadResources.Update ( ) [private]
```

Definition at line [27](#) of file [LoadResources.cs](#).

```
00028      {
00029          if (delay++ >= 1000)
00030              Serialization.Serialization.SaveQuests();
00031      }
```

6.36.3 Member Data Documentation

6.36.3.1 delay

```
int BeeGame.LoadResources.delay = 0 [private]
```

Definition at line 25 of file [LoadResources.cs](#).

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/[LoadResources.cs](#)

6.37 BeeGame.Terrain.Chunks.MeshData Class Reference

The data for a [Chunks](#)'s Mesh

Public Member Functions

- void [AddQuadTriangles](#) (bool addToRenderMesh=true)
Adds 2 triangles to the triangle list
- void [AddVertices](#) (THVector3 pos, bool addToRenderMesh=true, [Direction](#) direction=[Direction.DOWN](#))
Adds vertices to the render and collision Meshes
- void [AddTriangle](#) (int tri)
Adds a triangle to both the render and collision meshes

Public Attributes

- List< Vector3 > [verts](#) = new List<Vector3>()
Vertices for the [Chunk](#) render Mesh
- List< int > [tris](#) = new List<int>()
Triangles for the [Chunk](#) render Mesh
- List< Vector2 > [uv](#) = new List<Vector2>()
UV mapping for the [Chunk](#) render Mesh
- List< Vector3 > [colVerts](#) = new List<Vector3>()
Vertices for the [Chunk](#) collider Mesh
- List< int > [colTris](#) = new List<int>()
Triangles for the [Chunk](#) collider Mesh
- bool [shareMeshes](#) = true
Should this chunk share its collider and render Meshes
- bool [done](#) = false

6.37.1 Detailed Description

The data for a [Chunks](#)'s Mesh

Definition at line 11 of file [MeshData.cs](#).

6.37.2 Member Function Documentation

6.37.2.1 AddQuadTriangles()

```
void BeeGame.Terrain.Chunks.MeshData.AddQuadTriangles (
    bool addToRenderMesh = true )
```

Adds 2 triangles to the triangle list

Parameters

<code>addToRenderMesh</code>	Should the triangles be added to the render Mesh
------------------------------	--

Definition at line 46 of file [MeshData.cs](#).

```
00047      {
00048          /*adds the triangles in an anticlockwise order
00049
00050          if (addToRenderMesh)
00051          {
00052              tris.Add(verts.Count - 4);
00053              tris.Add(verts.Count - 3);
00054              tris.Add(verts.Count - 2);
00055              tris.Add(verts.Count - 4);
00056              tris.Add(verts.Count - 2);
00057              tris.Add(verts.Count - 1);
00058          }
00059
00060          colTris.Add(colVerts.Count - 4);
00061          colTris.Add(colVerts.Count - 3);
00062          colTris.Add(colVerts.Count - 2);
00063          colTris.Add(colVerts.Count - 4);
00064          colTris.Add(colVerts.Count - 2);
00065          colTris.Add(colVerts.Count - 1);
00066      }
```

6.37.2.2 AddTriangle()

```
void BeeGame.Terrain.Chunks.MeshData.AddTriangle (
    int tri )
```

Adds a triangle to both the render and collision meshes

Parameters

<code>tri</code>	triangle
------------------	----------

not used anymore remove?

Definition at line 91 of file [MeshData.cs](#).

```
00092      {
00093          tris.Add(tri);
00094
00095          colTris.Add(tri - (verts.Count - colVerts.Count));
00096      }
```

6.37.2.3 AddVertices()

```
void BeeGame.Terrain.Chunks.MeshData.AddVertices (
    THVector3 pos,
    bool addToRenderMesh = true,
    Direction direction = Direction.DOWN )
```

Adds vertices to the render and collision Meshes

Parameters

<i>pos</i>	Position of the vertice
<i>addToRenderMesh</i>	Should the vertice be added to the render Mesh
<i>direction</i>	What face is this vertice on

Definition at line 74 of file [MeshData.cs](#).

```

00075      {
00076          if (addToRenderMesh)
00077              verts.Add(pos);
00078
00079          /* if the vertice is on the top face make its positon slightly smaller
00080          if(direction == Direction.UP)
00081              colVerts.Add(pos - new THVector3(0.01f, 0, 0.01f));
00082      }

```

6.37.3 Member Data Documentation

6.37.3.1 colTris

```
List<int> BeeGame.Terrain.Chunks.MeshData.colTris = new List<int>()
```

Triangles for the [Chunk](#) collider Mesh

Definition at line 33 of file [MeshData.cs](#).

6.37.3.2 colVerts

```
List<Vector3> BeeGame.Terrain.Chunks.MeshData.colVerts = new List<Vector3>()
```

Vertices for the [Chunk](#) collider Mesh

Definition at line 29 of file [MeshData.cs](#).

6.37.3.3 done

```
bool BeeGame.Terrain.Chunks.MeshData.done = false
```

Definition at line 40 of file [MeshData.cs](#).

6.37.3.4 shareMeshes

```
bool BeeGame.Terrain.Chunks.MeshData.shareMeshes = true
```

Should this chunk share is collider and render Meshes

Definition at line 38 of file [MeshData.cs](#).

6.37.3.5 tris

`List<int> BeeGame.Terrain.Chunks.MeshData.tris = new List<int>()`

Triangles for the [Chunk](#) render Mesh

Definition at line [20](#) of file [MeshData.cs](#).

6.37.3.6 uv

`List<Vector2> BeeGame.Terrain.Chunks.MeshData.uv = new List<Vector2>()`

UV mapping for the [Chunk](#) render Mesh

Definition at line [24](#) of file [MeshData.cs](#).

6.37.3.7 verts

`List<Vector3> BeeGame.Terrain.Chunks.MeshData.verts = new List<Vector3>()`

Verticies for the [Chunk](#) render Mesh

Definition at line [16](#) of file [MeshData.cs](#).

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/[MeshData.cs](#)

6.38 BeeGame.Items.NormalBee Class Reference

Public Member Functions

- `override int GetHashCode ()`

Public Attributes

- `BeeSpecies pSpecies`
Primary BeeSpecies of the Bee
- `BeeLifeSpan pLifespan`
Primary BeeLifeSpan of the Bee
- `uint pFertility`
Primary Fertility of the Bee
- `BeeEffect pEffect`
Primary BeeEffect of the Bee
- `BeeProductionSpeed pProdSpeed`
Primary BeeProductionSpeed of the Bee
- `BeeSpecies sSpecies`
Secondary BeeGame.Enums.BeeSpecies of the Bee
- `BeeLifeSpan sLifespan`
Secondary BeeGame.Enums.BeeLifeSpan of the Bee
- `uint sFertility`
Secondary Fertility of the Bee
- `BeeEffect sEffect`
Secondary BeeGame.Enums.BeeEffect of the Bee
- `BeeProductionSpeed sProdSpeed`
Secondary BeeGame.Enums.BeeProductionSpeed of the Bee

6.38.1 Detailed Description

Definition at line 253 of file [Bee.cs](#).

6.38.2 Member Function Documentation

6.38.2.1 GetHashCode()

```
override int BeeGame.Items.NormalBee.GetHashCode ( )
```

Definition at line 305 of file [Bee.cs](#).

```
00306      {
00307          unchecked
00308          {
00309              //int hashcode = 13;
00310
00311              var temp = $""
00312              {(int)pSpecies}{(int)sSpecies}{(int)pLifespan}{(int)sLifespan}{(int)pFertility}{(int)sFertility}{(int)pEffect}{(int)sEffect}
00313              var hashcode = (int)(Int64.Parse(temp) ^ (127 * 13) / 159);
00314
00315              //hashcode += ((int)pSpecies ^ (int)pLifespan ^ (int)pFertility ^ (int)pEffect ^
00316              (int)pProdSpeed) * 127;
00317              //hashcode += ((int)sSpecies ^ (int)sLifespan ^ (int)sFertility ^ (int)sEffect ^
00318              (int)sProdSpeed) * 307;
00319              return hashcode;
00320          }
00320 }
```

6.38.3 Member Data Documentation

6.38.3.1 pEffect

[BeeEffect](#) BeeGame.Items.NormalBee.pEffect

Primary BeeEffect of the Bee

Definition at line 273 of file [Bee.cs](#).

6.38.3.2 pFertility

```
uint BeeGame.Items.NormalBee.pFertility
```

Primary Fertility of the Bee

Definition at line 269 of file [Bee.cs](#).

6.38.3.3 pLifespan

`BeeLifeSpan` `BeeGame.Items.NormalBee.pLifespan`

Primary BeeLifeSpan of the [Bee](#)

Definition at line [265](#) of file [Bee.cs](#).

6.38.3.4 pProdSpeed

`BeeProductionSpeed` `BeeGame.Items.NormalBee.pProdSpeed`

Primary BeeProductionSpeed of the [Bee](#)

Definition at line [277](#) of file [Bee.cs](#).

6.38.3.5 pSpecies

`BeeSpecies` `BeeGame.Items.NormalBee.pSpecies`

Primary BeeSpecies of the [Bee](#)

Definition at line [261](#) of file [Bee.cs](#).

6.38.3.6 sEffect

`BeeEffect` `BeeGame.Items.NormalBee.sEffect`

Secondary BeeGame.Enums.BeeEffect of the [Bee](#)

Definition at line [298](#) of file [Bee.cs](#).

6.38.3.7 sFertility

`uint` `BeeGame.Items.NormalBee.sFertility`

Secondary Fertility of the [Bee](#)

Definition at line [294](#) of file [Bee.cs](#).

6.38.3.8 sLifespan

`BeeLifeSpan` `BeeGame.Items.NormalBee.sLifespan`

Secondary `BeeGame.Enums.BeeLifeSpan` of the `Bee`

Definition at line 290 of file `Bee.cs`.

6.38.3.9 sProdSpeed

`BeeProductionSpeed` `BeeGame.Items.NormalBee.sProdSpeed`

Secondary `BeeGame.Enums.BeeProductionSpeed` of the `Bee`

Definition at line 302 of file `Bee.cs`.

6.38.3.10 sSpecies

`BeeSpecies` `BeeGame.Items.NormalBee.sSpecies`

Secondary `BeeGame.Enums.BeeSpecies` of the `Bee`

Definition at line 286 of file `Bee.cs`.

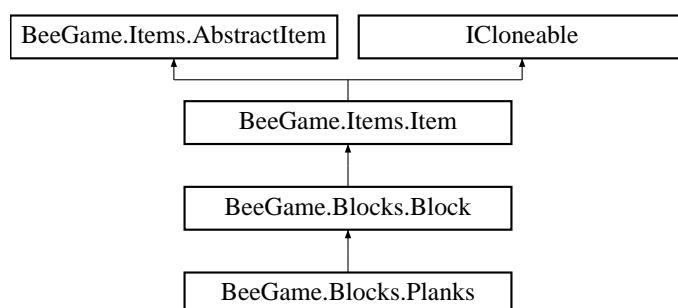
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/`Bee.cs`

6.39 BeeGame.Blocks.Planks Class Reference

Planks Block

Inheritance diagram for `BeeGame.Blocks.Planks`:



Public Member Functions

- [Planks \(\)](#)
Constructor
- [override Sprite GetItemSprite \(\)](#)
Returns the sprite for the item
- [override Tile TexturePosition \(Direction direction\)](#)
Position of the planks texture in the atlas
- [override int GetHashCode \(\)](#)
Base ID of the block
- [override string ToString \(\)](#)
Returns the name and ID of the block as a string

Static Public Attributes

- [static new int ID => 7](#)

Additional Inherited Members

6.39.1 Detailed Description

[Planks Block](#)

Definition at line [13](#) of file [Planks.cs](#).

6.39.2 Constructor & Destructor Documentation

6.39.2.1 [Planks\(\)](#)

`BeeGame.Blocks.Planks.Planks ()`

Constructor

Definition at line [21](#) of file [Planks.cs](#).

```
00021 : base("Planks") {}
```

6.39.3 Member Function Documentation

6.39.3.1 GetHashCode()

```
override int BeeGame.Blocks.Planks.GetHashCode ( ) [virtual]
```

Base ID of the block

Returns

5

Reimplemented from [BeeGame.Blocks.Block](#).

Definition at line 48 of file [Planks.cs](#).

```
00049     {  
00050         return ID;  
00051     }
```

6.39.3.2 GetItemSprite()

```
override Sprite BeeGame.Blocks.Planks.GetItemSprite ( ) [virtual]
```

Returns the sprite for the item

Returns

Sprite for this item

Reimplemented from [BeeGame.Blocks.Block](#).

Definition at line 25 of file [Planks.cs](#).

```
00026     {  
00027         return SpriteDictionary.GetSprite("Planks");  
00028     }
```

6.39.3.3 TexturePosition()

```
override Tile BeeGame.Blocks.Planks.TexturePosition (  
    Direction direction ) [virtual]
```

Position of the planks texture in the atlas

Parameters

<i>direction</i>	<input type="text"/>
------------------	----------------------

Returns

Reimplemented from [BeeGame.Items.Item](#).

Definition at line 37 of file [Planks.cs](#).

```
00038      {
00039          return new Tile { x = 2, y = 9 };
00040      }
```

6.39.3.4 ToString()

```
override string BeeGame.Blocks.Planks.ToString ( )
```

Returns the name and ID of the block as a string

Returns

A nicely formatted string

Definition at line 57 of file [Planks.cs](#).

```
00058      {
00059          return $"{itemName} \nID: {GetItemID()}";
00060      }
```

6.39.4 Member Data Documentation**6.39.4.1 ID**

```
new int BeeGame.Blocks.Planks.ID => 7 [static]
```

Definition at line 15 of file [Planks.cs](#).

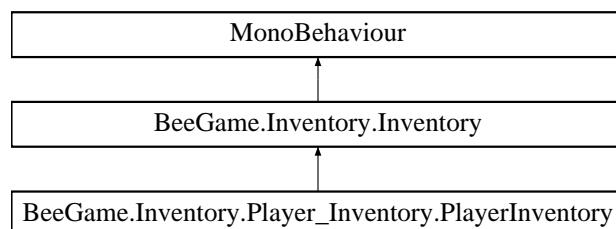
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/[Planks.cs](#)

6.40 BeeGame.Inventory.Player_Inventory.PlayerInventory Class Reference

Controls the player inventory

Inheritance diagram for BeeGame.Inventory.Player_Inventory.PlayerInventory:



Public Member Functions

- virtual void [SetPlayerInventory](#) (int size=36)
Set the size of the player inventory
- void [SelectedSlot](#) (int index)
Updates the currently selected hotbar slot
- bool [GetItemFromHotBar](#) (int slotIndex, out [Item](#) outItem)
Gets an item from the hotbar (9 [InventorySlots](#) at the bottom of the screen)
- void [RemoveItemFromInventory](#) (int index)
Removes 1 item from the given inventory index

Public Attributes

- GameObject [playerInventory](#)
Object that the inventory is

Protected Member Functions

- void [Awake](#) ()
Sets all required params for the inventory and loads ant saved versions of it
- void [Update](#) ()
Gives the inventory update ticks

Package Functions

- void [OpenPlayerInventory](#) ()
Show/Hide the player inventory

Private Member Functions

- void [PickupItem](#) ([ItemGameObject](#) item)
Pickup an item and put it into the [Inventory](#)

Additional Inherited Members

6.40.1 Detailed Description

Controls the player inventory

Definition at line 11 of file [PlayerInventory.cs](#).

6.40.2 Member Function Documentation

6.40.2.1 Awake()

```
void BeeGame.Inventory.Player_Inventory.PlayerInventory.Awake ( ) [protected]
```

Sets all required params for the inventory and loads ant saved versions of it

Definition at line 24 of file [PlayerInventory.cs](#).

```
00025      {
00026          SetPlayerInventory();
00027          inventoryName = "PlayerInventory";
00028          Serialization.SerializeInventory(this, inventoryName);
00029      }
```

6.40.2.2 GetItemFromHotBar()

```
bool BeeGame.Inventory.Player_Inventory.PlayerInventory.GetItemFromHotBar (
    int slotIndex,
    out Item outItem)
```

Gets an item from the hotbar (9 [InventorySlots](#) at the bottom of the screen)

Parameters

<i>slotIndex</i>	Index to get Item from
<i>outItem</i>	Item in the slot

Returns

true if *outItem* is placeable, false if *outItem* is null or not placeable

Definition at line 98 of file [PlayerInventory.cs](#).

```
00099      {
00100          /* get the item
00101          outItem = GetAllItems().itemsInInventory[slotIndex];
00102
00103          if (outItem == null)
00104              return false;
00105
00106          /* if the item is placeable and is not null remove 1 from the inventory as it is assumed it is
00107          //about to be placed in the world
00108          if(outItem.placeable)
00109              RemoveItemFromInventory(slotIndex);
00110
00111          return outItem.placeable;
00111      }
```

6.40.2.3 OpenPlayerInventory()

```
void BeeGame.Inventory.Player_Inventory.PlayerInventory.OpenPlayerInventory ( ) [package]
```

Show/Hide the player inventory

Definition at line 118 of file [PlayerInventory.cs](#).

```

00119      {
00120          if (floatingItem != null)
00121              return;
00122          thisInventoryOpen = !thisInventoryOpen;
00123          playerInventory.SetActive(!playerInventory.activeInHierarchy);
00124          THInput.isAnotherInventoryOpen = !
00125              THInput.isAnotherInventoryOpen;
00126          /* hides/shows the mouse depending on if te inventory is open or not
00127          if (playerInventory.activeInHierarchy)
00128          {
00129              Cursor.lockState = CursorLockMode.None;
00130              Cursor.visible = true;
00131          }
00132          else
00133          {
00134              Cursor.visible = false;
00135              Cursor.lockState = CursorLockMode.Locked;
00136          }
00137      }

```

6.40.2.4 PickupItem()

```
void BeeGame.Inventory.Player_Inventory.PlayerInventory.PickupItem (
    ItemGameObject item) [private]
```

Pickup an item and put it into the [Inventory](#)

Parameters

<i>item</i>	Item to try to put into the inventory
-------------	---------------------------------------

Definition at line 162 of file [PlayerInventory.cs](#).

```

00163      {
00164          item.item.itemStackCount = 1;
00165
00166          /* if the item can be added to the inventory do that
00167          if (AddItemToInventory(item.item))
00168          {
00169              QuestEvents.CallItemPickupEvent(item.
00170                  item.GetHashCode());
00171              /* if the item was added destroy its gameobject and save the inventory
00172              Destroy(item.gameObject);
00173              Serialization.Serialization.SerializeInventory(this,
00174                  inventoryName);
00175          }
00176      }

```

6.40.2.5 RemoveItemFromInventory()

```
void BeeGame.Inventory.Player_Inventory.PlayerInventory.RemoveItemFromInventory (
    int index)
```

Removes 1 item from the given inventory index

Parameters

<i>index</i>	
--------------	--

Definition at line 143 of file [PlayerInventory.cs](#).

```

00144      {
00145          /* if the item is already null nothing needs to be removed
00146          if (GetAllItems().itemsInInventory[index] != null)
00147          {
00148              /* remove 1 item and if that was the last in the stack remove the item from the inventory
00149              GetAllItems().itemsInInventory[index].
00150              itemStackCount -= 1;
00151              if (GetAllItems().itemsInInventory[index].itemStackCount <= 0)
00152                  GetAllItems().itemsInInventory[index] = null;
00153
00154              Serialization.Serialization.SerializeInventory(this,
00155              inventoryName);
00156          }

```

6.40.2.6 SelectedSlot()

```
void BeeGame.Inventory.Player_Inventory.PlayerInventory.SelectedSlot (
    int index )
```

Updates the currently selected hotbar slot

Parameters

<i>index</i>	Slot that is selected
--------------	-----------------------

Definition at line 82 of file [PlayerInventory.cs](#).

```

00083      {
00084          for (int i = 0; i < slots.Length; i++)
00085          {
00086              slots[i].selectedSlot = false;
00087          }
00088          slots[index].selectedSlot = true;
00090      }

```

6.40.2.7 SetPlayerInventory()

```
virtual void BeeGame.Inventory.Player_Inventory.PlayerInventory.SetPlayerInventory (
    int size = 36 ) [virtual]
```

Set the size of the player inventory

Definition at line 34 of file [PlayerInventory.cs](#).

```

00035      {
00036          if (!InventorySet())
00037              SetInventorySize(size);
00038      }

```

6.40.2.8 Update()

```
void BeeGame.Inventory.Player_Inventory.PlayerInventory.Update ( ) [protected]
```

Gives the inventory update ticks

Definition at line 44 of file [PlayerInventory.cs](#).

```
00045     {
00046         UpdateBase ();
00047
00048         /* checks if the inventory should be opened/closed
00049         if ((thisInventoryOpen || !playerInventory.activeInHierarchy)
00050             && !THInput.chestOpen && THInput.GetButtonDown("Player Inventory"))
00051         {
00052             if (THInput.blockInventoryJustClosed)
00053             {
00054                 THInput.blockInventoryJustClosed = false;
00055                 return;
00056             }
00057             {
00058                 OpenPlayerInventory ();
00059             }
00060         }
00061
00062         /* dont pickup items if the inventory is open
00063         if (THInput.isAnotherInventoryOpen)
00064             return;
00065
00066         /* checks if something should be picked up and put into the inventory
00067         RaycastHit[] hit = Physics.SphereCastAll(transform.position, 1f, transform.forward);
00068
00069         for (int i = hit.Length - 1; i >= 0; i--)
00070         {
00071             if (hit[i].collider.GetComponent<ItemGameObject>())
00072                 PickupItem(hit[i].collider.GetComponent<
00073                     ItemGameObject>());
00074         }
00075     }
```

6.40.3 Member Data Documentation

6.40.3.1 playerInventory

```
GameObject BeeGame.Inventory.Player_Inventory.PlayerInventory.playerInventory
```

Object that the inventory is

Definition at line 17 of file [PlayerInventory.cs](#).

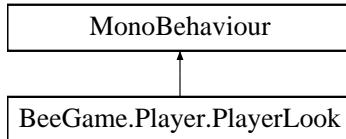
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/PlayerInventory/[PlayerInventory.cs](#)

6.41 BeeGame.Player.PlayerLook Class Reference

The look for the player

Inheritance diagram for BeeGame.Player.PlayerLook:



Public Attributes

- Transform `myTransform`
Player transform
- Transform `cameraTransform`
Camera transform
- float `rotationLock`
Lock for camera X rotation
- float `speed` = 5
Look move speed

Private Member Functions

- void `Start ()`
Locks teh cursor and hides it
- void `Update ()`
Every fixed update check if the look shoud be moved
- void `Look ()`
Moves the look rotation

Private Attributes

- float `yRot` = 0
Current Y rotation
- float `xRot` = 0
Current X rotation

6.41.1 Detailed Description

The look for the player

Definition at line 9 of file [PlayerLook.cs](#).

6.41.2 Member Function Documentation

6.41.2.1 Look()

```
void BeeGame.Player.PlayerLook.Look ( ) [private]
```

Moves the look rotation

Definition at line 66 of file [PlayerLook.cs](#).

```
00067      {
00068          //Only X/Y rotation needed as Z rotation would be wierd
00069          yRot += Input.GetAxis("Mouse X") * speed * Time.timeScale;
00070          xRot -= Input.GetAxis("Mouse Y") * speed * Time.timeScale;
00071
00072          //clamps the X rotation so the player camera cannot do flips
00073          xRot = Mathf.Clamp(xRot, -rotationLock,
00074                               rotationLock);
00074
00075          myTransform.rotation = Quaternion.Euler(0, yRot, 0);
00076          cameraTransform.localRotation = Quaternion.Euler(xRot, 0, 0);
00077      }
```

6.41.2.2 Start()

```
void BeeGame.Player.PlayerLook.Start ( ) [private]
```

Locks teh cursor and hides it

Definition at line 43 of file [PlayerLook.cs](#).

```
00044      {
00045          Cursor.lockState = CursorLockMode.Locked;
00046          Cursor.visible = false;
00047      }
```

6.41.2.3 Update()

```
void BeeGame.Player.PlayerLook.Update ( ) [private]
```

Every fixed update check if the look shoud be moved

Definition at line 52 of file [PlayerLook.cs](#).

```
00053      {
00054          /*the look wil not update when a inventory GUI is open
00055          if (!THInput.isAnotherInventoryOpen)
00056          {
00057              Look();
00058          }
00059      }
```

6.41.3 Member Data Documentation

6.41.3.1 cameraTransform

```
Transform BeeGame.Player.PlayerLook.cameraTransform
```

Camera transform

Definition at line 19 of file [PlayerLook.cs](#).

6.41.3.2 myTransform

```
Transform BeeGame.Player.PlayerLook.myTransform
```

Player transform

Definition at line 15 of file [PlayerLook.cs](#).

6.41.3.3 rotationLock

```
float BeeGame.Player.PlayerLook.rotationLock
```

Lock for camera X rotation

Definition at line 24 of file [PlayerLook.cs](#).

6.41.3.4 speed

```
float BeeGame.Player.PlayerLook.speed = 5
```

Look move speed

Definition at line 28 of file [PlayerLook.cs](#).

6.41.3.5 xRot

```
float BeeGame.Player.PlayerLook.xRot = 0 [private]
```

Current X rotation

Definition at line 36 of file [PlayerLook.cs](#).

6.41.3.6 yRot

```
float BeeGame.Player.PlayerLook.yRot = 0 [private]
```

Current Y rotation

Definition at line 32 of file [PlayerLook.cs](#).

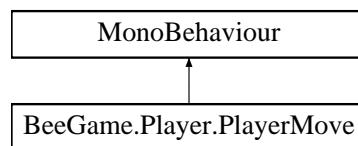
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/[PlayerLook.cs](#)

6.42 BeeGame.Player.PlayerMove Class Reference

Moves the player

Inheritance diagram for BeeGame.Player.PlayerMove:



Public Attributes

- float **speed** = 10f
Speed of the player
- float **gravity** = 9.81f
Gravity of the player
- float **maxVelocity** = 10f
Max velocity of the player
- float **jumpHeight** = 2f
How high can the player jump

Private Member Functions

- void **Awake** ()
Gets the rigidbody and sets its variables
- void **FixedUpdate** ()
Updates the player move
- void **OnCollisionStay** (Collision collision)
Sets that the player can jump when it hits the ground
- void **MovePlayer** ()
Moves the player
- float **VerticalJumpSpeed** ()
Vertical Jump speed of the character

Private Attributes

- bool `canJump` = false
Can the player jump?
- Rigidbody `myRigidBody`
Rigidbody for the player

6.42.1 Detailed Description

Moves the player

Definition at line 14 of file [PlayerMove.cs](#).

6.42.2 Member Function Documentation

6.42.2.1 Awake()

```
void BeeGame.Player.PlayerMove.Awake ( ) [private]
```

Gets the rigidbody and sets its variables

Definition at line 49 of file [PlayerMove.cs](#).

```
00050      {
00051          myRigidBody = GetComponent< Rigidbody > ();
00052
00053          //i want to use myown gravity and rotation
00054          myRigidBody.useGravity = false;
00055          myRigidBody.freezeRotation = true;
00056      }
```

6.42.2.2 FixedUpdate()

```
void BeeGame.Player.PlayerMove.FixedUpdate ( ) [private]
```

Updates the player move

Definition at line 61 of file [PlayerMove.cs](#).

```
00062      {
00063          //If the player is grounded it can move
00064          if (canJump)
00065          {
00066              MovePlayer();
00067          }
00068
00069          //adds the downward force
00070          myRigidBody.AddForce(new Vector3(0, myRigidBody.mass * -
00071              gravity, 0));
00071      }
```

6.42.2.3 MovePlayer()

```
void BeeGame.Player.PlayerMove.MovePlayer ( ) [private]
```

Moves the player

Definition at line 87 of file [PlayerMove.cs](#).

```
00088      {
00089          //Calculate the speed we want to achieve
00090          Vector3 targetVelocity = new Vector3(THInput.GetAxis("Horizontal"), 0,
00091                                              THInput.GetAxis("Vertical"));
00091          targetVelocity = transform.TransformDirection(targetVelocity);
00092          targetVelocity *= speed;
00093
00094          //Apply a force to reach the target speed
00095          Vector3 velocity = myRigidBody.velocity;
00096          Vector3 velocityChange = (targetVelocity - velocity);
00097
00098          //Clamping the velocity so that the player does not infinitely accelerate
00099          velocityChange.x = Mathf.Clamp(velocityChange.x, -maxVelocity,
00100                                              maxVelocity);
00100          velocityChange.z = Mathf.Clamp(velocityChange.z, -maxVelocity,
00101                                              maxVelocity);
00101          velocityChange.y = 0;
00102
00103          //Adds the force to the player so they move in the correct direction
00104          myRigidBody.AddForce(velocityChange, ForceMode.Impulse);
00105
00106          //Jumping
00107          if (canJump && THInput.GetButton("Jump"))
00108          {
00109              canJump = false;
00110              myRigidBody.velocity = new Vector3(velocity.x,
00111                                              VerticalJumpSpeed(), velocity.z);
00112          }
00112 }
```

6.42.2.4 OnCollisionStay()

```
void BeeGame.Player.PlayerMove.OnCollisionStay (
    Collision collision ) [private]
```

Sets that the player can jump when it hits the ground

Parameters

<i>collision</i>	What the player hit
------------------	---------------------

Definition at line 77 of file [PlayerMove.cs](#).

```
00078      {
00079          canJump = true;
00080      }
```

6.42.2.5 VerticalJumpSpeed()

```
float BeeGame.Player.PlayerMove.VerticalJumpSpeed ( ) [private]
```

Vertical Jump speed of the character

Returns

Speed of the jump

Definition at line 118 of file [PlayerMove.cs](#).

```
00119      {
00120          /*Gets the correct of fore required for the player to reach the desired apex
00121          /*Can this be done without Square Root as that take alot of work?
00122          return Mathf.Sqrt(2 * jumpHeight * gravity);
00123      }
```

6.42.3 Member Data Documentation

6.42.3.1 canJump

```
bool BeeGame.Player.PlayerMove.canJump = false [private]
```

Can the player jump?

Definition at line 33 of file [PlayerMove.cs](#).

6.42.3.2 gravity

```
float BeeGame.Player.PlayerMove.gravity = 9.81f
```

Gravity of the player

Definition at line 24 of file [PlayerMove.cs](#).

6.42.3.3 jumpHeight

```
float BeeGame.Player.PlayerMove.jumpHeight = 2f
```

How high can the player jump

Definition at line 37 of file [PlayerMove.cs](#).

6.42.3.4 maxVelocity

```
float BeeGame.Player.PlayerMove.maxVelocity = 10f
```

Max velocity of the player

Definition at line 28 of file [PlayerMove.cs](#).

6.42.3.5 myRigidBody

```
Rigidbody BeeGame.Player.PlayerMove.myRigidBody [private]
```

Rigidbody for the player

Definition at line 42 of file [PlayerMove.cs](#).

6.42.3.6 speed

```
float BeeGame.Player.PlayerMove.speed = 10f
```

Speed of the player

Definition at line 20 of file [PlayerMove.cs](#).

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/[PlayerMove.cs](#)

6.43 BeeGame.Core.Dictionaries.PrefabDictionary Class Reference

The prefabs available to the game

Static Public Member Functions

- static void [LoadPrefabs](#) ()
Loads the prefabs into the Dictionary
- static GameObject [GetPrefab](#) (string prefab)
Returns a GameObject in the prefab dictionary

Static Private Attributes

- static Dictionary< string, GameObject > [prefabDictionary](#) = new Dictionary<string, GameObject>()
All of the prefabs available to spawn in

6.43.1 Detailed Description

The prefabs available to the game

Definition at line 9 of file [PrefabDictionary.cs](#).

6.43.2 Member Function Documentation

6.43.2.1 GetPrefab()

```
static GameObject BeeGame.Core.Dictionaries.PrefabDictionary.GetPrefab (
    string prefab ) [static]
```

Returns a GameObject in the prefab dictionary

Parameters

<i>prefab</i>	Name of th prefab to get
---------------	--------------------------

Returns

Prefab of the given name

Definition at line 29 of file [PrefabDictionary.cs](#).

```
00030      {
00031          return prefabDictionary[prefab];
00032      }
```

6.43.2.2 LoadPrefabs()

```
static void BeeGame.Core.Dictionaries.PrefabDictionary.LoadPrefabs ( ) [static]
```

Loads the prefabs into the Dictionary

Definition at line 19 of file [PrefabDictionary.cs](#).

```
00020      {
00021          prefabDictionary = Resources.Resources.GetPrefabs();
00022      }
```

6.43.3 Member Data Documentation**6.43.3.1 prefabDictionary**

```
Dictionary<string, GameObject> BeeGame.Core.Dictionaries.PrefabDictionary.prefabDictionary =
new Dictionary<string, GameObject>() [static], [private]
```

All of the prefabs available to spawn in

Definition at line 14 of file [PrefabDictionary.cs](#).

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionaries/[PrefabDictionary.cs](#)

6.44 BeeGame.Items.QueenBee Class Reference**Public Member Functions**

- [QueenBee \(\)](#)
- [QueenBee \(NormalBee princess, NormalBee drone\)](#)
- override int [GetHashCode \(\)](#)

Properties

- **NormalBee queen** [get, set]
Original princess traits
- **NormalBee drone** [get, set]
Paired drone traits

6.44.1 Detailed Description

Definition at line 223 of file [Bee.cs](#).

6.44.2 Constructor & Destructor Documentation

6.44.2.1 QueenBee() [1/2]

BeeGame.Items.QueenBee.QueenBee ()

Definition at line 235 of file [Bee.cs](#).

00235 { }

6.44.2.2 QueenBee() [2/2]

BeeGame.Items.QueenBee.QueenBee (
 NormalBee princess,
 NormalBee drone)

Definition at line 237 of file [Bee.cs](#).

00238 {
00239 this.queen = princess;
00240 this.drone = drone;
00241 }

6.44.3 Member Function Documentation

6.44.3.1 GetHashCode()

override int BeeGame.Items.QueenBee.GetHashCode ()

Definition at line 243 of file [Bee.cs](#).

00244 {
00245 unchecked
00246 {
00247 return (int)Int64.Parse(\$"{queen.GetHashCode() ^ drone.GetHashCode()}");
00248 }
00249 }

6.44.4 Property Documentation

6.44.4.1 drone

`NormalBee BeeGame.Items.QueenBee.drone [get], [set]`

Paired drone traits

Definition at line 233 of file [Bee.cs](#).

6.44.4.2 queen

`NormalBee BeeGame.Items.QueenBee.queen [get], [set]`

Original princess traits

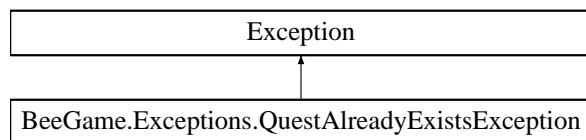
Definition at line 229 of file [Bee.cs](#).

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/[Bee.cs](#)

6.45 BeeGame.Exceptions.QuestAlreadyExistsException Class Reference

Inheritance diagram for BeeGame.Exceptions.QuestAlreadyExistsException:



Public Member Functions

- [QuestAlreadyExistsException \(\)](#)
- [QuestAlreadyExistsException \(string message\)](#)
- [QuestAlreadyExistsException \(string message, Exception innerException\)](#)

6.45.1 Detailed Description

Definition at line 5 of file [QuestAlreadyExistsException.cs](#).

6.45.2 Constructor & Destructor Documentation

6.45.2.1 QuestAlreadyExistsException() [1/3]

```
BeeGame.Exceptions.QuestAlreadyExistsException.QuestAlreadyExistsException ( )
```

Definition at line 7 of file [QuestAlreadyExistsException.cs](#).

```
00007 : base()
00008 {
00009 }
00010 }
```

6.45.2.2 QuestAlreadyExistsException() [2/3]

```
BeeGame.Exceptions.QuestAlreadyExistsException.QuestAlreadyExistsException (
    string message )
```

Definition at line 12 of file [QuestAlreadyExistsException.cs](#).

```
00012 : base(message)
00013 {
00014 }
00015 }
```

6.45.2.3 QuestAlreadyExistsException() [3/3]

```
BeeGame.Exceptions.QuestAlreadyExistsException.QuestAlreadyExistsException (
    string message,
    Exception innerException )
```

Definition at line 17 of file [QuestAlreadyExistsException.cs](#).

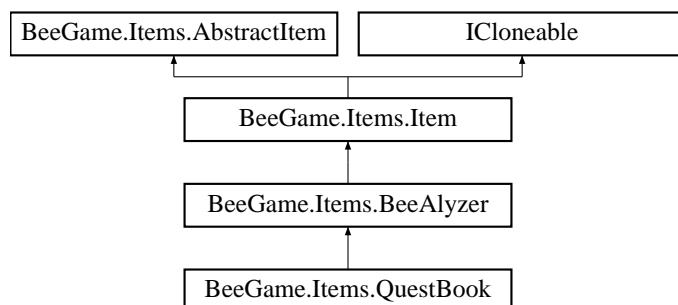
```
00017 : base(message,
00018     innerException)
00019 {
00020 }
```

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Exceptions/[QuestAlreadyExistsException.cs](#)

6.46 BeeGame.Items.QuestBook Class Reference

Inheritance diagram for BeeGame.Items.QuestBook:



Public Member Functions

- [QuestBook \(\)](#)
- [override void OpenItemInventory \(Inventory.Inventory playerInventory=null\)](#)
opens and closes the items inventory
- [override Sprite GetItemSprite \(\)](#)
Returns the items Sprite
- [override int GetHashCode \(\)](#)

Static Public Attributes

- [static new int ID = 15](#)

Properties

- [override int maxStackCount \[get\]](#)
- [override bool placeable \[get\]](#)

Additional Inherited Members

6.46.1 Detailed Description

Definition at line 13 of file [QuestBook.cs](#).

6.46.2 Constructor & Destructor Documentation

6.46.2.1 QuestBook()

BeeGame.Items.QuestBook.QuestBook ()

Definition at line 31 of file [QuestBook.cs](#).

```
00031 : base("Quest Book")
00032 {
00033 }
00034 }
```

6.46.3 Member Function Documentation

6.46.3.1 GetHashCode()

```
override int BeeGame.Items.QuestBook.GetHashCode ( ) [virtual]
```

Reimplemented from [BeeGame.Items.BeeAlyzer](#).

Definition at line 58 of file [QuestBook.cs](#).

```
00059      {
00060          return ID;
00061      }
```

6.46.3.2 GetItemSprite()

```
override Sprite BeeGame.Items.QuestBook.GetItemSprite ( ) [virtual]
```

Returns the items Sprite

Returns

Reimplemented from [BeeGame.Items.BeeAlyzer](#).

Definition at line 53 of file [QuestBook.cs](#).

```
00054      {
00055          return SpriteDictionary.GetSprite("QuestBook");
00056      }
```

6.46.3.3 OpenItemInventory()

```
override void BeeGame.Items.QuestBook.OpenItemInventory (
    Inventory.Inventory playerInventory = null ) [virtual]
```

opens and closes the items inventory

Parameters

<code>playerInventory</code>	Used when opening inventory to give the new inventory the players inventory(Inventory.Player_Inventory.PlayerInventory)
------------------------------	---

Reimplemented from [BeeGame.Items.BeeAlyzer](#).

Definition at line 36 of file [QuestBook.cs](#).

```
00037      {
00038          if (myInventory == null)
```

```

00039      {
00040          /* makes the inventory
00041      myInventory = (GameObject)UnityEngine.Object.Instantiate(
00042          UnityEngine.Resources.Load("Prefabs/QuestBookInventory"));
00043          /* opens the inventory and gives it the players inventory
00044      myInventory.GetComponent<QuestBookInventory>().ToggleInventory
00045      (playerInventory);
00046      myInventory.GetComponent<QuestBookInventory>().myItem = this;
00047  }
00048  {
00049      myInventory = null;
00050  }
00051 }
```

6.46.4 Member Data Documentation

6.46.4.1 ID

`new int BeeGame.Items.QuestBook.ID = 15 [static]`

Definition at line 15 of file [QuestBook.cs](#).

6.46.5 Property Documentation

6.46.5.1 maxStackCount

`override int BeeGame.Items.QuestBook.maxStackCount [get]`

Definition at line 17 of file [QuestBook.cs](#).

6.46.5.2 placeable

`override bool BeeGame.Items.QuestBook.placeable [get]`

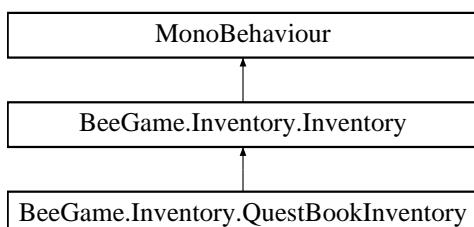
Definition at line 24 of file [QuestBook.cs](#).

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/[QuestBook.cs](#)

6.47 BeeGame.Inventory.QuestBookInventory Class Reference

Inheritance diagram for BeeGame.Inventory.QuestBookInventory:



Public Member Functions

- void [AddQuests \(\)](#)
- override void [ToggleInventory \(Inventory inv\)](#)

Opens and closes this inventory

Public Attributes

- Button [questButton](#)
- GameObject [scrollObject](#)

Protected Member Functions

- void [Start \(\)](#)
- void [Update \(\)](#)

Package Attributes

- [QuestBook myItem](#)

Private Member Functions

- void [QuestAchieved \(string key\)](#)

Additional Inherited Members

6.47.1 Detailed Description

Definition at line 14 of file [QuestBookInventory.cs](#).

6.47.2 Member Function Documentation

6.47.2.1 AddQuests()

```
void BeeGame.Inventory.QuestBookInventory.AddQuests ( )
```

Definition at line 31 of file [QuestBookInventory.cs](#).

```

00032      {
00033          for (int i = 0; i < scrollObject.transform.childCount; i++)
00034          {
00035              Destroy(scrollObject.transform.GetChild(i).gameObject);
00036          }
00037
00038          var compleatedQuests = Quests.ReturnCompleatedQuests();
00039          var currentQuests = Quests.ReturnCurrentQuests();
00040          var compleatClaimedQuests = Quests.
00041          ReturnCompleatedClaimedQuests();
00042
00043          for (int i = 0; i < compleatedQuests.Count; i++)
00044          {
00045              var go = Instantiate(questButton);
00046
00047              go.transform.SetParent(scrollObject.transform, false);
00048
00049              go.gameObject.SetActive(true);
00050
00051              go.GetComponentInChildren<Text>().text = (string)compleatedQuests[compleatedQuests.Keys.
ElementAt(i)][compleatedQuests[compleatedQuests.Keys.ElementAt(i)].Length - 1];
00052              go.GetComponentInChildren<Text>().color = Color.yellow;
00053
00054              var key = compleatedQuests.Keys.ElementAt(i);
00055              go.GetComponent<Button>().onClick.AddListener(() => QuestAchieved(key));
00056          }
00057
00058          for (int i = 0; i < currentQuests.Count; i++)
00059          {
00060              var go = Instantiate(questButton);
00061
00062              go.transform.SetParent(scrollObject.transform, false);
00063
00064              go.gameObject.SetActive(true);
00065
00066              go.GetComponentInChildren<Text>().text = (string)currentQuests[currentQuests.Keys.ElementAt
(i)][currentQuests[currentQuests.Keys.ElementAt(i)].Length - 1];
00067              go.GetComponentInChildren<Text>().color = Color.red;
00068          }
00069
00070          for (int i = 0; i < compleatClaimedQuests.Count; i++)
00071          {
00072              var go = Instantiate(questButton);
00073
00074              go.transform.SetParent(scrollObject.transform, false);
00075
00076              go.gameObject.SetActive(true);
00077
00078              go.GetComponentInChildren<Text>().text = (string)compleatClaimedQuests[
compleatClaimedQuests.Keys.ElementAt(i)][compleatClaimedQuests[compleatClaimedQuests.Keys.ElementAt(i)].Length - 1];
00079              go.GetComponentInChildren<Text>().color = Color.green;
00080          }
00081      }
00082  }
```

6.47.2.2 QuestAchieved()

```
void BeeGame.Inventory.QuestBookInventory.QuestAchieved (
    string key ) [private]
```

Definition at line 84 of file [QuestBookInventory.cs](#).

```

00085      {
00086          Quests.ClaimQuest(key);
00087          AddQuests();
00088          Serialization.Serialization.SaveQuests();
00089      }
```

6.47.2.3 Start()

```
void BeeGame.Inventory.QuestBookInventory.Start ( ) [protected]
```

Definition at line 20 of file [QuestBookInventory.cs](#).

```
00021     {
00022         AddQuests ();
00023     }
```

6.47.2.4 ToggleInventory()

```
override void BeeGame.Inventory.QuestBookInventory.ToggleInventory (
    Inventory inv) [virtual]
```

Opens and closes this inventory

Parameters

<i>inv</i>	
------------	--

Reimplemented from [BeeGame.Inventory.Inventory](#).

Definition at line 96 of file [QuestBookInventory.cs](#).

```
00097     {
00098         thisInventoryOpen = !thisInventoryOpen;
00099
00100         isAnotherInventoryOpen = thisInventoryOpen;
00101
00102         if (this.gameObject.activeInHierarchy && !thisInventoryOpen)
00103         {
00104             chestOpen = false;
00105
00106             /* removes all of the items from this inventory
00107
00108             /* tells item that inventory has been closed
00109             myItem.OpenItemInventory();
00110             Cursor.lockState = CursorLockMode.Locked;
00111             Cursor.visible = false;
00112             /* destroys this as it is not needed
00113             Destroy(this.gameObject);
00114         }
00115         else
00116         {
00117             chestOpen = true;
00118
00119             /* hides and locks the cursor
00120             Cursor.lockState = CursorLockMode.None;
00121             Cursor.visible = true;
00122         }
00123     }
```

6.47.2.5 Update()

```
void BeeGame.Inventory.QuestBookInventory.Update ( ) [protected]
```

Definition at line 25 of file [QuestBookInventory.cs](#).

```
00026     {
00027         if (GetButtonDown("Close Menu/Inventory"))
00028             ToggleInventory(this);
00029     }
```

6.47.3 Member Data Documentation

6.47.3.1 myItem

`QuestBook` BeeGame.Inventory.QuestBookInventory.myItem [package]

Definition at line 18 of file [QuestBookInventory.cs](#).

6.47.3.2 questButton

`Button` BeeGame.Inventory.QuestBookInventory.questButton

Definition at line 16 of file [QuestBookInventory.cs](#).

6.47.3.3 scrollObject

`GameObject` BeeGame.Inventory.QuestBookInventory.scrollObject

Definition at line 17 of file [QuestBookInventory.cs](#).

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/ItemInventory/QuestBookInventory.cs

6.48 BeeGame.Quest.QuestEvents Class Reference

Public Member Functions

- delegate void [QuestEventHandler](#) (int quest)

Static Public Member Functions

- static void [CallItemPickupEvent](#) (int id)
- static void [CallItemCraftedEvent](#) (int id)
- static void [CallBeeCraftedEvent](#) ([BeeSpecies](#) species)
- static void [CallPureBeeCraftedEvent](#) ([BeeSpecies](#) species1, [BeeSpecies](#) species2)

Events

- static [QuestEventHandler](#) itemCraftedEvent = [Quests.ReturnCraftingTableQuest](#)
- static [QuestEventHandler](#) itemPickupEvent = [Quests.ReturnPickupQuest](#)
- static [BeeQuestEventHandler](#) beeCraftedEvent = [Quests.ReturnBeeCraftingQuest](#)
- static [PureBeeQuestEventHandler](#) pureBeeCraftedEvent = [Quests.ReturnPureBreadBeeCraftingQuest](#)

Private Member Functions

- delegate void `BeeQuestEventHandler` (`BeeSpecies` species)
- delegate void `PureBeeQuestEventHandler` (`BeeSpecies` species1, `BeeSpecies` species2)

6.48.1 Detailed Description

Definition at line 10 of file [QuestEvents.cs](#).

6.48.2 Member Function Documentation

6.48.2.1 BeeQuestEventHandler()

```
delegate void BeeGame.Quest.QuestEvents.BeeQuestEventHandler (
    BeeSpecies species) [private]
```

6.48.2.2 CallBeeCraftedEvent()

```
static void BeeGame.Quest.QuestEvents.CallBeeCraftedEvent (
    BeeSpecies species) [static]
```

Definition at line 32 of file [QuestEvents.cs](#).

```
00033     {
00034         beeCraftedEvent(species);
00035     }
```

6.48.2.3 CallItemCraftedEvent()

```
static void BeeGame.Quest.QuestEvents.CallItemCraftedEvent (
    int id) [static]
```

Definition at line 27 of file [QuestEvents.cs](#).

```
00028     {
00029         itemCraftedEvent(id);
00030     }
```

6.48.2.4 CallItemPickupEvent()

```
static void BeeGame.Quest.QuestEvents.CallItemPickupEvent (
    int id ) [static]
```

Definition at line 22 of file [QuestEvents.cs](#).

```
00023     {
00024         itemPickupEvent(id);
00025     }
```

6.48.2.5 CallPureBeeCraftedEvent()

```
static void BeeGame.Quest.QuestEvents.CallPureBeeCraftedEvent (
    BeeSpecies species1,
    BeeSpecies species2 ) [static]
```

Definition at line 37 of file [QuestEvents.cs](#).

```
00038     {
00039         pureBeeCraftedEvent(species1, species2);
00040     }
```

6.48.2.6 PureBeeQuestEventHandler()

```
delegate void BeeGame.Quest.QuestEvents.PureBeeQuestEventHandler (
    BeeSpecies species1,
    BeeSpecies species2 ) [private]
```

6.48.2.7 QuestEventHandler()

```
delegate void BeeGame.Quest.QuestEvents.QuestEventHandler (
    int quest )
```

6.48.3 Event Documentation

6.48.3.1 beeCraftedEvent

```
BeeQuestEventHandler BeeGame.Quest.QuestEvents.beeCraftedEvent = Quests.ReturnBeeCraftingQuest
[static], [private]
```

Definition at line 17 of file [QuestEvents.cs](#).

6.48.3.2 itemCraftedEvent

```
QuestEventHandler BeeGame.Quest.QuestEvents.itemCraftedEvent = Quests.ReturnCraftingTableQuest  
[static], [private]
```

Definition at line 13 of file [QuestEvents.cs](#).

6.48.3.3 itemPickupEvent

```
QuestEventHandler BeeGame.Quest.QuestEvents.itemPickupEvent = Quests.ReturnPickupQuest [static],  
[private]
```

Definition at line 14 of file [QuestEvents.cs](#).

6.48.3.4 pureBeeCraftedEvent

```
PureBeeQuestEventHandler BeeGame.Quest.QuestEvents.pureBeeCraftedEvent = Quests.ReturnPure←  
BreadBeeCraftingQuest [static], [private]
```

Definition at line 20 of file [QuestEvents.cs](#).

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Quest/[QuestEvents.cs](#)

6.49 BeeGame.Quest.Quests Class Reference

Static Public Member Functions

- static Dictionary< string, object[]> [ReturnCompleatedQuests \(\)](#)
- static Dictionary< string, object[]> [ReturnCompleatedClaimedQuests \(\)](#)
- static Dictionary< string, object[]> [ReturnCurrentQuests \(\)](#)
- static Dictionary< string, object[]> [ReturnLockedQuests \(\)](#)
- static void [LoadQuests](#) (Dictionary< string, object[]> compleated, Dictionary< string, object[]> compleatedNotCollected, Dictionary< string, object[]> inProgress, Dictionary< string, object[]> locked)
- static void [ClaimQuest](#) (string key)
- static void [AddQuest](#) (string quest, [Item](#) result, string nextQuest)
- static void [ReturnPickupQuest](#) (int pickupID)
- static void [ReturnCraftingTableQuest](#) (int craftedItemID)
- static void [ReturnBeeCraftingQuest](#) ([BeeSpecies](#) primaryBeeSpecies)
- static void [ReturnPureBreadBeeCraftingQuest](#) ([BeeSpecies](#) primaryBeeSpecies, [BeeSpecies](#) secondary←
[BeeSpecies](#))

Static Private Member Functions

- static void [UnlockQuests](#) (string key)

Static Private Attributes

- static Dictionary< string, object[] > **compleatedQuests** = new Dictionary<string, object[]>()
Quests that the player has compleated
- static Dictionary< string, object[] > **compleatedUnclaimedQuests** = new Dictionary<string, object[]>()
Quests that have been compleated but the rewards have not been claimed
- static Dictionary< string, object[] > **currentQuests**
Quests the player is currently finishing
- static Dictionary< string, object[] > **lockedQuests**
Quests that the player does not have accest to yet

6.49.1 Detailed Description

Definition at line 10 of file [Quests.cs](#).

6.49.2 Member Function Documentation

6.49.2.1 AddQuest()

```
static void BeeGame.Quest.Quests.AddQuest (
    string quest,
    Item result,
    string nextQuest ) [static]
```

Definition at line 77 of file [Quests.cs](#).

```
00078      {
00079          if(!lockedQuests.ContainsKey(quest))
00080          {
00081              lockedQuests.Add(quest, new object[] { result, nextQuest });
00082              return;
00083          }
00084          else
00085          {
00086              lockedQuests[quest] = new object[] { result, nextQuest };
00087          }
00088          throw new QuestAlreadyExistsException($"Warning Quest: {quest},
00089 is already a quest and the old has been overriden!");
00090      }
```

6.49.2.2 ClaimQuest()

```
static void BeeGame.Quest.Quests.ClaimQuest (
    string key ) [static]
```

Definition at line 67 of file [Quests.cs](#).

```
00068      {
00069          var item = compleatedUnclaimedQuests[key];
00070          compleatedQuests.Add(key, compleatedUnclaimedQuests[
00071          key]);
00072          compleatedUnclaimedQuests.Remove(key);
00073          var temp = UnityEngine.Object.Instantiate(UnityEngine.Resources.Load("
Prefabs/ItemGameObject") as UnityEngine.GameObject, UnityEngine.Object.FindObjectOfType<
Player.PlayerMove>().transform.position + UnityEngine.Vector3.one,
UnityEngine.Quaternion.identity);
00074          temp.GetComponent<ItemGameObject>().item = (Item)item[0];
00075      }
```

6.49.2.3 LoadQuests()

```
static void BeeGame.Quest.Quests.LoadQuests (
    Dictionary< string, object[]> compleated,
    Dictionary< string, object[]> compleatedNotCollected,
    Dictionary< string, object[]> inProgress,
    Dictionary< string, object[]> locked ) [static]
```

Definition at line 59 of file [Quests.cs](#).

```
00060      {
00061          compleatedQuests = compleated;
00062          compleatedUnclaimedQuests = compleatedNotCollected;
00063          currentQuests = inProgress;
00064          lockedQuests = locked;
00065      }
```

6.49.2.4 ReturnBeeCraftingQuest()

```
static void BeeGame.Quest.Quests.ReturnBeeCraftingQuest (
    BeeSpecies primaryBeeSpecies ) [static]
```

Definition at line 108 of file [Quests.cs](#).

```
00109      {
00110          if (currentQuests.ContainsKey($"BeeCrafted: {primaryBeeSpecies}"))
00111          {
00112              UnlockQuests($"BeeCrafted: {primaryBeeSpecies}");
00113          }
00114      }
```

6.49.2.5 ReturnCompleatedClaimedQuests()

```
static Dictionary<string, object[]> BeeGame.Quest.Quests.ReturnCompleatedClaimedQuests ( )
[static]
```

Definition at line 44 of file [Quests.cs](#).

```
00045      {
00046          return compleatedQuests;
00047      }
```

6.49.2.6 ReturnCompleatedQuests()

```
static Dictionary<string, object[]> BeeGame.Quest.Quests.ReturnCompleatedQuests ( ) [static]
```

Definition at line 39 of file [Quests.cs](#).

```
00040      {
00041          return compleatedUnclaimedQuests;
00042      }
```

6.49.2.7 ReturnCraftingTableQuest()

```
static void BeeGame.Quest.Quests.ReturnCraftingTableQuest (
    int craftedItemID ) [static]
```

Definition at line 100 of file [Quests.cs](#).

```
00101      {
00102          if (currentQuests.ContainsKey($"Crafted: {craftedItemID}"))
00103          {
00104              UnlockQuests($"Crafted: {craftedItemID}");
00105          }
00106      }
```

6.49.2.8 ReturnCurrentQuests()

```
static Dictionary<string, object[]> BeeGame.Quest.Quests.ReturnCurrentQuests () [static]
```

Definition at line 49 of file [Quests.cs](#).

```
00050      {
00051          return currentQuests;
00052      }
```

6.49.2.9 ReturnLockedQuests()

```
static Dictionary<string, object[]> BeeGame.Quest.Quests.ReturnLockedQuests () [static]
```

Definition at line 54 of file [Quests.cs](#).

```
00055      {
00056          return lockedQuests;
00057      }
```

6.49.2.10 ReturnPickupQuest()

```
static void BeeGame.Quest.Quests.ReturnPickupQuest (
    int pickupID ) [static]
```

Definition at line 92 of file [Quests.cs](#).

```
00093      {
00094          if (currentQuests.ContainsKey($"Pickup: {pickupID}"))
00095          {
00096              UnlockQuests($"Pickup: {pickupID}");
00097          }
00098      }
```

6.49.2.11 ReturnPureBreadBeeCraftingQuest()

```
static void BeeGame.Quest.Quests.ReturnPureBreadBeeCraftingQuest (
    BeeSpecies primaryBeeSpecies,
    BeeSpecies secondaryBeeSpecies )  [static]
```

Definition at line 116 of file [Quests.cs](#).

```
00117      {
00118          if (currentQuests.ContainsKey($"PureBredBee: {primaryBeeSpecies}
00119 {secondaryBeeSpecies}"))
00120          {
00121              UnlockQuests($"PureBredBee: {primaryBeeSpecies} {secondaryBeeSpecies}");
00122          }
00123 }
```

6.49.2.12 UnlockQuests()

```
static void BeeGame.Quest.Quests.UnlockQuests (
    string key )  [static], [private]
```

Definition at line 124 of file [Quests.cs](#).

```
00125      {
00126          compleatedUnclaimedQuests.Add(key,
00127 currentQuests[key]);
00128          var objArray = currentQuests[key];
00129          if(objArray.Length > 2)
00130          {
00131              var next = objArray[1];
00132              if (next is string[] sa)
00133              {
00134                  foreach (var q in sa)
00135                  {
00136                      currentQuests.Add(q, lockedQuests[q]);
00137                      lockedQuests.Remove(q);
00138                  }
00139              }
00140          }
00141          else if (next is string ss)
00142          {
00143              if (lockedQuests.ContainsKey(ss))
00144              {
00145                  currentQuests.Add(ss, lockedQuests[ss]);
00146                  lockedQuests.Remove(ss);
00147              }
00148          }
00149      }
00150      currentQuests.Remove(key);
00151      Serialization.Serialization.SaveQuests();
00152  }
00153 }
```

6.49.3 Member Data Documentation

6.49.3.1 compleatedQuests

```
Dictionary<string, object[]> BeeGame.Quest.Quests.compleatedQuests = new Dictionary<string, object[]>() [static], [private]
```

Quests that the player has compleated

Definition at line 15 of file [Quests.cs](#).

6.49.3.2 compleatedUnclaimedQuests

```
Dictionary<string, object[]> BeeGame.Quest.Quests.compleatedUnclaimedQuests = new Dictionary<string, object[]>() [static], [private]
```

Quests that have been compleated but the rewards have not been claimed

Definition at line 20 of file [Quests.cs](#).

6.49.3.3 currentQuests

```
Dictionary<string, object[]> BeeGame.Quest.Quests.currentQuests [static], [private]
```

Initial value:

```
= new Dictionary<string, object[]>()
{
    { $"Pickup: {Wood.ID}", new object[] {new CraftingTable(), $"Crafted: {Grass.ID}", "Pickup A
Wood Block" } },
    { $"BeeCrafted: {BeeSpecies.COMMON}", new object[] {new CraftingTable(), "Make a Common Bee"} }
}
```

Quests the player is currently finishing

Definition at line 25 of file [Quests.cs](#).

6.49.3.4 lockedQuests

```
Dictionary<string, object[]> BeeGame.Quest.Quests.lockedQuests [static], [private]
```

Initial value:

```
= new Dictionary<string, object[]>()
{
    { $"Crafted: {Grass.ID}", new object[] {new Dirt(), "nothing", "Use some Dirt in the Workbench"
} }
}
```

Quests that the player does not have accest to yet

Definition at line 34 of file [Quests.cs](#).

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Quest/[Quests.cs](#)

6.50 BeeGame.Resources.Resources Class Reference

A strongly-typed resource class, for looking up localized strings, etc.

Package Functions

- [Resources \(\)](#)

Static Package Functions

- static Dictionary< string, Sprite > [GetSprites \(\)](#)
Gets the sprites from Sprites.dat file and loads them
- static Dictionary< string, GameObject > [GetPrefabs \(\)](#)
Gets the prefabs from Prefabs.dat file and loads them

Properties

- static global::System.Resources.ResourceManager [ResourceManager](#) [get]
Returns the cached ResourceManager instance used by this class.
- static global::System.Globalization.CultureInfo [Culture](#) [get, set]
Overrides the current thread's CurrentUICulture property for all resource lookups using this strongly typed resource class.
- static byte [] [Prefabs](#) [get]
Looks up a localized resource of type System.Byte[].
- static byte [] [Sprites](#) [get]
Looks up a localized resource of type System.Byte[].

Static Private Attributes

- static global::System.Resources.ResourceManager [resourceMan](#)
- static global::System.Globalization.CultureInfo [resourceCulture](#)

6.50.1 Detailed Description

A strongly-typed resource class, for looking up localized strings, etc.

Definition at line 26 of file [Resources.Designer.cs](#).

6.50.2 Constructor & Destructor Documentation

6.50.2.1 Resources()

```
BeeGame.Resources.Resources ( ) [package]
```

Definition at line 33 of file [Resources.Designer.cs](#).

```
00033          {  
00034      }
```

6.50.3 Member Function Documentation

6.50.3.1 GetPrefabs()

```
static Dictionary<string, GameObject> BeeGame.Resources.Resources.GetPrefabs () [static], [package]
```

Gets the prefabs from Prefabs.dat file and loads them

Returns

A dictionary of the Sprite with its reference name (dictionary key)

Definition at line 130 of file [Resources.Designer.cs](#).

```
00131      {
00132          string[] splitCharacters = new string[] { "," };
00133          object obj = ResourceManager.GetObject("Prefabs",
00134          resourceCulture);
00135          string text = System.Text.Encoding.Default.GetString((byte[])obj);
00136          text = text.Remove(0, 3);
00137          string lineText = "";
00138          string[] splitText;
00139          Dictionary<string, GameObject> objects = new Dictionary<string, GameObject>();
00140
00141          /* goes through all characters in the file
00142          for (int i = 0; i < text.Length; i++)
00143          {
00144              /* when their is a new line the path for that sprite is found
00145              if (text[i] != '\n')
00146              {
00147                  lineText += text[i];
00148              }
00149              else
00150              {
00151                  /* when their is a new line the path for that sprite is found
00152                  splitText = lineText.Split(splitCharacters, StringSplitOptions.RemoveEmptyEntries);
00153                  lineText = "";
00154                  objects.Add(splitText[0], UnityEngine.Resources.Load("Prefabs/" + splitText[1].Remove(splitText[1].Length - 1, 1)) as GameObject);
00155              }
00156          }
00157
00158          splitText = lineText.Split(splitCharacters, StringSplitOptions.RemoveEmptyEntries);
00159          lineText = "";
00160          objects.Add(splitText[0], UnityEngine.Resources.Load("Prefabs/" + splitText[1]) as
00161          GameObject);
00162
00163          return objects;
00164      }
```

6.50.3.2 GetSprites()

```
static Dictionary<string, Sprite> BeeGame.Resources.Resources.GetSprites () [static], [package]
```

Gets the sprites from Sprites.dat file and loads them

Returns

A dictionary of the Sprite with its reference name (dictionary key)

Definition at line 88 of file [Resources.Designer.cs](#).

```

00089      {
00090          string[] splitCharacters = new string[] { "," };
00091          object obj = ResourceManager.GetObject("Sprites",
00092          resourceCulture);
00093          /* gets the text from the spries.dat file
00094          string text = System.Text.Encoding.Default.GetString((byte[])obj);
00095          string lineText = "";
00096          string[] splitText;
00097          Texture2D tex;
00098          Dictionary<string, Sprite> sprites = new Dictionary<string, Sprite>();
00099
00100         /* goes through all characters in the file
00101         for (int i = 0; i < text.Length; i++)
00102         {
00103             /* when their is a new line the path for that sprite is found
00104             if (text[i] != '\n')
00105             {
00106                 lineText += text[i];
00107             }
00108             else
00109             {
00110                 splitText = lineText.Split(splitCharacters, StringSplitOptions.RemoveEmptyEntries);
00111                 lineText = "";
00112                 tex = UnityEngine.Resources.Load("Sprites/" + splitText[1].Remove(splitText[
00113                     1].Length - 1, 1)) as Texture2D;
00114                     /*
00115                     need to craft a sprite from a texture 2D because
00116                     Unity wont allow images to be loaded directly as sprites at runtime...for some reason
00117                     sprites.Add(splitText[0], Sprite.Create(tex, new UnityEngine.Rect(0, 0, tex.
00118                     width, tex.height), Vector2.zero));
00119
00120                     splitText = lineText.Split(splitCharacters, StringSplitOptions.RemoveEmptyEntries);
00121                     lineText = "";
00122                     tex = UnityEngine.Resources.Load("Sprites/" + splitText[1]) as Texture2D;
00123                     sprites.Add(splitText[0], Sprite.Create(tex, new UnityEngine.Rect(0, 0, tex.width,
00124                     tex.height), Vector2.zero));
00125
00126             }
00127
00128         }
00129
00130         return sprites;
00131     }

```

6.50.4 Member Data Documentation

6.50.4.1 resourceCulture

```
global.System.Globalization.CultureInfo BeeGame.Resources.Resources.resourceCulture [static],
[private]
```

Definition at line 30 of file [Resources.Designer.cs](#).

6.50.4.2 resourceMan

```
global.System.Resources.ResourceManager BeeGame.Resources.Resources.resourceMan [static],
[private]
```

Definition at line 28 of file [Resources.Designer.cs](#).

6.50.5 Property Documentation

6.50.5.1 Culture

```
global.System.Globalization.CultureInfo BeeGame.Resources.Resources.Culture [static], [get],  
[set], [package]
```

Overrides the current thread's CurrentUICulture property for all resource lookups using this strongly typed resource class.

Definition at line 55 of file [Resources.Designer.cs](#).

6.50.5.2 Prefabs

```
byte [] BeeGame.Resources.Resources.Prefabs [static], [get], [package]
```

Looks up a localized resource of type System.Byte[].

Definition at line 67 of file [Resources.Designer.cs](#).

6.50.5.3 ResourceManager

```
global.System.Resources.ResourceManager BeeGame.Resources.Resources.ResourceManager [static],  
[get], [package]
```

Returns the cached ResourceManager instance used by this class.

Definition at line 40 of file [Resources.Designer.cs](#).

6.50.5.4 Sprites

```
byte [] BeeGame.Resources.Resources.Sprites [static], [get], [package]
```

Looks up a localized resource of type System.Byte[].

Definition at line 77 of file [Resources.Designer.cs](#).

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Resources/[Resources.Designer.cs](#)

6.51 BeeGame.Terrain.Chunks.SaveChunk Class Reference

Saves a [Chunks](#) modified Blocks for save optimisation

Public Member Functions

- [SaveChunk \(Block\[, ,\] blockArray\)](#)

Will search all the the given Blocks for modified blocks

Public Attributes

- [Dictionary< ChunkWorldPos, Block > blocks = new Dictionary<ChunkWorldPos, Block>\(\)](#)
Blocks to be saved

6.51.1 Detailed Description

Saves a [Chunks](#) modified Blocks for save optimisation

Definition at line 12 of file [SaveChunk.cs](#).

6.51.2 Constructor & Destructor Documentation

6.51.2.1 SaveChunk()

```
BeeGame.Terrain.Chunks.SaveChunk (
    Block blockArray[, , ] )
```

Will search all the the given Blocks for modified blocks

Parameters

<code>blockArray</code>	Chunks blocks (Must be [16, 16, 16])
-------------------------	--

Definition at line 23 of file [SaveChunk.cs](#).

```
00024     {
00025         for (int x = 0; x < Chunk.chunkSize; x++)
00026         {
00027             for (int y = 0; y < Chunk.chunkSize; y++)
00028             {
00029                 for (int z = 0; z < Chunk.chunkSize; z++)
00030                 {
00031                     /* if the block has changed save it
00032                     if (blockArray[x, y, z].changed)
00033                         blocks.Add(new ChunkWorldPos(x, y, z), blockArray[x, y, z]);
00034                 }
00035             }
00036         }
00037     }
```

6.51.3 Member Data Documentation

6.51.3.1 blocks

```
Dictionary<ChunkWorldPos, Block> BeeGame.Terrain.Chunks.SaveChunk.blocks = new Dictionary<ChunkWorldPos, Block>()
```

Blocks to be saved

Definition at line 17 of file [SaveChunk.cs](#).

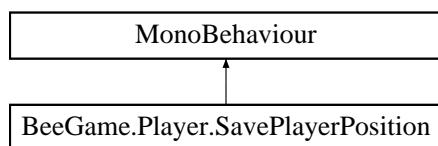
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/[SaveChunk.cs](#)

6.52 BeeGame.Player.SavePlayerPosition Class Reference

Saves the player position

Inheritance diagram for BeeGame.Player.SavePlayerPosition:



Private Member Functions

- void [Update \(\)](#)
Saves the player every 1000 frames

Private Attributes

- int [counter](#) = 0
Timer for saving the player

6.52.1 Detailed Description

Saves the player position

Definition at line 9 of file [SavePlayerPosition.cs](#).

6.52.2 Member Function Documentation

6.52.2.1 Update()

```
void BeeGame.Player.SavePlayerPosition.Update ( ) [private]
```

Saves the player every 1000 frames

Definition at line 19 of file [SavePlayerPosition.cs](#).

```
00020     {
00021         if(counter == 0)
00022     {
00023         counter = 1000;
00024         Serialization.Serialization.SavePlayerPosition(transform);
00025         //print("saved player");
00026     }
00027     counter--;
00028 }
```

6.52.3 Member Data Documentation

6.52.3.1 counter

```
int BeeGame.Player.SavePlayerPosition.counter = 0 [private]
```

Timer for saving the player

Definition at line 14 of file [SavePlayerPosition.cs](#).

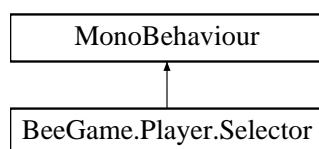
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/[SavePlayerPosition.cs](#)

6.53 BeeGame.Player.Selector Class Reference

Moves the Block selector

Inheritance diagram for BeeGame.Player.Selector:



Public Attributes

- GameObject **selector**
Selector
- PlayerInventory **playerInventory**
Player Inventory
- LayerMask **layers**
Layers for the selector to look at
- int **selectedHotbarSlot** = 27
What slot in the hotbar is selected

Private Member Functions

- void [Awake \(\)](#)
Make the selector
- void [FixedUpdate \(\)](#)
Updates the selector if an inventory is not open
- void [Update \(\)](#)
Breaks and places a Block if an inventory is no open
- void [UpdateSelector \(\)](#)
Updates teh selectors position
- void [SelectedSlot \(\)](#)
Changes what slot in the hotbar is currently selected by the player
- void [BreakBlock \(\)](#)
Breaks the Block in the selectors postion
- void [PlaceBlock \(\)](#)
Places s Block in the selector postion

Private Attributes

- RaycastHit [hit](#)
Where the raycast hit

6.53.1 Detailed Description

Moves the Block selector

Definition at line [15](#) of file [Selector.cs](#).

6.53.2 Member Function Documentation

6.53.2.1 Awake()

```
void BeeGame.Player.Selector.Awake ( ) [private]
```

Make the selector

Definition at line [47](#) of file [Selector.cs](#).

```
00048      {
00049          selector = Instantiate(selector);
00050      }
```

6.53.2.2 BreakBlock()

```
void BeeGame.Player.Selector.BreakBlock ( ) [private]
```

Breaks the Block in the selector's position

Definition at line 123 of file [Selector.cs](#).

```
00124      {
00125          Chunk chunk = GetChunk(selector.transform.position);
00126
00127          Block block = chunk.world.GetBlock((int)selector.transform.position.x, (int)
00128              selector.transform.position.y, (int)selector.transform.position.z);
00129
00130          if (!block.breakable)
00131              return;
00132
00133          chunk.world.SetBlock((int)selector.transform.position.x, (int)
00134              selector.transform.position.y, (int)selector.transform.position.z, new
00135              Air(), true);
00136
00137          /* set to changed so when block is placed down again it will be saved
00138          block.changed = true;
00139          block.BreakBlock(selector.transform.position);
00140      }
```

6.53.2.3 FixedUpdate()

```
void BeeGame.Player.Selector.FixedUpdate ( ) [private]
```

Updates the selector if an inventory is not open

Definition at line 55 of file [Selector.cs](#).

```
00056      {
00057          if (!isAnotherInventoryOpen)
00058              UpdateSelector();
00059      }
```

6.53.2.4 PlaceBlock()

```
void BeeGame.Player.Selector.PlaceBlock ( ) [private]
```

Places a Block in the selector's position

Definition at line 141 of file [Selector.cs](#).

```
00142      {
00143          Chunk chunk = GetChunk(selector.transform.position);
00144
00145          if (chunk == null)
00146              return;
00147
00148          transform.parent.GetComponentInChildren<PlayerInventory>().GetItemFromHotBar(
00149              selectedHotbarSlot, out var item);
00150
00151          if (item != null)
00152          {
00153              if (item.InteractWithObject())
00154                  item.InteractWithItem(playerInventory);
00155          }
00156      }
```

```

00156         }
00157         else if (item.placeable)
00158         {
00159             chunk.world.SetBlock((int)(selector.transform.position.x +
00160             hit.normal.x), (int)(selector.transform.position.y + hit.normal.y), (int)(
00161             selector.transform.position.z + hit.normal.z), (Block)item.CloneObject(), true);
00162         }
00163         if (!chunk.GetBlock((int)selector.transform.position.x - chunk.
00164             chunkWorldPos.x, (int)selector.transform.position.y - chunk.
00165             chunkWorldPos.y, (int)selector.transform.position.z - chunk.
00166             chunkWorldPos.z).InteractWithBlock(
00167                 playerInventory))
00168             return;
00169     }

```

6.53.2.5 SelectedSlot()

void BeeGame.Player.Selector.SelectedSlot () [private]

Changes what slot in the hotbar is currently selected by the player

Definition at line 98 of file [Selector.cs](#).

```

00099     {
00100         /* adds 1 to the selected slot and if that is out of range set it to the first hotbar slot
00101         if(Input.GetAxis("Mouse ScrollWheel") > 0)
00102         {
00103             selectedHotbarSlot += 1;
00104             if (selectedHotbarSlot == 36)
00105                 selectedHotbarSlot = 27;
00106         }
00107         /* removes one from the hotbar selector and if the selector would be inside the inventory set
00108         it to the last slot in the hotbar
00109         else if (Input.GetAxis("Mouse ScrollWheel") < 0)
00110         {
00111             selectedHotbarSlot -= 1;
00112             if (selectedHotbarSlot == 26)
00113                 selectedHotbarSlot = 35;
00114         }
00115         transform.parent.GetComponentInChildren<PlayerInventory>().
00116         SelectedSlot(selectedHotbarSlot);
00117     }

```

6.53.2.6 Update()

void BeeGame.Player.Selector.Update () [private]

Breaks and places a Block if an inventory is no open

Definition at line 64 of file [Selector.cs](#).

```

00065     {
00066         if (!isAnotherInventoryOpen)
00067         {
00068             if (GetButtonDown("Break Block"))
00069                 BreakBlock();
00070             if (GetButtonDown("Place"))
00071                 PlaceBlock();
00072         }
00073     }

```

6.53.2.7 UpdateSelector()

```
void BeeGame.Player.Selector.UpdateSelector ( ) [private]
```

Updates teh selectors position

Definition at line 80 of file [Selector.cs](#).

```
00081      {
00082          if (Physics.Raycast(transform.position, transform.forward, out hit, 15,
00083          layers))
00084          {
00085              selector.SetActive(true);
00086              selector.transform.position = GetBlockPos(hit);
00087              /*selector.SetActive(BlockInPosition(GetBlockPos(hit),
00088              hit.collider.GetComponent<Chunk>()));
00089          }
00090          else
00091          {
00092              selector.SetActive(false);
00093          }
00094      }
```

6.53.3 Member Data Documentation

6.53.3.1 hit

```
RaycastHit BeeGame.Player.Selector.hit [private]
```

Where the raycast hit

Definition at line 35 of file [Selector.cs](#).

6.53.3.2 layers

```
LayerMask BeeGame.Player.Selector.layers
```

Layers for the selector to look at

Definition at line 31 of file [Selector.cs](#).

6.53.3.3 playerInventory

```
PlayerInventory BeeGame.Player.Selector.playerInventory
```

[Player Inventory](#)

Definition at line 26 of file [Selector.cs](#).

6.53.3.4 selectedHotbarSlot

```
int BeeGame.Player.Selector.selectedHotbarSlot = 27
```

What slot in the hotbar is selected

Definition at line 40 of file [Selector.cs](#).

6.53.3.5 selector

```
GameObject BeeGame.Player.Selector.selector
```

[Selector](#)

Definition at line 21 of file [Selector.cs](#).

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/[Selector.cs](#)

6.54 BeeGame.Serialization.Serialization Class Reference

Serializes and Deserialises things

Static Public Member Functions

- static void [MakeDirectory\(\)](#)
Sets the paths for the save files
- static void [DeleteFile\(string fileName\)](#)
Deletes the given file if it exists, Starts in Application.dataPath
- static void [SavePlayerPosition\(Transform positon\)](#)
Saves the player positon, rotation, and scale
- static void [LoadPlayerPosition\(Transform playerTransform\)](#)
Loads the players positon, roatation, and scale if it has previously been saved
- static void [SerializeInventory\(Inventory.Inventory inventory, string inventoryName\)](#)
Serializes a given Inventory
- static void [DeSerializeInventory\(Inventory.Inventory inventory, string inventoryName\)](#)
Deserializesd an Inventory from its name into a given inventory
- static void [SaveChunk\(Chunk chunk\)](#)
Saves a given Chunk if a block in it has been changed
- static bool [LoadChunk\(Chunk chunk\)](#)
Load a Chunk
- static string [FileName\(ChunkWorldPos pos\)](#)
Sets the file name of the Chunk
- static void [SaveQuests\(\)](#)
- static void [LoadQuests\(\)](#)

Static Public Attributes

- static string **worldName** = "World"
Name if the world. If multiple world are ever added
- static string **saveFolderName** = "Saves"
Save folder

Static Private Member Functions

- static void **SaveFile** (object obj, string file)
Saves the given data in the given file
- static object **LoadFile** (string file)
Loads the file at the given path

Static Private Attributes

- static string **savePath**
Path to save things

6.54.1 Detailed Description

Serializes and Deserialises things

Binary serialization is SLOW try to only serialize only what is absolutely necessary

Definition at line 19 of file [Serialization.cs](#).

6.54.2 Member Function Documentation

6.54.2.1 DeleteFile()

```
static void BeeGame.Serialization.Serialization.DeleteFile (
    string fileName) [static]
```

Deletes the given file if it exists, Starts in Application.dataPath

Parameters

<code>fileName</code>	File to delete
-----------------------	----------------

Definition at line 51 of file [Serialization.cs](#).

```
00052      {
00053          string[] file = Directory.GetFiles(Application.dataPath + "/Saves", "*.dat", SearchOption.
AllDirectories);
00054 }
```

```

00055     string[] splitCharacters = { "/", "\\", ".dat" };
00056
00057     for (int i = 0; i < file.Length; i++)
00058     {
00059         string[] temp = file[i].Split(splitCharacters, System.StringSplitOptions.
00060         RemoveEmptyEntries);
00061
00062         if (temp[temp.Length - 1] == fileName)
00063         {
00064             File.Delete(file[i]);
00065
00066             return;
00067         }
00068     }

```

6.54.2.2 DeSerializeInventory()

```

static void BeeGame.Serialization.Serialization.DeSerializeInventory (
    Inventory.Inventory inventory,
    string inventoryName ) [static]

```

Deserializesd an [Inventory](#) from its name into a given *inventory*

Parameters

<i>inventory</i>	Inventory to apply the data to
<i>inventoryName</i>	Inventory to deserialize

Definition at line 132 of file [Serialization.cs](#).

```

00133     {
00134         /* make the path
00135         string inventorySavePath = $"{savePath}/Inventorys/{inventoryName}.dat";
00136
00137         /* checks that the file exists
00138         if (!File.Exists(inventorySavePath))
00139         {
00140             for (int i = 0; i < inventory.items.itemsInInventory.Length; i++)
00141             {
00142                 inventory.items.itemsInInventory[i] = null;
00143             }
00144
00145             SerializeInventory(inventory, inventoryName);
00146
00147             return;
00148         }
00149
00150         inventory.SetAllItems((ItemsInInventory)LoadFile($"{inventorySavePath}""
00151     ));
00151 }

```

6.54.2.3 FileName()

```

static string BeeGame.Serialization.Serialization.FileName (
    ChunkWorldPos pos ) [static]

```

Sets the file name of the Chunk

Parameters

<i>pos</i>	Position of teh Chunk
------------	-----------------------

Returns

The string of pos

Definition at line 204 of file [Serialization.cs](#).

```
00205     {
00206         return $"{pos.x}, {pos.y}, {pos.z}";
00207     }
```

6.54.2.4 LoadChunk()

```
static bool BeeGame.Serialization.Serialization.LoadChunk (
    Chunk chunk ) [static]
```

Load a Chunk

Parameters

<i>chunk</i>	
--------------	--

Returns

Definition at line 179 of file [Serialization.cs](#).

```
00180     {
00181         /* gets the save file
00182         string saveFile = $"{savePath}/{FileName(chunk.chunkWorldPos)}.dat";
00183
00184         /* if the file does not exist return false
00185         if (!File.Exists(saveFile))
00186             return false;
00187
00188         /* set all of the changed blocks in the chunk
00189         SaveChunk save = (SaveChunk)LoadFile(saveFile);
00190
00191         foreach (var block in save.blocks)
00192         {
00193             chunk.blocks[block.Key.x, block.Key.y, block.Key.z] = block.Value;
00194         }
00195
00196         return true;
00197     }
```

6.54.2.5 LoadFile()

```
static object BeeGame.Serialization.Serialization.LoadFile (
    string file ) [static], [private]
```

Loads the file at the given path

Parameters

<i>file</i>	File to load
-------------	--------------

Returns

returns the loaded file as an object

Definition at line 265 of file [Serialization.cs](#).

```

00266      {
00267          BinaryFormatter bf = new BinaryFormatter();
00268          FileStream fs = new FileStream(file, FileMode.Open);
00269
00270          try
00271          {
00272              return bf.Deserialize(fs);
00273          }
00274          catch(SerializationException e)
00275          {
00276              Debug.Log($"Deserialization Exception {e}");
00277              throw new SerializationException();
00278          }
00279          finally
00280          {
00281              fs.Close();
00282          }
00283      }

```

6.54.2.6 LoadPlayerPosition()

```
static void BeeGame.Serialization.Serialization.LoadPlayerPosition (
    Transform playerTransfom ) [static]
```

Loads the players positon, roatation, and scale if it has previously been saved

Parameters

<i>playerTransform</i>	Transform to apply the data to
------------------------	--------------------------------

Definition at line 92 of file [Serialization.cs](#).

```

00093      {
00094          string playerPosSavePath = $"{savePath}/player.dat";
00095
00096          if (!File.Exists(playerPosSavePath))
00097              return;
00098
00099          THVector3[] pos = (THVector3[])LoadFile(playerPosSavePath);
00100
00101          playerTransfom.position = pos[0];
00102          playerTransfom.rotation = (Quaternion)pos[1];
00103          playerTransfom.localScale = pos[2];
00104      }

```

6.54.2.7 LoadQuests()

```
static void BeeGame.Serialization.Serialization.LoadQuests ( ) [static]
```

Definition at line 220 of file [Serialization.cs](#).

```
00221      {
00222          string saveFile = $"{savePath}/quests.dat";
00223
00224          if (!File.Exists(saveFile))
00225              return;
00226
00227          var array = (System.Collections.Generic.Dictionary<string, object[]>[])
00228              LoadFile(saveFile);
00229          Quests.LoadQuests(array[0], array[1], array[2], array[3]);
00230
00231      }
```

6.54.2.8 MakeDirectorys()

```
static void BeeGame.Serialization.Serialization.MakeDirectorys ( ) [static]
```

Sets the paths for the save files

Definition at line 39 of file [Serialization.cs](#).

```
00040      {
00041          savePath = $"{Application.dataPath}/{saveFolderName}/{worldName}";
00042
00043          if (!(Directory.Exists(savePath)))
00044              Directory.CreateDirectory(savePath);
00045      }
```

6.54.2.9 SaveChunk()

```
static void BeeGame.Serialization.Serialization.SaveChunk (
    Chunk chunk ) [static]
```

Saves a given Chunk if a block in it has been changed

Parameters

<i>chunk</i>	
--------------	--

Definition at line 159 of file [Serialization.cs](#).

```
00160      {
00161          /* saves the blocks
00162          SaveChunk save = new SaveChunk(chunk.blocks);
00163
00164          /* if no block was changed return early
00165          if (save.blocks.Count == 0)
00166              return;
00167      }
```

```

00168     /* otherwise save the file
00169     string saveFile = $"{savePath}/{fileName(chunk.chunkWorldPos)}.dat";
00170
00171     SaveFile(save, saveFile);
00172 }
```

6.54.2.10 SaveFile()

```
static void BeeGame.Serialization.Serialization.SaveFile (
    object obj,
    string file ) [static], [private]
```

Saves the given data in the given file

Parameters

<i>obj</i>	Object to save
<i>file</i>	File path to save to

Definition at line 240 of file [Serialization.cs](#).

```

00241     {
00242         BinaryFormatter bf = new BinaryFormatter();
00243         FileStream fs = new FileStream(file, FileMode.OpenOrCreate);
00244
00245         try
00246         {
00247             bf.Serialize(fs, obj);
00248         }
00249         catch(SerializationException e)
00250         {
00251             Debug.Log($"Serialization Exception: {e}");
00252             throw new SerializationException();
00253         }
00254         finally
00255         {
00256             fs.Close();
00257         }
00258     }
```

6.54.2.11 SavePlayerPosition()

```
static void BeeGame.Serialization.Serialization.SavePlayerPosition (
    Transform positon ) [static]
```

Saves the player positon, rotation, and scale

Parameters

<i>positon</i>	Transform to get the data from
----------------	--------------------------------

Definition at line 75 of file [Serialization.cs](#).

```
00076     {
```

```

00077     THVector3[] playerTransform = new THVector3[3];
00078
00079     playerTransform[0] = positon.position;
00080     playerTransform[1] = positon.rotation.eulerAngles;
00081     playerTransform[2] = positon.localScale;
00082
00083     string playerPosSavePath = $"{savePath}/player.dat";
00084
00085     SaveFile(playerTransform, playerPosSavePath);
00086 }
```

6.54.2.12 SaveQuests()

```
static void BeeGame.Serialization.Serialization.SaveQuests () [static]
```

Definition at line 211 of file [Serialization.cs](#).

```

00212     {
00213         var array = new System.Collections.Generic.Dictionary<string, object>[] {
00214             Quests.ReturnCompledClaimedQuests(),
00215             Quests.ReturnCompledQuests(), Quests.
00216             ReturnCurrentQuests(), Quests.ReturnLockedQuests() };
00217
00218         string saveFile = $"{savePath}/quests.dat";
00219         SaveFile(array, saveFile);
00220     }
```

6.54.2.13 SerializeInventory()

```
static void BeeGame.Serialization.Serialization.SerializeInventory (
    Inventory.Inventory inventory,
    string inventoryName ) [static]
```

Serializes a given [Inventory](#)

Parameters

<i>inventory</i>	Invenotry to Serialize
<i>inventoryName</i>	Name of the inventory

The name of the inventory for the player is "PlayerInventory".

For all other ivnetorys the name is the block type + its position eg, Apiay@0, 0, 0

Definition at line 117 of file [Serialization.cs](#).

```

00118     {
00119         string inventorySavePath = $"{savePath}/Inventorys";
00120
00121         if (!Directory.Exists(inventorySavePath))
00122             Directory.CreateDirectory(inventorySavePath);
00123
00124         SaveFile(inventory.GetAllItems(), $"{inventorySavePath}/{inventoryName}.dat");
00125     }
```

6.54.3 Member Data Documentation

6.54.3.1 saveFolderName

```
string BeeGame.Serialization.Serialization.saveFolderName = "Saves" [static]
```

Save folder

Definition at line 29 of file [Serialization.cs](#).

6.54.3.2 savePath

```
string BeeGame.Serialization.Serialization.savePath [static], [private]
```

Path to save things

Definition at line 33 of file [Serialization.cs](#).

6.54.3.3 worldName

```
string BeeGame.Serialization.Serialization.worldName = "World" [static]
```

Name if the world. If multiple world are ever added

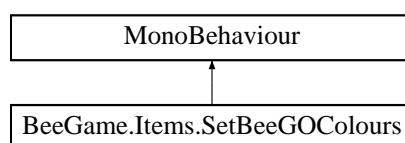
Definition at line 25 of file [Serialization.cs](#).

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Serialization/[Serialization.cs](#)

6.55 BeeGame.Items.SetBeeGOColours Class Reference

Inheritance diagram for BeeGame.Items.SetBeeGOColours:



Public Attributes

- GameObject [] [objects](#)
- GameObject [crown](#)
- GameObject [tiara](#)
- [BeeType](#) [beeType](#)
- Color [colour](#)

Protected Member Functions

- void [Start \(\)](#)

6.55.1 Detailed Description

Definition at line 11 of file [SetBeeGOColours.cs](#).

6.55.2 Member Function Documentation

6.55.2.1 Start()

```
void BeeGame.Items.SetBeeGOColours.Start () [protected]
```

Definition at line 19 of file [SetBeeGOColours.cs](#).

```
00020      {
00021          switch (beeType)
00022          {
00023              case BeeType.DRONE:
00024                  crown.SetActive(false);
00025                  tiara.SetActive(false);
00026                  break;
00027              case BeeType.PRINCESS:
00028                  crown.SetActive(false);
00029                  break;
00030          }
00031
00032          foreach (var item in objects)
00033          {
00034              item.GetComponent<Renderer>().material.SetColor("_Color", colour);
00035          }
00036      }
```

6.55.3 Member Data Documentation

6.55.3.1 beeType

```
BeeType BeeGame.Items.SetBeeGOColours.beeType
```

Definition at line 16 of file [SetBeeGOColours.cs](#).

6.55.3.2 colour

```
Color BeeGame.Items.SetBeeGOColours.colour
```

Definition at line 17 of file [SetBeeGOColours.cs](#).

6.55.3.3 crown

`GameObject BeeGame.Items.SetBeeGOColours.crown`

Definition at line 14 of file [SetBeeGOColours.cs](#).

6.55.3.4 objects

`GameObject [] BeeGame.Items.SetBeeGOColours.objects`

Definition at line 13 of file [SetBeeGOColours.cs](#).

6.55.3.5 tiara

`GameObject BeeGame.Items.SetBeeGOColours.tiara`

Definition at line 15 of file [SetBeeGOColours.cs](#).

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/GameObjectStuff/[SetBeeGOColours.cs](#)

6.56 BeeGame.Terrain.LandGeneration.Noise.SimplexNoise Class Reference

Implementation of the Perlin simplex noise, an improved Perlin noise algorithm. Based loosely on SimplexNoise1234 by Stefan Gustavson <http://staffwww.itn.liu.se/~stegu/aqsis/aqsis-newnoise/>

Static Public Member Functions

- static float [Generate](#) (float x)
1D simplex noise
- static float [Generate](#) (float x, float y)
2D simplex noise
- static float [Generate](#) (float x, float y, float z)

Static Public Attributes

- static byte [] [perm](#)

Static Private Member Functions

- static int [FastFloor](#) (float x)
- static int [Mod](#) (int x, int m)
- static float [grad](#) (int hash, float x)
- static float [grad](#) (int hash, float x, float y)
- static float [grad](#) (int hash, float x, float y, float z)
- static float [grad](#) (int hash, float x, float y, float z, float t)

6.56.1 Detailed Description

Implementation of the Perlin simplex noise, an improved Perlin noise algorithm. Based loosely on SimplexNoise1234 by Stefan Gustavson <http://staffwww.itn.liu.se/~stegu/aqsis/aqsis-newnoise/>

Definition at line 37 of file [SimplexNoise.cs](#).

6.56.2 Member Function Documentation

6.56.2.1 FastFloor()

```
static int BeeGame.Terrain.LandGeneration.Noise.SimplexNoise.FastFloor (
    float x )  [static], [private]
```

Definition at line 272 of file [SimplexNoise.cs](#).

```
00273     {
00274         return (x > 0) ? ((int)x) : (((int)x) - 1);
00275     }
```

6.56.2.2 Generate() [1/3]

```
static float BeeGame.Terrain.LandGeneration.Noise.SimplexNoise.Generate (
    float x )  [static]
```

1D simplex noise

Parameters

x	
---	--

Returns

Definition at line 44 of file [SimplexNoise.cs](#).

```
00045     {
00046         int i0 = FastFloor(x);
00047         int i1 = i0 + 1;
00048         float x0 = x - i0;
00049         float x1 = x0 - 1.0f;
00050
00051         float n0, n1;
00052
00053         float t0 = 1.0f - x0 * x0;
00054         t0 *= t0;
00055         n0 = t0 * t0 * grad(perm[i0 & 0xff], x0);
00056
00057         float t1 = 1.0f - x1 * x1;
00058         t1 *= t1;
```

```

00059     n1 = t1 * t1 * grad(perm[i1 & 0xff], x1);
00060     /* The maximum value of this noise is 8*(3/4)^4 = 2.53125
00061     /* A factor of 0.395 scales to fit exactly within [-1,1]
00062     return 0.395f * (n0 + n1);
00063 }

```

6.56.2.3 Generate() [2/3]

```

static float BeeGame.Terrain.LandGeneration.Noise.SimplexNoise.Generate (
    float x,
    float y )  [static]

```

2D simplex noise

Parameters

x	
y	

Returns

Definition at line 71 of file [SimplexNoise.cs](#).

```

00072     {
00073         const float F2 = 0.366025403f; /* F2 = 0.5*(sqrt(3.0)-1.0)
00074         const float G2 = 0.211324865f; /* G2 = (3.0-Math.sqrt(3.0))/6.0
00075
00076         float n0, n1, n2; /* Noise contributions from the three corners
00077
00078         /* Skew the input space to determine which simplex cell we're in
00079         float s = (x + y) * F2; /* Hairy factor for 2D
00080         float xs = x + s;
00081         float ys = y + s;
00082         int i = FastFloor(xs);
00083         int j = FastFloor(ys);
00084
00085         float t = (float)(i + j) * G2;
00086         float X0 = i - t; /* Unskew the cell origin back to (x,y) space
00087         float Y0 = j - t;
00088         float x0 = x - X0; /* The x,y distances from the cell origin
00089         float y0 = y - Y0;
00090
00091         /* For the 2D case, the simplex shape is an equilateral triangle.
00092         /* Determine which simplex we are in.
00093         int i1, j1; /* Offsets for second (middle) corner of simplex in (i,j) coords
00094         if (x0 > y0) { i1 = 1; j1 = 0; } /* lower triangle, XY order: (0,0)->(1,0)->(1,1)
00095         else { i1 = 0; j1 = 1; } /* upper triangle, YX order: (0,0)->(0,1)->(1,1)
00096
00097         /* A step of (1,0) in (i,j) means a step of (1-c,-c) in (x,y), and
00098         /* a step of (0,1) in (i,j) means a step of (-c,1-c) in (x,y), where
00099         /* c = (3-sqrt(3))/6
00100
00101         float x1 = x0 - i1 + G2; /* Offsets for middle corner in (x,y) unskewed coords
00102         float y1 = y0 - j1 + G2;
00103         float x2 = x0 - 1.0f + 2.0f * G2; /* Offsets for last corner in (x,y) unskewed coords
00104         float y2 = y0 - 1.0f + 2.0f * G2;
00105
00106         /* Wrap the integer indices at 256, to avoid indexing perm[] out of bounds
00107         int ii = i % 256;
00108         int jj = j % 256;
00109
00110         /* Calculate the contribution from the three corners
00111         float t0 = 0.5f - x0 * x0 - y0 * y0;
00112         if (t0 < 0.0f) n0 = 0.0f;
00113         else

```

```

00114     {
00115         t0 *= t0;
00116         n0 = t0 * t0 * grad(perm[ii + perm[jj]], x0, y0);
00117     }
00118
00119     float t1 = 0.5f - x1 * x1 - y1 * y1;
00120     if (t1 < 0.0f) n1 = 0.0f;
00121     else
00122     {
00123         t1 *= t1;
00124         n1 = t1 * t1 * grad(perm[ii + ii + perm[jj + jj]], x1, y1);
00125     }
00126
00127     float t2 = 0.5f - x2 * x2 - y2 * y2;
00128     if (t2 < 0.0f) n2 = 0.0f;
00129     else
00130     {
00131         t2 *= t2;
00132         n2 = t2 * t2 * grad(perm[ii + 1 + perm[jj + 1]], x2, y2);
00133     }
00134
00135     /* Add contributions from each corner to get the final noise value.
00136     /* The result is scaled to return values in the interval [-1,1].
00137     return 40.0f * (n0 + n1 + n2); /* TODO: The scale factor is preliminary!
00138 }
```

6.56.2.4 Generate() [3/3]

```
static float BeeGame.Terrain.LandGeneration.Noise.SimplexNoise.Generate (
    float x,
    float y,
    float z) [static]
```

Definition at line 141 of file [SimplexNoise.cs](#).

```

00142     {
00143         /* Simple skewing factors for the 3D case
00144         const float F3 = 0.333333333f;
00145         const float G3 = 0.166666667f;
00146
00147         float n0, n1, n2, n3; /* Noise contributions from the four corners
00148
00149         /* Skew the input space to determine which simplex cell we're in
00150         float s = (x + y + z) * F3; /* Very nice and simple skew factor for 3D
00151         float xs = x + s;
00152         float ys = y + s;
00153         float zs = z + s;
00154         int i = FastFloor(xs);
00155         int j = FastFloor(ys);
00156         int k = FastFloor(zs);
00157
00158         float t = (float)(i + j + k) * G3;
00159         float X0 = i - t; /* Unskew the cell origin back to (x,y,z) space
00160         float Y0 = j - t;
00161         float Z0 = k - t;
00162         float x0 = x - X0; /* The x,y,z distances from the cell origin
00163         float y0 = y - Y0;
00164         float z0 = z - Z0;
00165
00166         /* For the 3D case, the simplex shape is a slightly irregular tetrahedron.
00167         /* Determine which simplex we are in.
00168         int i1, j1, k1; /* Offsets for second corner of simplex in (i,j,k) coords
00169         int i2, j2, k2; /* Offsets for third corner of simplex in (i,j,k) coords
00170
00171         /* This code would benefit from a backport from the GLSL version! */
00172         if (x0 >= y0)
00173         {
00174             if (y0 >= z0)
00175                 { i1 = 1; j1 = 0; k1 = 0; i2 = 1; j2 = 1; k2 = 0; } /* X Y Z order
00176                 else if (x0 >= z0) { i1 = 1; j1 = 0; k1 = 0; i2 = 1; j2 = 0; k2 = 1; } /* X Z Y order
00177                 else { i1 = 0; j1 = 0; k1 = 1; i2 = 1; j2 = 0; k2 = 1; } /* Z X Y order
00178             }
00179         else
00180             { /* x0 < y0
00181                 if (y0 < z0) { i1 = 0; j1 = 0; k1 = 1; i2 = 0; j2 = 1; k2 = 1; } /* Z Y X order
00182                 else if (x0 < z0) { i1 = 0; j1 = 1; k1 = 0; i2 = 0; j2 = 1; k2 = 1; } /* Y Z X order
00183             }
```

```

00183     else { i1 = 0; j1 = 1; k1 = 0; i2 = 1; j2 = 1; k2 = 0; } /* Y X Z order
00184 }
00185
00186     /* A step of (1,0,0) in (i,j,k) means a step of (1-c,-c,-c) in (x,y,z),
00187     /* a step of (0,1,0) in (i,j,k) means a step of (-c,1-c,-c) in (x,y,z), and
00188     /* a step of (0,0,1) in (i,j,k) means a step of (-c,-c,1-c) in (x,y,z), where
00189     /* c = 1/6.
00190
00191     float x1 = x0 - i1 + G3; /* Offsets for second corner in (x,y,z) coords
00192     float y1 = y0 - j1 + G3;
00193     float z1 = z0 - k1 + G3;
00194     float x2 = x0 - i2 + 2.0f * G3; /* Offsets for third corner in (x,y,z) coords
00195     float y2 = y0 - j2 + 2.0f * G3;
00196     float z2 = z0 - k2 + 2.0f * G3;
00197     float x3 = x0 - 1.0f + 3.0f * G3; /* Offsets for last corner in (x,y,z) coords
00198     float y3 = y0 - 1.0f + 3.0f * G3;
00199     float z3 = z0 - 1.0f + 3.0f * G3;
00200
00201     /* Wrap the integer indices at 256, to avoid indexing perm[] out of bounds
00202     int ii = Mod(i, 256);
00203     int jj = Mod(j, 256);
00204     int kk = Mod(k, 256);
00205
00206     /* Calculate the contribution from the four corners
00207     float t0 = 0.6f - x0 * x0 - y0 * y0 - z0 * z0;
00208     if (t0 < 0.0f) n0 = 0.0f;
00209     else
00210     {
00211         t0 *= t0;
00212         n0 = t0 * t0 * grad(perm[ii + perm[jj + perm[kk]]], x0, y0, z0);
00213     }
00214
00215     float t1 = 0.6f - x1 * x1 - y1 * y1 - z1 * z1;
00216     if (t1 < 0.0f) n1 = 0.0f;
00217     else
00218     {
00219         t1 *= t1;
00220         n1 = t1 * t1 * grad(perm[ii + i1 + perm[jj + j1 +
00221             perm[kk + k1]]], x1, y1, z1);
00222     }
00223
00224     float t2 = 0.6f - x2 * x2 - y2 * y2 - z2 * z2;
00225     if (t2 < 0.0f) n2 = 0.0f;
00226     else
00227     {
00228         t2 *= t2;
00229         n2 = t2 * t2 * grad(perm[ii + i2 + perm[jj + j2 +
00230             perm[kk + k2]]], x2, y2, z2);
00231
00232     float t3 = 0.6f - x3 * x3 - y3 * y3 - z3 * z3;
00233     if (t3 < 0.0f) n3 = 0.0f;
00234     else
00235     {
00236         t3 *= t3;
00237         n3 = t3 * t3 * grad(perm[ii + 1 + perm[jj + 1 + perm[kk + 1]]], x3, y3, z3)
00238     }
00239
00240     /* Add contributions from each corner to get the final noise value.
00241     /* The result is scaled to stay just inside [-1,1]
00242     return 32.0f * (n0 + n1 + n2 + n3); /* TODO: The scale factor is preliminary!
00243 }
```

6.56.2.5 grad() [1/4]

```

static float BeeGame.Terrain.LandGeneration.Noise.SimplexNoise.grad (
    int hash,
    float x ) [static], [private]
```

Definition at line 283 of file [SimplexNoise.cs](#).

```

00284     {
00285         int h = hash & 15;
00286         float grad = 1.0f + (h & 7); /* Gradient value 1.0, 2.0, ..., 8.0
00287         if ((h & 8) != 0) grad = -grad; /* Set a random sign for the gradient
00288         return (grad * x); /* Multiply the gradient with the distance
00289     }
```

6.56.2.6 grad() [2/4]

```
static float BeeGame.Terrain.LandGeneration.Noise.SimplexNoise.grad (
    int hash,
    float x,
    float y ) [static], [private]
```

Definition at line 291 of file [SimplexNoise.cs](#).

```
00292     {
00293         int h = hash & 7;      /* Convert low 3 bits of hash code
00294         float u = h < 4 ? x : y; /* into 8 simple gradient directions,
00295         float v = h < 4 ? y : x; /* and compute the dot product with (x,y).
00296         return ((h & 1) != 0 ? -u : u) + ((h & 2) != 0 ? -2.0f * v : 2.0f * v);
00297     }
```

6.56.2.7 grad() [3/4]

```
static float BeeGame.Terrain.LandGeneration.Noise.SimplexNoise.grad (
    int hash,
    float x,
    float y,
    float z ) [static], [private]
```

Definition at line 299 of file [SimplexNoise.cs](#).

```
00300     {
00301         int h = hash & 15;      /* Convert low 4 bits of hash code into 12 simple
00302         float u = h < 8 ? x : y; /* gradient directions, and compute dot product.
00303         float v = h < 4 ? y : h == 12 || h == 14 ? x : z; /* Fix repeats at h = 12 to 15
00304         return ((h & 1) != 0 ? -u : u) + ((h & 2) != 0 ? -v : v);
00305     }
```

6.56.2.8 grad() [4/4]

```
static float BeeGame.Terrain.LandGeneration.Noise.SimplexNoise.grad (
    int hash,
    float x,
    float y,
    float z,
    float t ) [static], [private]
```

Definition at line 307 of file [SimplexNoise.cs](#).

```
00308     {
00309         int h = hash & 31;      /* Convert low 5 bits of hash code into 32 simple
00310         float u = h < 24 ? x : y; /* gradient directions, and compute dot product.
00311         float v = h < 16 ? y : z;
00312         float w = h < 8 ? z : t;
00313         return ((h & 1) != 0 ? -u : u) + ((h & 2) != 0 ? -v : v) + ((h & 4) != 0 ? -w : w);
00314     }
```

6.56.2.9 Mod()

```
static int BeeGame.Terrain.LandGeneration.Noise.SimplexNoise.Mod (
    int x,
    int m ) [static], [private]
```

Definition at line 277 of file [SimplexNoise.cs](#).

```
00278     {
00279         int a = x % m;
00280         return a < 0 ? a + m : a;
00281     }
```

6.56.3 Member Data Documentation

6.56.3.1 perm

```
byte [] BeeGame.Terrain.LandGeneration.Noise.SimplexNoise.perm [static]
```

Initial value:

```
= new byte[512] { 151,160,137,91,90,15,
 131,13,201,95,96,53,194,233,7,225,140,36,103,30,69,142,8,99,37,240,21,10,23,
 190, 6,148,247,120,234,75,0,26,197,62,94,252,219,203,117,35,11,32,57,177,33,
 88,237,149,56,87,174,20,125,136,171,168, 68,175,74,165,71,134,139,48,27,166,
 77,146,158,231,83,111,229,122,60,211,133,230,220,105,92,41,55,46,245,40,244,
 102,143,54, 65,25,63,161, 1,216,80,73,209,76,132,187,208, 89,18,169,200,196,
 135,130,116,188,159,86,164,100,109,198,173,186, 3,64,52,217,226,250,124,123,
 5,202,38,147,118,126,255,82,85,212,207,206,59,227,47,16,58,17,182,189,28,42,
 223,183,170,213,119,248,152, 2,44,154,163, 70,221,153,101,155,167, 43,172,9,
 129,22,39,253, 19,98,108,110,79,113,224,232,178,185, 112,104,218,246,97,228,
 251,34,242,193,238,210,144,12,191,179,162,241, 81,51,145,235,249,14,239,107,
 49,192,214, 31,181,199,106,157,184, 84,204,176,115,121,50,45,127, 4,150,254,
 138,236,205,93,222,114,67,29,24,72,243,141,128,195,78,66,215,61,156,180,
 151,160,137,91,90,15,
 131,13,201,95,96,53,194,233,7,225,140,36,103,30,69,142,8,99,37,240,21,10,23,
 190, 6,148,247,120,234,75,0,26,197,62,94,252,219,203,117,35,11,32,57,177,33,
 88,237,149,56,87,174,20,125,136,171,168, 68,175,74,165,71,134,139,48,27,166,
 77,146,158,231,83,111,229,122,60,211,133,230,220,105,92,41,55,46,245,40,244,
 102,143,54, 65,25,63,161, 1,216,80,73,209,76,132,187,208, 89,18,169,200,196,
 135,130,116,188,159,86,164,100,109,198,173,186, 3,64,52,217,226,250,124,123,
 5,202,38,147,118,126,255,82,85,212,207,206,59,227,47,16,58,17,182,189,28,42,
 223,183,170,213,119,248,152, 2,44,154,163, 70,221,153,101,155,167, 43,172,9,
 129,22,39,253, 19,98,108,110,79,113,224,232,178,185, 112,104,218,246,97,228,
 251,34,242,193,238,210,144,12,191,179,162,241, 81,51,145,235,249,14,239,107,
 49,192,214, 31,181,199,106,157,184, 84,204,176,115,121,50,45,127, 4,150,254,
 138,236,205,93,222,114,67,29,24,72,243,141,128,195,78,66,215,61,156,180
}
```

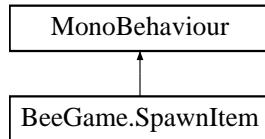
Definition at line 244 of file [SimplexNoise.cs](#).

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/LandGeneration/Noise/[SimplexNoise.cs](#)

6.57 BeeGame.SpawnItem Class Reference

Inheritance diagram for BeeGame.SpawnItem:



Private Member Functions

- void [Start \(\)](#)
- void [OnDrawGizmos \(\)](#)

6.57.1 Detailed Description

Definition at line [12](#) of file [SpawnItem.cs](#).

6.57.2 Member Function Documentation

6.57.2.1 OnDrawGizmos()

```
void BeeGame.SpawnItem.OnDrawGizmos ( ) [private]
```

Definition at line [56](#) of file [SpawnItem.cs](#).

```
00057      {  
00058          //Gizmos.color = Color.green;  
00059          //Gizmos.DrawSphere(transform.position, 0.5f);  
00060      }
```

6.57.2.2 Start()

```
void BeeGame.SpawnItem.Start ( ) [private]
```

Definition at line 14 of file [SpawnItem.cs](#).

```
00015      {
00016          //GameObject go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as
00017          //GameObject, transform.position, Quaternion.identity) as GameObject;
00018          //go.GetComponent<ItemGameObject>().item = new Bee(BeeType.DRONE, new NormalBee() { pSpecies =
00019          BeeSpecies.FOREST, sSpecies = BeeSpecies.FOREST });
00020
00021          //go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
00022          transform.position, Quaternion.identity) as GameObject;
00023          //go.GetComponent<ItemGameObject>().item = new Bee(BeeType.PRINCESS, new NormalBee() { pSpecies =
00024          BeeSpecies.FOREST, sSpecies = BeeSpecies.FOREST });
00025
00026          //go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
00027          transform.position, Quaternion.identity) as GameObject;
00028          //go.GetComponent<ItemGameObject>().item = new Bee(BeeType.QUEEN, new QueenBee() { queen = new
00029          NormalBee() { pSpecies = BeeSpecies.FOREST, sSpecies = BeeSpecies.FOREST }, drone = new NormalBee() {
00030          pSpecies = BeeSpecies.FOREST, sSpecies = BeeSpecies.FOREST } });
00031
00032          //go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
00033          transform.position, Quaternion.identity) as GameObject;
00034          //go.GetComponent<ItemGameObject>().item = new Bee(BeeType.DRONE, new NormalBee() { pSpecies =
00035          BeeSpecies.COMMON, sSpecies = BeeSpecies.COMMON });
00036
00037          //go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
00038          transform.position, Quaternion.identity) as GameObject;
00039          //go.GetComponent<ItemGameObject>().item = new HoneyComb(HoneyCombType.ICEY);
00040
00041          //go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
00042          transform.position, Quaternion.identity) as GameObject;
00043          //go.GetComponent<ItemGameObject>().item = new HoneyComb(HoneyCombType.HONEY);
00044
00045          //go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
00046          transform.position, Quaternion.identity) as GameObject;
00047          //go.GetComponent<ItemGameObject>().item = new Chest();
00048
00049          GameObject go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as
00050          GameObject, transform.position, Quaternion.identity) as GameObject;
00051          go.GetComponent<ItemGameObject>().item = new QuestBook();
00052
00053          go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
00054          transform.position, Quaternion.identity) as GameObject;
00055          go.GetComponent<ItemGameObject>().item = new BeeAlyzer();
00056      }
```

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/[SpawnItem.cs](#)

6.58 BeeGame.Core.Dictionaries.SpriteDictionary Class Reference

All of the sprites available to the game

Static Public Member Functions

- static Sprite [GetSprite](#) (string spriteName)
Get a sprite of the given name
- static void [LoadSprites](#) ()
Loads the sprites into the dictionary

Static Private Attributes

- static Dictionary< string, Sprite > [itemSpriteDictionary](#) = new Dictionary<string, Sprite>()
All of the sprites available to spawn in

6.58.1 Detailed Description

All of the sprites available to the game

Definition at line 9 of file [SpriteDictionary.cs](#).

6.58.2 Member Function Documentation

6.58.2.1 [GetSprite\(\)](#)

```
static Sprite BeeGame.Core.Dictionaries.SpriteDictionary.GetSprite (
    string spriteName ) [static]
```

Get a sprite of the given name

Parameters

<i>spriteName</i>	Name of sprite to get
-------------------	-----------------------

Returns

A sprite of the given name, null if no sprite of that name exists

Definition at line 21 of file [SpriteDictionary.cs](#).

```
00022     {
00023         itemSpriteDictionary.TryGetValue(spriteName, out Sprite sprite);
00024
00025         if (sprite == null)
00026             return new Sprite();
00027
00028         return sprite;
00029     }
```

6.58.2.2 LoadSprites()

```
static void BeeGame.Core.Dictionaries.SpriteDictionary.LoadSprites () [static]
```

Loads the sprites into the dictionary

Definition at line 34 of file [SpriteDictionary.cs](#).

```
00035      {
00036          itemSpriteDictionary = Resources.Resources.GetSprites();
00037      }
```

6.58.3 Member Data Documentation

6.58.3.1 itemSpriteDictionary

```
Dictionary<string, Sprite> BeeGame.Core.Dictionaries.SpriteDictionary.itemSpriteDictionary =
new Dictionary<string, Sprite>() [static], [private]
```

All of the sprites available to spawn in

Definition at line 14 of file [SpriteDictionary.cs](#).

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionaries/[SpriteDictionary.cs](#)

6.59 BeeGame.Terrain.LandGeneration.Terrain Class Reference

Should use as an interface between the rest of the game and the terrain

Static Public Member Functions

- static [ChunkWorldPos GetBlockPos \(THVector3 pos\)](#)
Gets a block position from a THVector3
- static [THVector3 GetBlockPos \(RaycastHit hit\)](#)
Returns the position of the block hit as a THVector3
- static [ChunkWorldPos GetBlockPosFromRayCast \(RaycastHit hit\)](#)
GetBlockPos(THVector3) does the same thing but returns a ChunkWorldPos
- static float [Round \(float pos, float norm, bool adjacent=false\)](#)
Rounds the given pos to the correct position
- static [ChunkWorldPos GetBlockPos \(RaycastHit hit, bool adjacent=false\)](#)
Gets a Chunks world position
- static [Block GetBlock \(RaycastHit hit, bool adjacent=false\)](#)
Get a Block at the given position
- static [Block GetBlock \(THVector3 pos\)](#)
- static bool [BlockInPosition \(THVector3 pos, Chunk chunk\)](#)
- static [Chunk GetChunk \(THVector3 vec3\)](#)
- static bool [SetBlock \(RaycastHit hit, Block block, bool adjacent=false\)](#)
Sets the Block at the given point the given Block

Static Public Attributes

- static [World world](#)

Static Private Member Functions

- static float [RoundXZ](#) (float pos, float normal)
Used to round the X/Z values when getting a block
- static float [RoundY](#) (float pos, float normal)
Round the Y value of the given coord

6.59.1 Detailed Description

Should use as an interface between the rest of the game and the terrain

Definition at line [12](#) of file [Terrain.cs](#).

6.59.2 Member Function Documentation**6.59.2.1 BlockInPosition()**

```
static bool BeeGame.Terrain.LandGeneration.Terrain.BlockInPosition (
    THVector3 pos,
    Chunk chunk )  [static]
```

Definition at line [247](#) of file [Terrain.cs](#).

```
00248     {
00249         if (chunk == null)
00250             return false;
00251
00252         if (chunk.GetBlock((int)pos.x, (int)pos.y, (int)pos.z) != new
00253             Air())
00254             return true;
00255         return false;
00256     }
```

6.59.2.2 GetBlock() [1/2]

```
static Block BeeGame.Terrain.LandGeneration.Terrain.GetBlock (
    RaycastHit hit,
    bool adjacent = false )  [static]
```

Get a Block at the given position

Parameters

<i>hit</i>	Where to get the block from
<i>adjacent</i>	Should the adjacent Block be returned

Returns

Block at `hit.point`, Null if no block was found

Definition at line 221 of file [Terrain.cs](#).

```
00222      {
00223          /* checks that a chunk was hit and if it wasnt return early
00224          Chunk chunk = hit.collider.GetComponent<Chunk>();
00225
00226          if (chunk == null)
00227              return null;
00228
00229          /* allignes the hit to the block grid and returns the block
00230          ChunkWorldPos pos = GetBlockPos(hit, adjacent);
00231
00232          return chunk.world.GetBlock(pos.x, pos.y, pos.z);
00233      }
```

6.59.2.3 GetBlock() [2/2]

```
static Block BeeGame.Terrain.LandGeneration.Terrain.GetBlock (
    THVector3 pos ) [static]
```

Definition at line 235 of file [Terrain.cs](#).

```
00236      {
00237          Chunk chunk = GetChunk(pos);
00238
00239          if (chunk == null)
00240              return new Air();
00241
00242          chunk.world.GetBlock((int)pos.x, (int)pos.y, (int)pos.z);
00243
00244          return new Block();
00245      }
```

6.59.2.4 GetBlockPos() [1/3]

```
static ChunkWorldPos BeeGame.Terrain.LandGeneration.Terrain.GetBlockPos (
    THVector3 pos ) [static]
```

Gets a block position from a THVector3

Parameters

<code>pos</code>	Position of the block as a THVector3
------------------	--------------------------------------

Returns

`ChunkWorldPos` of the Block

Definition at line 22 of file [Terrain.cs](#).

```

00023      {
00024          return new ChunkWorldPos()
00025      {
00026          x = Mathf.RoundToInt(pos.x),
00027          y = Mathf.RoundToInt(pos.y),
00028          z = Mathf.RoundToInt(pos.z)
00029      };
00030  }

```

6.59.2.5 GetBlockPos() [2/3]

```
static THVector3 BeeGame.Terrain.LandGeneration.Terrain.GetBlockPos (
    RaycastHit hit) [static]
```

Returns the positon of the block hit as a THVector3

Parameters

<i>hit</i>	RaycastHit
<i>adjacent</i>	Do you want the face adjecent to the block hit

Returns

THVector3 of the block you hit in world cordinates

Definition at line 38 of file [Terrain.cs](#).

```

00039      {
00040          THVector3 vec3 = new THVector3()
00041      {
00042          x = RoundXZ(hit.point.x, hit.normal.x),
00043          y = RoundY(hit.point.y, hit.normal.y),
00044          z = RoundXZ(hit.point.z, hit.normal.z)
00045      };
00046      return (vec3);
00047  }

```

6.59.2.6 GetBlockPos() [3/3]

```
static ChunkWorldPos BeeGame.Terrain.LandGeneration.Terrain.GetBlockPos (
    RaycastHit hit,
    bool adjacent = false) [static]
```

Gets a Chunks world positon

Parameters

<i>hit</i>	Where the raycast hit
<i>adjacent</i>	Should the adjacent Chunk position be returned?

Returns[ChunkWorldPos](#) of the Chunk**Returns**Definition at line 204 of file [Terrain.cs](#).

```

00205      {
00206          return GetBlockPos(new THVector3()
00207          {
00208              /* rounds the hit to the correct position
00209              x = Round(hit.point.x, hit.normal.x, adjacent),
00210              y = Round(hit.point.y, hit.normal.y, adjacent),
00211              z = Round(hit.point.z, hit.normal.z, adjacent)
00212          });
00213      }

```

6.59.2.7 GetBlockPosFromRayCast()

```
static ChunkWorldPos BeeGame.Terrain.LandGeneration.Terrain.GetBlockPosFromRayCast (
    RaycastHit hit) [static]
```

[GetBlockPos\(THVector3\)](#) does the same thing but returns a [ChunkWorldPos](#)**Parameters**

<i>hit</i>	
------------	--

ReturnsDefinition at line 54 of file [Terrain.cs](#).

```

00055      {
00056          return new ChunkWorldPos((int)RoundXZ(hit.point.x, hit.normal.x), (int)
00057          RoundY(hit.point.y, hit.normal.y), (int)RoundXZ(hit.point.z, hit.normal.z));
00057      }

```

6.59.2.8 GetChunk()

```
static Chunk BeeGame.Terrain.LandGeneration.Terrain.GetChunk (
    THVector3 vec3) [static]
```

Definition at line 259 of file [Terrain.cs](#).

```

00260      {
00261          return world.GetChunk((int)vec3.x, (int)vec3.y, (int)vec3.
00262          z);
00262      }

```

6.59.2.9 Round()

```
static float BeeGame.Terrain.LandGeneration.Terrain.Round (
    float pos,
    float norm,
    bool adjacent = false ) [static]
```

Rounds the given pos to the correct position

Parameters

<i>pos</i>	Position that needs to be rounded
<i>norm</i>	Normal for the face
<i>adjacent</i>	Should the adjacent block be received

Returns

rounded value of *pos* as a float

Check how this performs. Possibly change all uses of this to [RoundXZ\(float, float\)](#) and [RoundY\(float, float\)](#)

Definition at line 179 of file [Terrain.cs](#).

```
00180      {
00181          if(pos - (int)pos == 0.5f || pos - (int)pos == -0.5f)
00182          {
00183              if(adjacent)
00184              {
00185                  pos += (norm / 2);
00186              }
00187              else
00188              {
00189                  pos -= (norm / 2);
00190              }
00191          }
00192
00193         return pos;
00194     }
```

6.59.2.10 RoundXZ()

```
static float BeeGame.Terrain.LandGeneration.Terrain.RoundXZ (
    float pos,
    float normal ) [static], [private]
```

Used to round the X/Z values when getting a block

Parameters

<i>pos</i>	X/Y pos
<i>normal</i>	X/Y normal

Returns

rounded pos

Do I realy need to do all this?

Definition at line 68 of file [Terrain.cs](#).

```

00069      {
00070          /* if we are looking at + x/z vlaues
00071          if (pos > 0)
00072          {
00073              if (normal > 0)
00074              {
00075                  pos = (int)pos;
00076                  return pos;
00077              }
00078              else if (normal < 0)
00079              {
00080                  pos = (int)pos;
00081                  return pos - 1;
00082              }
00083          else
00084          {
00085              if ((pos - (int)pos) > 0.5)
00086              {
00087                  return (int)pos + 1;
00088              }
00089              return (int)pos;
00090          }
00091      }
00092      /* if we are looking at - x/z values
00093      else
00094      {
00095          /* if poitive normal
00096          if (normal > 0)
00097          {
00098              pos = (int)pos;
00099              return pos - 1;
00100          }
00101          /* if negative nomrmal
00102          if (normal < 0)
00103          {
00104              pos = (int)pos;
00105              return pos;
00106          }
00107          /* if their is no normal
00108          /* if pos is greater than 0.5 we are in the next block so go to it
00109          if ((-pos - (int)-pos) > 0.5)
00110          {
00111              return (int)pos - 1;
00112          }
00113          return (int)pos;
00114      }
00115  }
00116 }
```

6.59.2.11 RoundY()

```
static float BeeGame.Terrain.LandGeneration.Terrain.RoundY (
    float pos,
    float normal ) [static], [private]
```

Round the Y value of the given coord

Parameters

<i>pos</i>	Y pos
<i>normal</i>	Y normal

Returns

pos rounded to 1 DP

Do I have to do this? or is there an easier way to do this

Definition at line 129 of file [Terrain.cs](#).

```

00130      {
00131          pos = (float)Math.Round(pos, 1);
00132          if (pos >= 0)
00133          {
00134              if(normal > 0)
00135              {
00136                  if((int)pos % 2 == 0)
00137                      return Mathf.RoundToInt((float)Math.Round(pos, 1));
00138                  return Mathf.RoundToInt((float)Math.Round(pos, 1)) - normal;
00139              }
00140          }
00141          if((int)pos % 2 == 0)
00142              return Mathf.RoundToInt((float)Math.Round(pos, 1)) - normal;
00143          return Mathf.RoundToInt((float)Math.Round(pos, 1));
00144      }
00145      if(pos <= 0)
00146      {
00147          if (normal > 0)
00148          {
00149              if ((int)pos % 2 == 0)
00150                  /* the Math.Round removes strange rounding errors shown with Mathf.Round eg
sometimes 0.5 would round to 0 not 1
00151                  return Mathf.RoundToInt((float)Math.Round(pos, 1)) - normal;
00152              return Mathf.RoundToInt((float)Math.Round(pos, 1)); // - normal;
00153          }
00154          if ((int)pos % 2 == 0)
00155              return Mathf.RoundToInt((float)Math.Round(pos, 1));
00156          return Mathf.RoundToInt((float)Math.Round(pos, 1)) - normal;
00157      }
00158      if ((int)pos % 2 == 0)
00159          return Mathf.RoundToInt((float)Math.Round(pos, 1));
00160      return Mathf.RoundToInt((float)Math.Round(pos, 1));
00161      return Mathf.RoundToInt((float)Math.Round(pos, 1)) - normal;
00162  }
00163  }
00164  }
00165  return Mathf.RoundToInt((float)Math.Round(pos, 1));
00166 }
```

6.59.2.12 SetBlock()

```
static bool BeeGame.Terrain.LandGeneration.Terrain.SetBlock (
    RaycastHit hit,
    Block block,
    bool adjacent = false ) [static]
```

Sets the Block at the given point the given Block

Parameters

<i>hit</i>	Where the block should be set
<i>block</i>	Block to be set
<i>adjacent</i>	Should the adjacent Block be set

Returns

true if block was set

Definition at line 272 of file [Terrain.cs](#).

```

00273         {
00274             /* checks that a chnk was hit
00275             Chunk chunk = hit.collider.GetComponent<Chunk>();
00276
00277             if (chunk == null)
00278                 return false;
00279
00280             /* alligns the hit to the block grid
00281             ChunkWorldPos pos = GetBlockPosFromRayCast(hit);
00282
00283             /* checks that the block trying to be replaced can be replaced eg bedrock cannot be replaced
00284             if (GetBlock(hit, adjacent).breakable)
00285             {
00286                 /* sets the position of the block and saves the chunk
00287                 chunk.world.SetBlock(pos.x, pos.y, pos.z, block);
00288                 Serialization.Serialization.SaveChunk(chunk);
00289             }
00290
00291             return true;
00292         }

```

6.59.3 Member Data Documentation**6.59.3.1 world**

[World](#) BeeGame.Terrain.LandGeneration.Terrain.world [static]

Definition at line 14 of file [Terrain.cs](#).

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/LandGeneration/[Terrain.cs](#)

6.60 BeeGame.Terrain.LandGeneration.TerrainGeneration Class Reference

Generates the terrain for the game

Public Member Functions

- [Chunk ChunkGen \(Chunk chunk\)](#)
Generates a Chunk in a new thread
- [void ChunkGenThread \(Chunk chunk, out Chunk outChunk\)](#)
Generates a new Chunk
- [Chunk GenChunkColum \(Chunk chunk, int x, int z\)](#)
Generates a colum of the Chunk

Static Public Member Functions

- static int [GetNoise](#) (int x, int y, int z, float scale, int max)
Get a noise value
- static void [SetBlock](#) (int x, int y, int z, [Blocks.Block](#) block, [Chunk](#) chunk, bool replacesBlocks=false)
Sets a Block in the position

Private Member Functions

- void [CreateTree](#) (int x, int y, int z, [Chunk](#) chunk)
Makes a tree

Private Attributes

- float [stoneBaseHeight](#) = -24
Base height of stone
- float [stoneBaseNoise](#) = 0.05f
Base noise of stone
- float [stoneBaseNoiseHeight](#) = 4
Base noise height for stone
- float [stoneMountainHeight](#) = 48
Base height for a mountain
- float [stoneMountainFrequency](#) = 0.008f
Frequency of mountains (larger value = more choppy terrain)
- float [stoneMinHeight](#) = -12
Minimum height for stone
- float [dirtBaseHeight](#) = 1
Where does dirt start
- float [dirtNoise](#) = 0.04f
How much of the surface is dirt
- float [dirtNoiseHeight](#) = 3
How tall dirt can be
- float [treeFrequency](#) = 0.2f
Frequency of trees
- int [treeDensity](#) = 3
Density of trees
- float [caveFrequency](#) = 0.025f
How often do caves happen
- int [caveSize](#) = 8
Threshold for making a cave

6.60.1 Detailed Description

Generates the terrain for the game

Definition at line 13 of file [TerrainGeneration.cs](#).

6.60.2 Member Function Documentation

6.60.2.1 ChunkGen()

```
Chunk BeeGame.Terrain.LandGeneration.TerrainGeneration.ChunkGen (
    Chunk chunk )
```

Generates a Chunk in a new thread

Parameters

<i>chunk</i>	Chunk to populate with Blocks
--------------	-------------------------------

Returns

Chunk with Blocks generated

Definition at line 79 of file [TerrainGeneration.cs](#).

```

00080      {
00081          Chunk outChunk = chunk;
00082          lock (chunk)
00083          {
00084              Thread thread = new Thread(() => ChunkGenThread(chunk, out outChunk)) { Name
= $"Generate Chunk Thread @ {chunk.chunkWorldPos}" };
00085              thread.Start();
00086              return outChunk;
00087          }
00088      }
00089  }
```

6.60.2.2 ChunkGenThread()

```
void BeeGame.Terrain.LandGeneration.TerrainGeneration.ChunkGenThread (
    Chunk chunk,
    out Chunk outChunk )
```

Generates a new Chunk

Parameters

<i>chunk</i>	Chunk to be generated
<i>outChunk</i>	Generated Chunk to return

Definition at line 96 of file [TerrainGeneration.cs](#).

```

00097      {
00098          /* for each x and z position in teh chunk
00099          for (int x = chunk.chunkWorldPos.x-3; x < chunk.
chunkWorldPos.x + Chunk.chunkSize + 3; x++)
00100          {
00101              for (int z = chunk.chunkWorldPos.z-3; z < chunk.
chunkWorldPos.z + Chunk.chunkSize + 3; z++)
00102              {
00103                  chunk = GenChunkColum(chunk, x, z);
00104              }
00105          }
00106
00107          chunk.SetBlocksUnmodified();
00108          outChunk = chunk;
00109      }
```

6.60.2.3 CreateTree()

```
void BeeGame.Terrain.LandGeneration.TerrainGeneration.CreateTree (
    int x,
    int y,
    int z,
    Chunk chunk ) [private]
```

Makes a tree

Parameters

<i>x</i>	X pos of the trunk
<i>y</i>	Y pos of the trunk
<i>z</i>	Z pos of the trunk
<i>chunk</i>	Chunk to make the tree in

Trees will always look the same, possibly add to leafs can have different shapes

Definition at line 210 of file [TerrainGeneration.cs](#).

```
00211      {
00212          /* makes the leaves of teh tree
00213          for (int xi = -2; xi <= 2; xi++)
00214          {
00215              for (int yi = 4; yi <= 8; yi++)
00216              {
00217                  for (int zi = -2; zi <= 2; zi++)
00218                  {
00219                      SetBlock(xi + x, yi + y, zi + z, new Blocks.Leaves(), chunk, true);
00220                  }
00221              }
00222          }
00223
00224          /* makes the trunk of the tree
00225          for (int i = 0; i < 6; i++)
00226          {
00227              SetBlock(x, y + i, z, new Blocks.Wood(), chunk, true);
00228          }
00229      }
```

6.60.2.4 GenChunkColum()

```
Chunk BeeGame.Terrain.LandGeneration.TerrainGeneration.GenChunkColum (
    Chunk chunk,
    int x,
    int z )
```

Generates a colum of the Chunk

Parameters

<i>chunk</i>	Chunk to generate a colum for
<i>x</i>	X pos to make the colum
<i>z</i>	Z pos to make the colum

Returns

Chunk with a new colum ob blocks generated

Definition at line 118 of file [TerrainGeneration.cs](#).

```

00119      {
00120          /* the height of the mountain
00121          int stoneHeight = Mathf.FloorToInt(stoneBaseHeight);
00122          stoneHeight += GetNoise(-x, 0, z, stoneMountainFrequency, Mathf.
FloorToInt(stoneMountainHeight));
00123
00124          /* if the colum is currently to low make it not so low
00125          if (stoneHeight < stoneMinHeight)
00126              stoneHeight = Mathf.FloorToInt(stoneMinHeight);
00127
00128          /* add the height of normal stone on to the mountain
00129          stoneHeight += GetNoise(x, 0, -z, stoneBaseNoise, Mathf.RoundToInt(
stoneBaseNoiseHeight));
00130
00131          //put dirt on top
00132          int dirtHeight = stoneHeight + Mathf.FloorToInt(dirtBaseHeight);
00133          dirtHeight += GetNoise(x, 100, z, dirtNoise, Mathf.FloorToInt(
dirtNoiseHeight));
00134
00135          /* set the colum to the correct blocks
00136          for (int y = chunk.chunkWorldPos.y - 8; y < chunk.
chunkWorldPos.y + Chunk.chunkSize; y++)
00137          {
00138              int caveChance = GetNoise(x + 40, y + 100, z - 50,
caveFrequency, 200);
00139
00140              /* puts a layer of bedrock at the bottom the the world
00141              if (y <= (chunk.chunkWorldPos.y) && chunk.
chunkWorldPos.y == -16)
00142              {
00143                  SetBlock(x, y, z, new Blocks.Bedrock(), chunk);
00144              }
00145              else if (y <= stoneHeight && caveSize < caveChance)
00146              {
00147                  SetBlock(x, y, z, new Blocks.Block(), chunk);
00148              }
00149              else if (y <= dirtHeight && caveSize < caveChance)
00150              {
00151                  SetBlock(x, y, z, new Blocks.Grass(), chunk);
00152                  if (y == dirtHeight && GetNoise(x, 0, z,
treeFrequency, 100) < treeDensity)
00153                      CreateTree(x, y + 1, z, chunk);
00154              }
00155              else
00156              {
00157                  SetBlock(x, y, z, new Blocks.Air(), chunk);
00158              }
00159          }
00160
00161          return chunk;
00162      }

```

6.60.2.5 GetNoise()

```

static int BeeGame.Terrain.LandGeneration.TerrainGeneration.GetNoise (
    int x,
    int y,
    int z,
    float scale,
    int max ) [static]

```

Get a noise value

Parameters

<i>x</i>	X pos of the noise
<i>y</i>	Y pos of the noise
<i>z</i>	Z pos of the noise
<i>scale</i>	What the step shout bee from the last x, y, z
<i>max</i>	Max value of the noise

Returns

A noise value as an int

Definition at line 173 of file [TerrainGeneration.cs](#).

```
00174         {
00175             return Mathf.FloorToInt((SimplexNoise.Generate(x * scale, y * scale, z *
00176             scale) + 1f) * (max / 2f));
00176         }
```

6.60.2.6 SetBlock()

```
static void BeeGame.Terrain.LandGeneration.TerrainGeneration.SetBlock (
    int x,
    int y,
    int z,
    Blocks.Block block,
    Chunk chunk,
    bool replacesBlocks = false ) [static]
```

Sets a Block in the position

Parameters

<i>x</i>	X pos of the block
<i>y</i>	Y pos of the block
<i>z</i>	Z pos of the block
<i>block</i>	Block to set
<i>chunk</i>	Chunk to set the block in
<i>replacesBlocks</i>	Can it replace blocks

Definition at line 187 of file [TerrainGeneration.cs](#).

```
00188         {
00189             /* corrects the x, y, z pos of the so that the block is placed in the correct position
00190             x -= chunk.chunkWorldPos.x;
00191             y -= chunk.chunkWorldPos.y;
00192             z -= chunk.chunkWorldPos.z;
00193
00194             /* checks that the block is in the chunk and that no block is already their then sets it
00195             if (Chunk.InRange(x) && Chunk.InRange(y) &&
00196                 Chunk.InRange(z))
00196                 if (replacesBlocks || chunk.blocks[x, y, z] == null)
00197                     chunk.SetBlock(x, y, z, block, false);
00198         }
```

6.60.3 Member Data Documentation

6.60.3.1 caveFrequency

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.caveFrequency = 0.025f [private]
```

How often do caves happen

Definition at line [67](#) of file [TerrainGeneration.cs](#).

6.60.3.2 caveSize

```
int BeeGame.Terrain.LandGeneration.TerrainGeneration.caveSize = 8 [private]
```

Threshold for makeing a cave

Definition at line [71](#) of file [TerrainGeneration.cs](#).

6.60.3.3 dirtBaseHeight

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.dirtBaseHeight = 1 [private]
```

Where does dirt start

Definition at line [45](#) of file [TerrainGeneration.cs](#).

6.60.3.4 dirtNoise

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.dirtNoise = 0.04f [private]
```

How much of the surface is dirt

Definition at line [49](#) of file [TerrainGeneration.cs](#).

6.60.3.5 dirtNoiseHeight

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.dirtNoiseHeight = 3 [private]
```

How tall dirt can be

Definition at line [53](#) of file [TerrainGeneration.cs](#).

6.60.3.6 stoneBaseHeight

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.stoneBaseHeight = -24 [private]
```

Base height of stone

Definition at line 19 of file [TerrainGeneration.cs](#).

6.60.3.7 stoneBaseNoise

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.stoneBaseNoise = 0.05f [private]
```

Base noise of stone

Definition at line 23 of file [TerrainGeneration.cs](#).

6.60.3.8 stoneBaseNoiseHeight

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.stoneBaseNoiseHeight = 4 [private]
```

Base noise heigh for stone

Definition at line 27 of file [TerrainGeneration.cs](#).

6.60.3.9 stoneMinHeight

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.stoneMinHeight = -12 [private]
```

Minimun height for stone

Definition at line 40 of file [TerrainGeneration.cs](#).

6.60.3.10 stoneMountainFrequency

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.stoneMountainFrequency = 0.008f [private]
```

Frequency of mountains (larger value = more choppy terrain)

Definition at line 36 of file [TerrainGeneration.cs](#).

6.60.3.11 stoneMountainHeight

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.stoneMountainHeight = 48 [private]
```

Base height for a mountain

Definition at line 32 of file [TerrainGeneration.cs](#).

6.60.3.12 treeDensity

```
int BeeGame.Terrain.LandGeneration.TerrainGeneration.treeDensity = 3 [private]
```

Desity of trees

Definition at line 62 of file [TerrainGeneration.cs](#).

6.60.3.13 treeFrequency

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.treeFrequency = 0.2f [private]
```

Frequency of trees

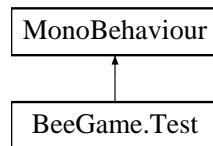
Definition at line 58 of file [TerrainGeneration.cs](#).

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/LandGeneration/TerrainGeneration.cs

6.61 BeeGame.Test Class Reference

Inheritance diagram for BeeGame.Test:



Private Member Functions

- void [Start \(\)](#)
- void [Update \(\)](#)

6.61.1 Detailed Description

Definition at line 9 of file [test.cs](#).

6.61.2 Member Function Documentation

6.61.2.1 Start()

```
void BeeGame.Test.Start () [private]
```

Definition at line 11 of file [test.cs](#).

```
00012      {
00013          CraftingRecipes.AddShapedRecipie(new object[] { " ", " X ",
00014              " ", "X", Dirt.ID }, new Grass());
00015          CraftingRecipes.AddShaplessRecipie(new object[] { new
    Grass(), 1 }, new Dirt());
00015 }
```

6.61.2.2 Update()

```
void BeeGame.Test.Update () [private]
```

Definition at line 17 of file [test.cs](#).

```
00018      {
00019          //var temp = Quest.Quests.ReturnCompleatedQuests();
00020
00021          //if(temp.Count > 0)
00022          //{
00023              //foreach (var item in temp)
00024              //{
00025                  //    Quest.Quests.ClaimQuest(item.Key);
00026              //}
00027          //}
00028      }
```

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/[test.cs](#)

6.62 BeeGame.Core.THInput Class Reference

My implementation of the unity input system. Acts as a buffer layer to the unity system so that the input keys can be changed at runtime

Static Public Member Functions

- static bool [GetButtonDown](#) (string button)
Has the given button been pressed this update
- static bool [GetButton](#) (string button)
Is the given button currently being held down
- static bool [GetButtonUp](#) (string button)
Has the given button been released this update
- static int [GetAxis](#) (string axis)
Gets the axis of a button press

Static Public Attributes

- static bool `isAnotherInventoryOpen`
If another inventory is open true, else false
- static bool `blockInventoryJustClosed`
Was a Block inventory just closed

Static Package Attributes

- static bool `chestOpen`
Stops the player from being able to open the `BeeGame.Inventory.Player_Inventory.PlayerInventory` whilst a block/item `BeeGame.Inventory.Inventory` is open

Static Private Attributes

- static Dictionary< string, object > `inputButtons`
Button identifiers and KeyCode

6.62.1 Detailed Description

My implementation of the unity input system. Acts as a buffer layer to the unity system so that the input keys can be changed at runtime

Definition at line 11 of file [THInput.cs](#).

6.62.2 Member Function Documentation

6.62.2.1 GetAxis()

```
static int BeeGame.Core.THInput.GetAxis (
    string axis ) [static]
```

Gets the axis of a button press

Parameters

<code>axis</code>	Axis to check, Horizontal or Vertical
-------------------	---------------------------------------

Returns

+1 or -1

Definition at line 140 of file [THInput.cs](#).

```

00142         int returnAxis = 0;
00143
00144         if (axis == "Horizontal")
00145         {
00146             if (GetButton("Right"))
00147             {
00148                 returnAxis += 1;
00149             }
00150
00151             if (GetButton("Left"))
00152             {
00153                 returnAxis -= 1;
00154             }
00155         }
00156         else if (axis == "Vertical")
00157         {
00158             if (GetButton("Forward"))
00159             {
00160                 returnAxis += 1;
00161             }
00162
00163             if (GetButton("Backward"))
00164             {
00165                 returnAxis -= 1;
00166             }
00167         }
00168
00169     return returnAxis;
00170 }
```

6.62.2.2 GetButton()

```
static bool BeeGame.Core.THInput.GetButton (
    string button ) [static]
```

Is the given button currently being held down

Parameters

<i>button</i>	The button name e.g. "Forward"
---------------	--------------------------------

Returns

true if the given button is currently being held down

Definition at line 80 of file [THInput.cs](#).

```

00081         {
00082             if (!inputButtons.ContainsKey(button))
00083             {
00084                 throw new InputException($"Key input name not defined: {button}");
00085             }
00086
00087             switch (inputButtons[button])
00088             {
00089                 case KeyCode[] arry:
00090                     /*for each possible key, check if it was pressed and if it was return that it was; if
none of them was pressed return false
00091                     foreach (var item in arry)
00092                     {
00093                         if (Input.GetKey(item))
00094                         {
00095                             return true;
00096                         }
00097                     }
00098
00099                     return false;
00100                 default:
00101                     return Input.GetKey((KeyCode)inputButtons[button]);
00102             }
00103 }
```

6.62.2.3 GetButtonDown()

```
static bool BeeGame.Core.THInput.GetButtonDown (
    string button ) [static]
```

Has the given button been pressed this update

Parameters

<i>button</i>	The button name eg "Inventory"
---------------	--------------------------------

Returns

true if the given button has been pressed this update

Definition at line 50 of file [THInput.cs](#).

```
00051      {
00052          if (!inputButtons.ContainsKey(button))
00053          {
00054              throw new InputException($"Key input name not defined: {button}");
00055          }
00056
00057          switch (inputButtons[button])
00058          {
00059              case KeyCode[] arry:
00060                  /*for each possible key, check if it was pressed and if it was return that it was; if
none of them was pressed, return false
00061                  foreach (var item in arry)
00062                  {
00063                      if (Input.GetKeyDown(item))
00064                      {
00065                          return true;
00066                      }
00067                  }
00068
00069                  return false;
00070          default:
00071              return Input.GetKeyDown((KeyCode)inputButtons[button]);
00072          }
00073      }
```

6.62.2.4 GetButtonUp()

```
static bool BeeGame.Core.THInput.GetButtonUp (
    string button ) [static]
```

Has the given button been released this update

Parameters

<i>button</i>	Button name e.g. "Inventory"
---------------	------------------------------

Returns

true if the button has been released during this update

Definition at line 110 of file [THInput.cs](#).

```

00111     {
00112         if (!inputButtons.ContainsKey(button))
00113         {
00114             throw new InputException($"Key input name not defined: {button}");
00115         }
00116
00117         switch (inputButtons[button])
00118         {
00119             case KeyCode[] arry:
00120                 /*for each possible key, check if it was pressed and if it was return that it was; if
none of them was pressed return false
00121                 foreach (var item in arry)
00122                 {
00123                     if (Input.GetKeyUp(item))
00124                     {
00125                         return true;
00126                     }
00127                 }
00128
00129                 return false;
00130             default:
00131                 return Input.GetKeyUp((KeyCode)inputButtons[button]);
00132         }
00133     }

```

6.62.3 Member Data Documentation

6.62.3.1 blockInventoryJustClosed

```
bool BeeGame.Core.THInput.blockInventoryJustClosed [static]
```

Was a Block inventory just closed

Definition at line 39 of file [THInput.cs](#).

6.62.3.2 chestOpen

```
bool BeeGame.Core.THInput.chestOpen [static], [package]
```

Stops the player from being able to open the [BeeGame.Inventory.Player_Inventory.PlayerInventory](#) whilst a block/item [BeeGame.Inventory.Inventory](#) is open

Definition at line 43 of file [THInput.cs](#).

6.62.3.3 inputButtons

```
Dictionary<string, object> BeeGame.Core.THInput.inputButtons [static], [private]
```

Initial value:

```
= new Dictionary<string, object>()
{
    {"Forward", KeyCode.W},
    {"Backward", KeyCode.S },
    {"Right", KeyCode.D },
    {"Left", KeyCode.A },
    {"Player Inventory", KeyCode.E },
    {"Quest Book", KeyCode.Mouse1 },
    {"Interact", KeyCode.Mouse1 },
    {"Place", KeyCode.Mouse1 },
    {"Break Block", KeyCode.Mouse0 },
    {"Close Menu/Inventory", new KeyCode[2] { KeyCode.Escape, KeyCode.E } },
    {"Jump", KeyCode.Space }
}
```

Button identifiers and KeyCode

Definition at line 16 of file [THInput.cs](#).

6.62.3.4 isAnotherInventoryOpen

```
bool BeeGame.Core.THInput.isAnotherInventoryOpen [static]
```

If another inventory is open true, else false

Definition at line 34 of file [THInput.cs](#).

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityTypeReplacements/[T← HInput.cs](#)

6.63 BeeGame.Core.UnityTypeReplacements.THQuaternion Struct Reference

Public Member Functions

- [THQuaternion \(float x, float y, float z, float w\)](#)
- [THQuaternion \(THVector3 vector, float w\)](#)

Static Public Member Functions

- static [THQuaternion operator+ \(THQuaternion a, THQuaternion b\)](#)
- static [THQuaternion operator+ \(THQuaternion a, float b\)](#)
- static [THQuaternion operator- \(THQuaternion a, THQuaternion b\)](#)
- static [THQuaternion operator- \(THQuaternion a, float b\)](#)
- static [THQuaternion operator* \(THQuaternion a, THQuaternion b\)](#)
- static [THQuaternion operator* \(THQuaternion a, float b\)](#)
- static [THQuaternion operator/ \(THQuaternion a, THQuaternion b\)](#)
- static [THQuaternion operator/ \(THQuaternion a, float b\)](#)
- static implicit [operator THQuaternion \(Quaternion q\)](#)
- static implicit [operator Quaternion \(THQuaternion q\)](#)

Public Attributes

- float `x`
- float `y`
- float `z`
- float `w`

Static Public Attributes

- static Quaternion `identity` => new Quaternion(0, 0, 0, 1)

6.63.1 Detailed Description

Definition at line 7 of file [THQuaternion.cs](#).

6.63.2 Constructor & Destructor Documentation

6.63.2.1 THQuaternion() [1/2]

```
BeeGame.Core.UnityTypeReplacements.THQuaternion.THQuaternion (
    float x,
    float y,
    float z,
    float w )
```

Definition at line 16 of file [THQuaternion.cs](#).

```
00017     {
00018         this.x = x;
00019         this.y = y;
00020         this.z = z;
00021         this.w = w;
00022     }
```

6.63.2.2 THQuaternion() [2/2]

```
BeeGame.Core.UnityTypeReplacements.THQuaternion.THQuaternion (
    THVector3 vector,
    float w )
```

Definition at line 24 of file [THQuaternion.cs](#).

```
00025     {
00026         x = vector.x;
00027         y = vector.y;
00028         z = vector.z;
00029         this.w = w;
00030     }
```

6.63.3 Member Function Documentation

6.63.3.1 operator Quaternion()

```
static implicit BeeGame.Core.UnityTypeReplacements.THQuaternion.operator Quaternion (
    THQuaternion q) [static]
```

Definition at line 120 of file [THQuaternion.cs](#).

```
00121     {
00122         return new Quaternion(q.x, q.y, q.z, q.w);
00123     }
```

6.63.3.2 operator THQuaternion()

```
static implicit BeeGame.Core.UnityTypeReplacements.THQuaternion.operator THQuaternion (
    Quaternion q) [static]
```

Definition at line 115 of file [THQuaternion.cs](#).

```
00116     {
00117         return new THQuaternion(q.x, q.y, q.z, q.w);
00118     }
```

6.63.3.3 operator*() [1/2]

```
static THQuaternion BeeGame.Core.UnityTypeReplacements.THQuaternion.operator* (
    THQuaternion a,
    THQuaternion b) [static]
```

Definition at line 73 of file [THQuaternion.cs](#).

```
00074     {
00075         a.x *= b.x;
00076         a.y *= b.y;
00077         a.z *= b.z;
00078         a.w *= b.w;
00079
00080         return a;
00081     }
```

6.63.3.4 operator*() [2/2]

```
static THQuaternion BeeGame.Core.UnityTypeReplacements.THQuaternion.operator* (
    THQuaternion a,
    float b )  [static]
```

Definition at line 83 of file [THQuaternion.cs](#).

```
00084      {
00085          a.x *= b;
00086          a.y *= b;
00087          a.z *= b;
00088          a.w *= b;
00089      return a;
00090  }
00091 }
```

6.63.3.5 operator+() [1/2]

```
static THQuaternion BeeGame.Core.UnityTypeReplacements.THQuaternion.operator+ (
    THQuaternion a,
    THQuaternion b )  [static]
```

Definition at line 33 of file [THQuaternion.cs](#).

```
00034      {
00035          a.x += b.x;
00036          a.y += b.y;
00037          a.z += b.z;
00038          a.w += b.w;
00039      return a;
00040  }
00041 }
```

6.63.3.6 operator+() [2/2]

```
static THQuaternion BeeGame.Core.UnityTypeReplacements.THQuaternion.operator+ (
    THQuaternion a,
    float b )  [static]
```

Definition at line 43 of file [THQuaternion.cs](#).

```
00044      {
00045          a.x += b;
00046          a.y += b;
00047          a.z += b;
00048          a.w += b;
00049      return a;
00050  }
00051 }
```

6.63.3.7 operator-() [1/2]

```
static THQuaternion BeeGame.Core.UnityTypeReplacements.THQuaternion.operator- (
    THQuaternion a,
    THQuaternion b )  [static]
```

Definition at line 53 of file [THQuaternion.cs](#).

```
00054      {
00055          a.x -= b.x;
00056          a.y -= b.y;
00057          a.z -= b.z;
00058          a.w -= b.w;
00059      return a;
00060  }
00061 }
```

6.63.3.8 operator-() [2/2]

```
static THQuaternion BeeGame.Core.UnityTypeReplacements.THQuaternion.operator- (
    THQuaternion a,
    float b )  [static]
```

Definition at line 63 of file [THQuaternion.cs](#).

```
00064      {
00065          a.x -= b;
00066          a.y -= b;
00067          a.z -= b;
00068          a.w -= b;
00069      return a;
00070  }
00071 }
```

6.63.3.9 operator/() [1/2]

```
static THQuaternion BeeGame.Core.UnityTypeReplacements.THQuaternion.operator/ (
    THQuaternion a,
    THQuaternion b )  [static]
```

Definition at line 93 of file [THQuaternion.cs](#).

```
00094      {
00095          a.x /= b.x;
00096          a.y /= b.y;
00097          a.z /= b.z;
00098          a.w /= b.w;
00099      return a;
00100  }
00101 }
```

6.63.3.10 operator/() [2/2]

```
static THQuaternion BeeGame.Core.UnityTypeReplacements.THQuaternion.operator/ (
    THQuaternion a,
    float b ) [static]
```

Definition at line 103 of file [THQuaternion.cs](#).

```
00104         {
00105             a.x /= b;
00106             a.y /= b;
00107             a.z /= b;
00108             a.w /= b;
00109             return a;
00110         }
```

6.63.4 Member Data Documentation

6.63.4.1 identity

```
Quaternion BeeGame.Core.UnityTypeReplacements.THQuaternion.identity => new Quaternion(0, 0, 0,
1) [static]
```

Definition at line 14 of file [THQuaternion.cs](#).

6.63.4.2 w

```
float BeeGame.Core.UnityTypeReplacements.THQuaternion.w
```

Definition at line 12 of file [THQuaternion.cs](#).

6.63.4.3 x

```
float BeeGame.Core.UnityTypeReplacements.THQuaternion.x
```

Definition at line 9 of file [THQuaternion.cs](#).

6.63.4.4 y

```
float BeeGame.Core.UnityTypeReplacements.THQuaternion.y
```

Definition at line 10 of file [THQuaternion.cs](#).

6.63.4.5 z

```
float BeeGame.Core.UnityTypeReplacements.THQuaternion.z
```

Definition at line 11 of file [THQuaternion.cs](#).

The documentation for this struct was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityTypeReplacements/[T](#)↔
[HQuaternion.cs](#)

6.64 BeeGame.Core.THVector2 Struct Reference

Serializable version of Vector2

Public Member Functions

- [THVector2 \(float x, float y\)](#)
Constructor from 2 floats
- [THVector2 \(THVector2 vec2\)](#)
Constructor from another THVector2
- [THVector2 \(Vector2 vec2\)](#)
Constructor from Vector2
- [override bool Equals \(object obj\)](#)
- [override int GetHashCode \(\)](#)
- [override string ToString \(\)](#)

Static Public Member Functions

- [static bool operator== \(THVector2 a, THVector2 b\)](#)
- [static bool operator!= \(THVector2 a, THVector2 b\)](#)
- [static THVector2 operator+ \(THVector2 a, THVector2 b\)](#)
- [static THVector2 operator+ \(THVector2 a, float b\)](#)
- [static THVector2 operator+ \(float a, THVector2 b\)](#)
- [static THVector2 operator- \(THVector2 a, THVector2 b\)](#)
- [static THVector2 operator- \(THVector2 a, float b\)](#)
- [static THVector2 operator- \(float a, THVector2 b\)](#)
- [static THVector2 operator* \(THVector2 a, THVector2 b\)](#)
- [static THVector2 operator* \(THVector2 a, float b\)](#)
- [static THVector2 operator* \(float a, THVector2 b\)](#)
- [static THVector2 operator/ \(THVector2 a, THVector2 b\)](#)
- [static THVector2 operator/ \(THVector2 a, float b\)](#)
- [static THVector2 operator/ \(float a, THVector2 b\)](#)
- [static implicit operator Vector2 \(THVector2 vec2\)](#)
- [static implicit operator THVector2 \(Vector2 vec2\)](#)

Public Attributes

- [float x](#)
X position
- [float y](#)
Y position

6.64.1 Detailed Description

Serializable version of Vector2

Definition at line 10 of file [THVector2.cs](#).

6.64.2 Constructor & Destructor Documentation

6.64.2.1 THVector2() [1/3]

```
BeeGame.Core.THVector2.THVector2 (
    float x,
    float y )
```

Constructor from 2 floats

Parameters

x	X position
y	Y position

Definition at line 29 of file [THVector2.cs](#).

```
00030     {
00031         this.x = x;
00032         this.y = y;
00033     }
```

6.64.2.2 THVector2() [2/3]

```
BeeGame.Core.THVector2.THVector2 (
    THVector2 vec2 )
```

Constructor from another [THVector2](#)

Parameters

vec2	Vector to make this from
------	--------------------------

Definition at line 39 of file [THVector2.cs](#).

```
00040     {
00041         this = vec2;
00042     }
```

6.64.2.3 THVector2() [3/3]

```
BeeGame.Core.THVector2.THVector2 (
    Vector2 vec2 )
```

Constructor from Vector2

Parameters

vec2	Vector to make this from
------	--------------------------

Definition at line 48 of file [THVector2.cs](#).

```
00049     {
00050         this = vec2;
00051     }
```

6.64.3 Member Function Documentation

6.64.3.1 Equals()

```
override bool BeeGame.Core.THVector2.Equals (
    object obj )
```

Definition at line 55 of file [THVector2.cs](#).

```
00056     {
00057         if (!(obj is THVector2))
00058             return false;
00059         if (obj.GetHashCode() == GetHashCode())
00060             return true;
00061         return false;
00062     }
```

6.64.3.2 GetHashCode()

```
override int BeeGame.Core.THVector2.GetHashCode ( )
```

Definition at line 64 of file [THVector2.cs](#).

```
00065     {
00066         unchecked
00067         {
00068             int hash = 13;
00069             hash *= 443 * x.GetHashCode();
00070             hash *= 373 * y.GetHashCode();
00071             return hash;
00072         }
00073     }
```

6.64.3.3 operator THVector2()

```
static implicit BeeGame.Core.THVector2.operator THVector2 (
    Vector2 vec2 ) [static]
```

Definition at line 171 of file [THVector2.cs](#).

```
00172     {
00173         return new THVector2(vec2.x, vec2.y);
00174     }
```

6.64.3.4 operator Vector2()

```
static implicit BeeGame.Core.THVector2.operator Vector2 (
    THVector2 vec2 ) [static]
```

Definition at line 166 of file [THVector2.cs](#).

```
00167     {
00168         return new Vector2(vec2.x, vec2.y);
00169     }
```

6.64.3.5 operator"!=()

```
static bool BeeGame.Core.THVector2.operator!= (
    THVector2 a,
    THVector2 b ) [static]
```

Definition at line 86 of file [THVector2.cs](#).

```
00087     {
00088         return !(a == b);
00089     }
```

6.64.3.6 operator*() [1/3]

```
static THVector2 BeeGame.Core.THVector2.operator* (
    THVector2 a,
    THVector2 b ) [static]
```

Definition at line 127 of file [THVector2.cs](#).

```
00128     {
00129         a.x *= b.x;
00130         a.y *= b.y;
00131     }
00132     return a;
00133 }
```

6.64.3.7 operator*() [2/3]

```
static THVector2 BeeGame.Core.THVector2.operator* (
    THVector2 a,
    float b ) [static]
```

Definition at line 134 of file [THVector2.cs](#).

```
00135     {
00136         a.x *= b;
00137         a.y *= b;
00138     }
00139     return a;
00140 }
```

6.64.3.8 operator*() [3/3]

```
static THVector2 BeeGame.Core.THVector2.operator* (
    float a,
    THVector2 b ) [static]
```

Definition at line 141 of file [THVector2.cs](#).

```
00142     {
00143         return new THVector2(a * b.x, a * b.y);
00144     }
```

6.64.3.9 operator+() [1/3]

```
static THVector2 BeeGame.Core.THVector2.operator+ (
    THVector2 a,
    THVector2 b ) [static]
```

Definition at line 91 of file [THVector2.cs](#).

```
00092     {
00093         a.x += b.x;
00094         a.y += b.y;
00095     }
00096     return a;
00097 }
```

6.64.3.10 operator+() [2/3]

```
static THVector2 BeeGame.Core.THVector2.operator+ (
    THVector2 a,
    float b ) [static]
```

Definition at line 98 of file [THVector2.cs](#).

```
00099     {
00100         a.x += b;
00101         a.y += b;
00102     }
00103     return a;
00104 }
```

6.64.3.11 operator+() [3/3]

```
static THVector2 BeeGame.Core.THVector2.operator+ (
    float a,
    THVector2 b ) [static]
```

Definition at line 105 of file [THVector2.cs](#).

```
00106     {
00107         return new THVector2(a + b.x, a + b.y);
00108     }
```

6.64.3.12 operator-() [1/3]

```
static THVector2 BeeGame.Core.THVector2.operator- (
    THVector2 a,
    THVector2 b ) [static]
```

Definition at line 109 of file [THVector2.cs](#).

```
00110     {
00111         a.x -= b.x;
00112         a.y -= b.y;
00113     }
00114     return a;
00115 }
```

6.64.3.13 operator-() [2/3]

```
static THVector2 BeeGame.Core.THVector2.operator- (
    THVector2 a,
    float b ) [static]
```

Definition at line 116 of file [THVector2.cs](#).

```
00117     {
00118         a.x += b;
00119         a.y += b;
00120     }
00121     return a;
00122 }
```

6.64.3.14 operator-() [3/3]

```
static THVector2 BeeGame.Core.THVector2.operator- (
    float a,
    THVector2 b ) [static]
```

Definition at line 123 of file [THVector2.cs](#).

```
00124     {
00125         return new THVector2(a - b.x, a - b.y);
00126     }
```

6.64.3.15 operator/() [1/3]

```
static THVector2 BeeGame.Core.THVector2.operator/ (
    THVector2 a,
    THVector2 b ) [static]
```

Definition at line 145 of file [THVector2.cs](#).

```
00146     {
00147         a.x /= b.x;
00148         a.y /= b.y;
00149         return a;
00150     }
```

6.64.3.16 operator/() [2/3]

```
static THVector2 BeeGame.Core.THVector2.operator/ (
    THVector2 a,
    float b ) [static]
```

Definition at line 152 of file [THVector2.cs](#).

```
00153     {
00154         a.x /= b;
00155         a.y /= b;
00156         return a;
00157     }
```

6.64.3.17 operator/() [3/3]

```
static THVector2 BeeGame.Core.THVector2.operator/ (
    float a,
    THVector2 b ) [static]
```

Definition at line 159 of file [THVector2.cs](#).

```
00160     {
00161         return new THVector2(a / b.x, a / b.y);
00162     }
```

6.64.3.18 operator==()

```
static bool BeeGame.Core.THVector2.operator== (
    THVector2 a,
    THVector2 b ) [static]
```

Definition at line 82 of file [THVector2.cs](#).

```
00083     {
00084         return a.Equals(b);
00085     }
```

6.64.3.19 `ToString()`

```
override string BeeGame.Core.THVector2.ToString ( )
```

Definition at line 77 of file [THVector2.cs](#).

```
00078      {
00079          return $"{{x}}, {{y}}";
00080      }
```

6.64.4 Member Data Documentation

6.64.4.1 `x`

```
float BeeGame.Core.THVector2.x
```

X position

Definition at line 16 of file [THVector2.cs](#).

6.64.4.2 `y`

```
float BeeGame.Core.THVector2.y
```

Y position

Definition at line 20 of file [THVector2.cs](#).

The documentation for this struct was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityTypeReplacements/[THVector2.cs](#)

6.65 BeeGame.Core.THVector3 Struct Reference

Serializable version of Vector3

Public Member Functions

- [THVector3](#) (float `x`, float `y`, float `z`)

Constructor from 3 floats
- [THVector3](#) ([THVector3](#) `vec3`)

Constructor from another [THVector3](#)
- [THVector3](#) ([Vector3](#) `vec3`)

Constructor from another [Vector3](#)
- [THVector3](#) ([Terrain.ChunkWorldPos](#) `vec3`)

Constructor from another [Terrain.ChunkWorldPos](#)
- override bool [Equals](#) (object `obj`)

This this vector == to another
- override int [GetHashCode](#) ()

Gets the hascode for the vector
- override string [ToString](#) ()

Formats the vector as a nice string

Static Public Member Functions

- static float [Distance](#) ([THVector3](#) a, [THVector3](#) b)
Distance between 2 vectors
- static bool [operator==](#) ([THVector3](#) a, [THVector3](#) b)
Checks if a == b
- static bool [operator!=](#) ([THVector3](#) a, [THVector3](#) b)
Inverse of ==
- static [THVector3 operator+](#) ([THVector3](#) a, [THVector3](#) b)
Adds vector a and b
- static [THVector3 operator+](#) ([THVector3](#) a, float b)
Adds b to vector a
- static [THVector3 operator+](#) (float a, [THVector3](#) b)
Adds a to vector b
- static [THVector3 operator-](#) ([THVector3](#) a, [THVector3](#) b)
Subtracts vector a and b
- static [THVector3 operator-](#) ([THVector3](#) a, float b)
Subtracts b from vector a
- static [THVector3 operator-](#) (float a, [THVector3](#) b)
Subtracts a from vector b
- static [THVector3 operator*](#) ([THVector3](#) a, [THVector3](#) b)
Multiples vector a and b
- static [THVector3 operator*](#) ([THVector3](#) a, float b)
Multiples b to vector a
- static [THVector3 operator*](#) (float a, [THVector3](#) b)
Multiples a to vector b
- static [THVector3 operator/](#) ([THVector3](#) a, [THVector3](#) b)
Divides vector a by vector b
- static [THVector3 operator/](#) ([THVector3](#) a, float b)
Divides vector a by vector b
- static [THVector3 operator/](#) (float a, [THVector3](#) b)
Divides vector b by vector a
- static implicit [operator Vector3](#) ([THVector3](#) vec3)
Converts THVector3 to Vector3 implicitly
- static implicit [operator THVector3](#) ([Vector3](#) vec3)
Converts Vector3 to THVector3 implicitly
- static [operator Quaternion](#) ([THVector3](#) vec3)
Converts a THVector3 to a Quaternion

Public Attributes

- float [x](#)
X position
- float [y](#)
Y position
- float [z](#)
Z position

6.65.1 Detailed Description

Serializable version of Vector3

Definition at line 10 of file [THVector3.cs](#).

6.65.2 Constructor & Destructor Documentation

6.65.2.1 THVector3() [1/4]

```
BeeGame.Core.THVector3.THVector3 (
    float x,
    float y,
    float z )
```

Constructor from 3 floats

Parameters

<i>x</i>	X position
<i>y</i>	Y position
<i>z</i>	Z position

Definition at line 34 of file [THVector3.cs](#).

```
00035     {
00036         this.x = x;
00037         this.y = y;
00038         this.z = z;
00039     }
```

6.65.2.2 THVector3() [2/4]

```
BeeGame.Core.THVector3.THVector3 (
    THVector3 vec3 )
```

Constructor from another [THVector3](#)

Parameters

<i>vec3</i>	Vector to make this from
-------------	--------------------------

Definition at line 45 of file [THVector3.cs](#).

```
00046     {
00047         this = vec3;
00048     }
```

6.65.2.3 THVector3() [3/4]

```
BeeGame.Core.THVector3.THVector3 (
    Vector3 vec3 )
```

Constructor from another Vector3

Parameters

vec3	Vector to make this from
------	--------------------------

Definition at line 54 of file [THVector3.cs](#).

```
00055     {
00056         this = vec3;
00057     }
```

6.65.2.4 THVector3() [4/4]

```
BeeGame.Core.THVector3.THVector3 (
    Terrain.ChunkWorldPos vec3 )
```

Constructor from another [Terrain.ChunkWorldPos](#)

Parameters

vec3	Vector to make this from
------	--------------------------

Definition at line 63 of file [THVector3.cs](#).

```
00064     {
00065         this = vec3;
00066     }
```

6.65.3 Member Function Documentation

6.65.3.1 Distance()

```
static float BeeGame.Core.THVector3.Distance (
    THVector3 a,
    THVector3 b ) [static]
```

Distance between 2 vectors

Parameters

a	First Vector
b	Second Vector

Returns

Distance between *a* and *b*

Definition at line 76 of file [THVector3.cs](#).

```
00077      {
00078          return (float)Math.Sqrt(Math.Pow((a.x - b.x), 2) + Math.Pow((a.y - b.y), 2) + Math.Pow((a.z - b
00079 .z), 2));
}
```

6.65.3.2 Equals()

```
override bool BeeGame.Core.THVector3.Equals (
    object obj )
```

This this vector == to another

Parameters

<i>obj</i>	object to check against
------------	-------------------------

Returns

Definition at line 88 of file [THVector3.cs](#).

```
00089      {
00090          if (!(obj is THVector3))
00091              return false;
00092          if (obj.GetHashCode() == GetHashCode())
00093              return true;
00094          return false;
00095      }
```

6.65.3.3 GetHashCode()

```
override int BeeGame.Core.THVector3.GetHashCode ( )
```

Gets the hascode for the vector

Returns

Definition at line 101 of file [THVector3.cs](#).

```
00102      {
00103          unchecked
00104          {
00105              int hash = 13;
00106
00107              hash *= 443 * x.GetHashCode();
00108              hash *= 373 * y.GetHashCode();
00109              hash *= 127 * z.GetHashCode();
00110
00111          }
00112      }
00113 }
```

6.65.3.4 operator Quaternion()

```
static BeeGame.Core.THVector3.operator Quaternion (
    THVector3 vec3 ) [explicit], [static]
```

Converts a [THVector3](#) to a Quaternion

Parameters

<code>vec3</code>	Vector to convert to Quaternion
-------------------	---------------------------------

Explicit as conversion is not exact

Definition at line 327 of file [THVector3.cs](#).

```
00328     {
00329         return new Quaternion(vec3.x, vec3.y, vec3.z, 0);
00330     }
```

6.65.3.5 operator THVector3()

```
static implicit BeeGame.Core.THVector3.operator THVector3 (
    Vector3 vec3 ) [static]
```

Converts Vector3 to [THVector3](#) implicitly

Parameters

<code>vec3</code>	Vector to convert
-------------------	-------------------

Definition at line 313 of file [THVector3.cs](#).

```
00314     {
00315         return new THVector3(vec3.x, vec3.y, vec3.z);
00316     }
```

6.65.3.6 operator Vector3()

```
static implicit BeeGame.Core.THVector3.operator Vector3 (
    THVector3 vec3 ) [static]
```

Converts [THVector3](#) to Vector3 implicitly

Parameters

<code>vec3</code>	Vector to convert
-------------------	-------------------

Definition at line 304 of file [THVector3.cs](#).

```
00305      {
00306          return new Vector3(vec3.x, vec3.y, vec3.z);
00307      }
```

6.65.3.7 operator"!=()

```
static bool BeeGame.Core.THVector3.operator!= (
    THVector3 a,
    THVector3 b ) [static]
```

Inverse of ==

Parameters

<i>a</i>	First vector
<i>b</i>	Second vector

Returns

true if *a* != *b*

Definition at line 140 of file [THVector3.cs](#).

```
00141      {
00142          return !(a == b);
00143      }
```

6.65.3.8 operator*() [1/3]

```
static THVector3 BeeGame.Core.THVector3.operator* (
    THVector3 a,
    THVector3 b ) [static]
```

Multiplies vector *a* and *b*

Parameters

<i>a</i>	Vector <i>a</i>
<i>b</i>	Vector <i>b</i>

Returns

returns new vector that is the product of *a* and *b*

Definition at line 227 of file [THVector3.cs](#).

```
00228      {
00229          a.x *= b.x;
00230          a.y *= b.y;
00231          a.z *= b.z;
00232
00233          return a;
00234      }
```

6.65.3.9 operator*() [2/3]

```
static THVector3 BeeGame.Core.THVector3.operator* (
    THVector3 a,
    float b ) [static]
```

Multiples b to vector a

Parameters

a	Vector a
b	float b

Returns

returns new vector that is the product of a and b

Definition at line 241 of file [THVector3.cs](#).

```
00242      {
00243          a.x *= b;
00244          a.y *= b;
00245          a.z *= b;
00246
00247          return a;
00248      }
```

6.65.3.10 operator*() [3/3]

```
static THVector3 BeeGame.Core.THVector3.operator* (
    float a,
    THVector3 b ) [static]
```

Multiples a to vector b

Parameters

a	Vector a
b	float b

Returns

returns new vector that is the product of a and b

Definition at line 255 of file [THVector3.cs](#).

```
00256     {
00257         return new THVector3(a * b.x, a * b.y, a * b.z);
00258     }
```

6.65.3.11 operator+() [1/3]

```
static THVector3 BeeGame.Core.THVector3.operator+
    THVector3 a,
    THVector3 b ) [static]
```

Adds vector a and b

Parameters

<i>a</i>	Vector a
<i>b</i>	Vector b

Returns

returns new vector that is the sum of a and b

Definition at line 151 of file [THVector3.cs](#).

```
00152     {
00153         a.x += b.x;
00154         a.y += b.y;
00155         a.z += b.z;
00156
00157         return a;
00158     }
```

6.65.3.12 operator+() [2/3]

```
static THVector3 BeeGame.Core.THVector3.operator+
    THVector3 a,
    float b ) [static]
```

Adds b to vector a

Parameters

<i>a</i>	Vector a
<i>b</i>	float b

Returns

returns new vector that is the sum of a and b

Definition at line 165 of file [THVector3.cs](#).

```
00166      {
00167          a.x += b;
00168          a.y += b;
00169          a.z += b;
00170
00171          return a;
00172      }
```

6.65.3.13 operator+() [3/3]

```
static THVector3 BeeGame.Core.THVector3.operator+ (
    float a,
    THVector3 b ) [static]
```

Adds a to vector b

Parameters

<i>a</i>	Vector a
<i>b</i>	float b

Returns

returns new vector that is the sum of a and b

Definition at line 179 of file [THVector3.cs](#).

```
00180      {
00181          return new THVector3(a + b.x, a + b.y, a + b.z);
00182      }
```

6.65.3.14 operator-() [1/3]

```
static THVector3 BeeGame.Core.THVector3.operator- (
    THVector3 a,
    THVector3 b ) [static]
```

Subtracts vector a and b

Parameters

<i>a</i>	Vector a
<i>b</i>	Vector b

Returns

returns new vector that is the subtraction of a from b

Definition at line 189 of file [THVector3.cs](#).

```
00190         {
00191             a.x -= b.x;
00192             a.y -= b.y;
00193             a.z -= b.z;
00194
00195         return a;
00196     }
```

6.65.3.15 operator-() [2/3]

```
static THVector3 BeeGame.Core.THVector3.operator- (
    THVector3 a,
    float b ) [static]
```

Subtracts b from vector a

Parameters

<i>a</i>	Vector a
<i>b</i>	float b

Returns

returns new vector that is the subtraction of a from b

Definition at line 203 of file [THVector3.cs](#).

```
00204         {
00205             a.x += b;
00206             a.y += b;
00207             a.z += b;
00208
00209         return a;
00210     }
```

6.65.3.16 operator-() [3/3]

```
static THVector3 BeeGame.Core.THVector3.operator- (
    float a,
    THVector3 b ) [static]
```

Subtracts a from vector b

Parameters

<i>a</i>	Vector a
<i>b</i>	float b

Returns

returns new vector that is the subtraction of a from b

Definition at line 217 of file [THVector3.cs](#).

```
00218     {
00219         return new THVector3(a - b.x, a - b.y, a - b.z);
00220     }
```

6.65.3.17 operator/() [1/3]

```
static THVector3 BeeGame.Core.THVector3.operator/ (
    THVector3 a,
    THVector3 b ) [static]
```

Divides vector a by vector b

Parameters

<i>a</i>	Vector a
<i>b</i>	Vector b

Returns

returns new vector that is the division of a by b

Definition at line 265 of file [THVector3.cs](#).

```
00266     {
00267         a.x /= b.x;
00268         a.y /= b.y;
00269         a.z /= b.z;
00270
00271         return a;
00272     }
```

6.65.3.18 operator/() [2/3]

```
static THVector3 BeeGame.Core.THVector3.operator/ (
    THVector3 a,
    float b ) [static]
```

Divides vector a by vector b

Parameters

<i>a</i>	Vector a
<i>b</i>	float b

Returns

returns new vector that is the division of a by b

Definition at line 279 of file [THVector3.cs](#).

```
00280         {
00281             a.x /= b;
00282             a.y /= b;
00283             a.z /= b;
00284
00285             return a;
00286 }
```

6.65.3.19 operator/() [3/3]

```
static THVector3 BeeGame.Core.THVector3.operator/ (
    float a,
    THVector3 b ) [static]
```

Divides vector b by vector a

Parameters

<i>a</i>	Vector a
<i>b</i>	float b

Returns

returns new vector that is the division of b by a

Definition at line 293 of file [THVector3.cs](#).

```
00294         {
00295             return new THVector3(a / b.x, a / b.y, a / b.z);
00296 }
```

6.65.3.20 operator==()

```
static bool BeeGame.Core.THVector3.operator== (
    THVector3 a,
    THVector3 b ) [static]
```

Checks if *a* == *b*

Parameters

<i>a</i>	First vector
<i>b</i>	Second vector

Returns

true if $a == b$

Definition at line 130 of file [THVector3.cs](#).

```
00131     {  
00132         return a.Equals(b);  
00133     }
```

6.65.3.21 ToString()

```
override string BeeGame.Core.THVector3.ToString ( )
```

Formats the vector as a nice string

Returns

The vector as a nice string

Definition at line 119 of file [THVector3.cs](#).

```
00120     {  
00121         return $"{x}, {y}, {z}";  
00122     }
```

6.65.4 Member Data Documentation

6.65.4.1 x

```
float BeeGame.Core.THVector3.x
```

X position

Definition at line 16 of file [THVector3.cs](#).

6.65.4.2 y

```
float BeeGame.Core.THVector3.y
```

Y position

Definition at line 20 of file [THVector3.cs](#).

6.65.4.3 z

```
float BeeGame.Core.THVector3.z
```

Z position

Definition at line 24 of file [THVector3.cs](#).

The documentation for this struct was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityTypeReplacements/[T←HVector3.cs](#)

6.66 BeeGame.Items.Tile Struct Reference

Position of the items texture

Public Attributes

- int [x](#)
X pos of the texture
- int [y](#)
Y pos of the texture

6.66.1 Detailed Description

Position of the items texture

Definition at line 415 of file [Item.cs](#).

6.66.2 Member Data Documentation

6.66.2.1 x

```
int BeeGame.Items.Tile.x
```

X pos of the texture

Definition at line 420 of file [Item.cs](#).

6.66.2.2 y

```
int BeeGame.Items.Tile.y
```

Y pos of the texture

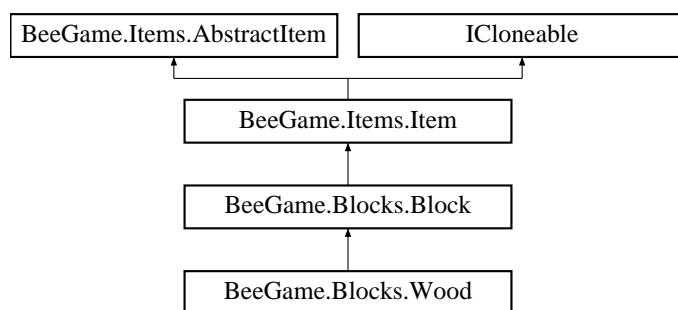
Definition at line 424 of file [Item.cs](#).

The documentation for this struct was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/[Item.cs](#)

6.67 BeeGame.Blocks.Wood Class Reference

Inheritance diagram for BeeGame.Blocks.Wood:



Public Member Functions

- [Wood \(\)](#)
- override Sprite [GetItemSprite \(\)](#)
Returns the sprite for the item
- override Tile TexturePosition ([Direction direction](#))
Texture position of the items texture
- override int [GetHashCode \(\)](#)
Base ID of the block
- override string [ToString \(\)](#)
Returns the name and ID of the block as a string

Static Public Attributes

- static new int [ID => 5](#)

Additional Inherited Members

6.67.1 Detailed Description

Definition at line 13 of file [Wood.cs](#).

6.67.2 Constructor & Destructor Documentation

6.67.2.1 Wood()

BeeGame.Blocks.Wood.Wood ()

Definition at line 17 of file [Wood.cs](#).

```
00017           : base("Wood")
00018     {
00019     }
00020 }
```

6.67.3 Member Function Documentation

6.67.3.1 GetHashCode()

override int BeeGame.Blocks.Wood.GetHashCode () [virtual]

Base ID of the block

Returns

5

Reimplemented from [BeeGame.Blocks.Block](#).

Definition at line 39 of file [Wood.cs](#).

```
00040     {
00041         return ID;
00042     }
```

6.67.3.2 GetItemSprite()

override Sprite BeeGame.Blocks.Wood.GetItemSprite () [virtual]

Returns the sprite for the item

Returns

Sprite for this item

Reimplemented from [BeeGame.Blocks.Block](#).

Definition at line 23 of file [Wood.cs](#).

```
00024     {
00025         return SpriteDictionary.GetSprite("Wood");
00026     }
```

6.67.3.3 TexturePosition()

```
override Tile BeeGame.Blocks.Wood.TexturePosition (
    Direction direction) [virtual]
```

Texture position of the items texture

Parameters

<i>direction</i>	Direction for the texture
------------------	---------------------------

Returns

Position of the texture

Reimplemented from [BeeGame.Items.Item](#).

Definition at line [29](#) of file [Wood.cs](#).

```
00030     {  
00031         return new Tile() { x = 7, y = 9 };  
00032     }
```

6.67.3.4 ToString()

```
override string BeeGame.Blocks.Wood.ToString ()
```

Returns the name and ID of the block as a string

Returns

A nicely formatted string

Definition at line [48](#) of file [Wood.cs](#).

```
00049     {  
00050         return $"{itemName} \nID: {GetItemID()}";  
00051     }
```

6.67.4 Member Data Documentation

6.67.4.1 ID

```
new int BeeGame.Blocks.Wood.ID => 5 [static]
```

Definition at line [15](#) of file [Wood.cs](#).

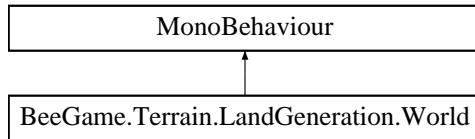
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/[Wood.cs](#)

6.68 BeeGame.Terrain.LandGeneration.World Class Reference

Allows inter Chunk communication as it stores a list of active chunks

Inheritance diagram for BeeGame.Terrain.LandGeneration.World:



Public Member Functions

- void [CreateChunk](#) (int x, int y, int z)
Creates a chunk at the given x, y, z
- void [DestroyChunk](#) (int x, int y, int z)
Destroys a Chunk st the given x, y, z position
- void [SetBlock](#) (int x, int y, int z, [Block](#) block, bool saveChunk=false)
Sets a Block at the given position
- [Chunk](#) [GetChunk](#) (int x, int y, int z)
Gets a chunk at eh given x, y, z
- [Block](#) [GetBlock](#) (int x, int y, int z)
Gets a Block at the given position

Public Attributes

- Dictionary<[ChunkWorldPos](#), [Chunk](#) > [chunks](#) = new Dictionary<[ChunkWorldPos](#), [Chunk](#)>()
All of the currently loaded chunks
- GameObject [chunkPrefab](#)
The chunk prefab
- bool [chunkHasMadeCollisionMesh](#) = false
Has a Chunk made a collision mesh?

Private Member Functions

- void [UpdateIfEqual](#) (int value1, int value2, [ChunkWorldPos](#) pos)
Updates a chunk if value1 and value2 are equal

6.68.1 Detailed Description

Allows inter Chunk communication as it stores a list of active chunks

Definition at line 14 of file [World.cs](#).

6.68.2 Member Function Documentation

6.68.2.1 CreateChunk()

```
void BeeGame.Terrain.LandGeneration.World.CreateChunk (
    int x,
    int y,
    int z )
```

Creates a chunk at the given x, y, z

Parameters

<i>x</i>	X pos to make the new chunk
<i>y</i>	Y pos to make the new chunk
<i>z</i>	Z pos to make the new chunk

Definition at line 41 of file [World.cs](#).

```

00042         {
00043             /* pos of the chunk
00044             ChunkWorldPos pos = new ChunkWorldPos(x, y, z);
00045
00046             /* makes the chunk at the given position
00047             GameObject newChunk = Instantiate(chunkPrefab, new Vector3(x, y, z), Quaternion.
00048             identity);
00049             Chunk chunk = newChunk.GetComponent<Chunk>();
00050
00051             /* setting the chunks pos and a reference to this
00052             chunk.chunkWorldPos = pos;
00053             chunk.world = this;
00054
00055             /* adds the new chunk to the dictionary
00056             chunks.Add(pos, chunk);
00057
00058             /* generates the new chunks blocks
00059             chunk = new TerrainGeneration().ChunkGen(chunk);
00060
00061             //loads any blocks that the chunk has had modified
00062             Serialization.Serialization.LoadChunk(chunk);
00063
00064             /* updates all chunks around this one to reduce drawing of unnecessary faces
00065             chunks.TryGetValue(new ChunkWorldPos(x, y - 16, z), out chunk);
00066             if (chunk != null)
00067                 chunk.update = true;
00068
00069             chunks.TryGetValue(new ChunkWorldPos(x, y, z - 16), out chunk);
00070             if (chunk != null)
00071                 chunk.update = true;
00072
00073             chunks.TryGetValue(new ChunkWorldPos(x - 16, y, z), out chunk);
00074             if (chunk != null)
00075                 chunk.update = true;
00076
00077             chunks.TryGetValue(new ChunkWorldPos(x, y + 16, z), out chunk);
00078             if (chunk != null)
00079                 chunk.update = true;
00080
00081             chunks.TryGetValue(new ChunkWorldPos(x, y, z + 16), out chunk);
00082             if (chunk != null)
00083                 chunk.update = true;
00084
00085             chunks.TryGetValue(new ChunkWorldPos(x + 16, y, z), out chunk);
00086             if (chunk != null)
00087                 chunk.update = true;
00088             /* the chunk will then make its meshes
00089     }

```

6.68.2.2 DestroyChunk()

```

void BeeGame.Terrain.LandGeneration.World.DestroyChunk (
    int x,
    int y,
    int z )

```

Destroys a Chunk at the given x, y, z position

Parameters

x	X pos if the chunk
y	Y pos if the chunk
z	Z pos if the chunk

Definition at line 97 of file [World.cs](#).

```
00098      {
00099          /* if teh chnks exists destroy it
00100         if (chunks.TryGetValue(new ChunkWorldPos(x, y, z), out Chunk chunk))
00101         {
00102             /* saves the chunk before destroying it incase any block were changed in it
00103             Serialization.Serialization.SaveChunk(chunk);
00104             Destroy(chunk.gameObject);
00105             chunks.Remove(new ChunkWorldPos(x, y, z));
00106         }
00107     }
```

6.68.2.3 GetBlock()

```
Block BeeGame.Terrain.LandGeneration.World.GetBlock (
    int x,
    int y,
    int z )
```

Gets a Block at the given position

Parameters

x	X pos of the block
y	Y pos of the block
z	Z pos of the block

Returns

Block at given x, y, z position

Definition at line 184 of file [World.cs](#).

```
00185      {
00186          /* gets the chunk that the block is in
00187          Chunk chunk = GetChunk(x, y, z);
00188
00189          if(chunk != null)
00190          {
00191              /* gets the block in the chunk
00192              return chunk.GetBlock(x - chunk.chunkWorldPos.
00193                         x, y - chunk.chunkWorldPos.y, z - chunk.chunkWorldPos.
00194                         z) ?? new Air();
00195
00196          /* returns an empty block is the chunk was not found
00197          return new Air();
00198      }
```

6.68.2.4 GetChunk()

```
Chunk BeeGame.Terrain.LandGeneration.World.GetChunk (
    int x,
    int y,
    int z )
```

Gets a chunk at eh given x, y, z

Parameters

x	X pos of the chunk
y	Y pos of the chunk
z	Z pos of the chunk

Returns

Chunk at given x, y, z

Definition at line 160 of file [World.cs](#).

```
00161      {
00162          float multiple = Chunk.chunkSize;
00163          /* rounds the given x, y, z to a multiple of 16 as chunks are 16x16x16 in size
00164          ChunkWorldPos pos = new ChunkWorldPos()
00165          {
00166              x = Mathf.FloorToInt(x / multiple) * Chunk.chunkSize,
00167              y = Mathf.FloorToInt(y / multiple) * Chunk.chunkSize,
00168              z = Mathf.FloorToInt(z / multiple) * Chunk.chunkSize
00169          };
00170
00171          /* gets the chunk if it exists
00172          chunks.TryGetValue(pos, out Chunk chunk);
00173          /* if the chunk does not exist will return null
00174          return chunk;
00175      }
```

6.68.2.5 SetBlock()

```
void BeeGame.Terrain.LandGeneration.World.SetBlock (
    int x,
    int y,
    int z,
    Block block,
    bool saveChunk = false )
```

Sets a Block at the given position

Parameters

x	X pos of the block
y	Y pos of the block
z	Z pos of the block
block	Block to be placed

Definition at line 118 of file [World.cs](#).

```

00119      {
00120          /*gets the chunk for the block to be placed in
00121          Chunk chunk = GetChunk(x, y, z);
00122
00123          /*if the chunk is not null and the block trying to be replaced is replaceable, replace it
00124          if(chunk != null && chunk.blocks[x - chunk.chunkWorldPos.
00125          x, y - chunk.chunkWorldPos.y, z - chunk.chunkWorldPos.
00126          z].breakable)
00127          {
00128              chunk.SetBlock(x - chunk.chunkWorldPos.x, y - chunk.
00129              chunkWorldPos.y, z - chunk.chunkWorldPos.z, block);
00130              chunk.update = true;
00131
00132              /*updates the nebouring chunks as when a block is broken it may be in the edje of the
00133              chunk so their meshes also need to be updated
00134              /*only updates chunks that need to be updated as not every chunk will need to be and
00135              sometimes none of them will need to be
00136
00137              /*checks if the block chaged is in the edge if the x value for the chunk
00138              UpdateIfEqual(x - chunk.chunkWorldPos.
00139              x, 0, new ChunkWorldPos(x - 1, y, z));
00140              UpdateIfEqual(x - chunk.chunkWorldPos.
00141              x, Chunk.chunkSize - 1, new ChunkWorldPos(x + 1, y, z));
00142
00143              /*checks if the block chaged is in the edge if the y value for the chunk
00144              UpdateIfEqual(y - chunk.chunkWorldPos.
00145              y, 0, new ChunkWorldPos(x, y - 1, z));
00146              UpdateIfEqual(y - chunk.chunkWorldPos.
00147              y, Chunk.chunkSize - 1, new ChunkWorldPos(x, y + 1, z));
00148
00149              /*checks if the block chaged is in the edge if the z value for the chunk
00150              UpdateIfEqual(z - chunk.chunkWorldPos.
00151              z, 0, new ChunkWorldPos(x, y, z - 1));
00152              UpdateIfEqual(z - chunk.chunkWorldPos.
00153              z, Chunk.chunkSize - 1, new ChunkWorldPos(x, y, z + 1));
00154
00155              if (saveChunk)
00156                  Serialization.Serialization.SaveChunk(chunk);
00157          }
00158      }

```

6.68.2.6 UpdateIfEqual()

```

void BeeGame.Terrain.LandGeneration.World.UpdateIfEqual (
    int value1,
    int value2,
    ChunkWorldPos pos ) [private]

```

Updates a chunk if *value1* and *value2* are equal

Parameters

<i>value1</i>	First value to check
<i>value2</i>	Second value to check
<i>pos</i>	Position of chunk to update if values are equal

Definition at line 206 of file [World.cs](#).

```

00207      {
00208          if(value1 == value2)
00209          {
00210              Chunk chunk = GetChunk(pos.x, pos.y, pos.z);
00211
00212              if (chunk != null)
00213                  chunk.update = true;
00214          }
00215      }

```

6.68.3 Member Data Documentation

6.68.3.1 chunkHasMadeCollisionMesh

```
bool BeeGame.Terrain.LandGeneration.World.chunkHasMadeCollisionMesh = false
```

Has a Chunk made a collision mesh?

Definition at line [30](#) of file [World.cs](#).

6.68.3.2 chunkPrefab

```
GameObject BeeGame.Terrain.LandGeneration.World.chunkPrefab
```

The chunk prefab

Definition at line [25](#) of file [World.cs](#).

6.68.3.3 chunks

```
Dictionary<ChunkWorldPos, Chunk> BeeGame.Terrain.LandGeneration.World.chunks = new Dictionary<Chunk<WorldPos, Chunk>()
```

All of the currently loaded chunks

Definition at line [20](#) of file [World.cs](#).

The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/LandGeneration/[World.cs](#)

7 File Documentation

7.1 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Air.cs File Reference

Classes

- class [BeeGame.Blocks.Air](#)
Air Block is an empty block that does not render and has no collider

Namespaces

- namespace [BeeGame.Blocks](#)

7.2 Air.cs

```

00001 using System;
00002 using BeeGame.Core.Enums;
00003 using BeeGame.Terrain.Chunks;
00004 using BeeGame.Core;
00005
00006 namespace BeeGame.Blocks
00007 {
00011     [Serializable]
00012     public class Air : Block
00013     {
00014         public new static int ID => 0;
00015
00016         public Air() : base("Air")
00017         {
00018         }
00019
00024         public override void BreakBlock(THVector3 pos)
00025         {
00026             return;
00027         }
00028
00033         public override MeshData BlockData(Chunk chunk, int x, int y, int z,
00034             MeshData meshData, bool addRoRenderMesh = true)
00035         {
00036             return meshData;
00037         }
00043         public override bool IsSolid(Direction direction)
00044         {
00045             return false;
00046         }
00047
00052         public override int GetHashCode()
00053         {
00054             return ID;
00055         }
00056
00061         public override string ToString()
00062         {
00063             return $"{itemName} \nID: {GetItemID()}";
00064         }
00065     }
00066 }

```

7.3 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Apiary.cs File Reference

Classes

- class [BeeGame.Blocks.Apiary](#)
Apiary Block

Namespaces

- namespace [BeeGame.Blocks](#)

7.4 Apiary.cs

```

00001 using System;
00002 using UnityEngine;
00003 using BeeGame.Core;
00004 using BeeGame.Items;
00005 using BeeGame.Quest;
00006 using BeeGame.Inventory;
00007 using BeeGame.Core.Enums;
00008 using BeeGame.Terrain.Chunks;
00009 using BeeGame.Core.Dictionaries;
00010

```

```

00011 namespace BeeGame.Blocks
00012 {
00016     [Serializable]
00017     public class Apiary : Block
00018     {
00019         [NonSerialized]
00020         private GameObject myGameObject;
00021
00022         public int mutationMultipler;
00023
00024         public new static int ID => 10;
00025
00026         #region Constructor
00027         public Apiary() : base("Apiary")
00028         {
00029             usesGameObject = true;
00030         }
00031         #endregion
00035
00036         #region Block Overrides
00037         public override GameObject GetGameObject()
00038         {
00039             return PrefabDictionary.GetPrefab("Apiary");
00040         }
00045
00054         public override Tile TexturePosition(Direction direction)
00055         {
00056             return new Tile() { x = 0, y = 9 };
00057         }
00058
00072         public override MeshData BlockData(Chunk chunk, int x, int y, int z,
00073         MeshData meshData, bool addToRenderMesh = true)
00074         {
00075             if (myGameObject == null)
00076             {
00077                 myGameObject = UnityEngine.Object.Instantiate(
00078                     PrefabDictionary.GetPrefab("Apiary"), new THVector3(x, y, z) + chunk.
00079                     chunkWorldPos, Quaternion.identity, chunk.transform);
00080                 myGameObject.GetComponent<ChestInventory>().inventoryPosition = new
00081                     THVector3(x, y, z) + chunk.chunkWorldPos;
00082                 myGameObject.GetComponent<ChestInventory>().SetChestInventory();
00083             }
00084             return base.BlockData(chunk, x, y, z, meshData, true);
00085         }
00087
00088         public override void BreakBlock(THVector3 pos)
00089         {
00090             /* removes the blocks blocks inventory save file and destroys the game object
00091             Serialization.Serialization.DeleteFile(myGameObject.GetComponent<
00092                 ApiaryInventory>().inventoryName);
00093             UnityEngine.Object.Destroy(myGameObject);
00094             /* removes the collision mesh from the chunk
00095             base.BreakBlock(pos);
00096
00097             public override Sprite GetItemSprite()
00098             {
00099                 return SpriteDictionary.GetSprite("Apiary");
00100             }
00101             #endregion
00102
00103             #region Overrides
00104             public override int GetHashCode()
00105             {
00106                 return ID;
00107             }
00108
00116             public override string ToString()
00117             {
00118                 return $"{itemName} \nID: {GetItemID()}";
00119             }
00120             #endregion
00121
00127             public override bool InteractWithBlock(Inventory.Inventory inv)
00128             {
00129                 myGameObject.GetComponent<ApiaryInventory>().myblock = this;
00130                 myGameObject.GetComponent<ApiaryInventory>().ToggleInventory(inv);
00131                 return true;
00132             }
00133
00134             #region Bee Combining Stuff
00135             public void MakeBees(Bee queen, ref Item[] inventory)
00136             {
00137                 Item[] producedItems = new Item[9];
00138
00139                 /* will always return a new princess and drone
00140                 producedItems[0] = MakeBee(BeeType.PRINCESS, queen.queenBee);
00141
00142
00143
00144
00145
00146
00147
00148
00149
00150
00151
00152
00153
00154
00155
00156
00157
00158
00159
00160
00161
00162
00163
00164
00165
00166
00167
00168
00169
00170
00171
00172
00173
00174
00175
00176
00177
00178
00179
00180
00181
00182
00183
00184
00185
00186
00187
00188
00189
00190
00191
00192
00193
00194
00195
00196
00197
00198
00199
00200
00201
00202
00203
00204
00205
00206
00207
00208
00209
00210
00211
00212
00213
00214
00215
00216
00217
00218
00219
00220
00221
00222
00223
00224
00225
00226
00227
00228
00229
00230
00231
00232
00233
00234
00235
00236
00237
00238
00239
00240
00241
00242
00243
00244
00245
00246
00247
00248
00249
00250
00251
00252
00253
00254
00255
00256
00257
00258
00259
00260
00261
00262
00263
00264
00265
00266
00267
00268
00269
00270
00271
00272
00273
00274
00275
00276
00277
00278
00279
00280
00281
00282
00283
00284
00285
00286
00287
00288
00289
00290
00291
00292
00293
00294
00295
00296
00297
00298
00299
00300
00301
00302
00303
00304
00305
00306
00307
00308
00309
00310
00311
00312
00313
00314
00315
00316
00317
00318
00319
00320
00321
00322
00323
00324
00325
00326
00327
00328
00329
00330
00331
00332
00333
00334
00335
00336
00337
00338
00339
00340
00341
00342
00343
00344
00345
00346
00347
00348
00349
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378
00379
00380
00381
00382
00383
00384
00385
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402
00403
00404
00405
00406
00407
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417
00418
00419
00420
00421
00422
00423
00424
00425
00426
00427
00428
00429
00430
00431
00432
00433
00434
00435
00436
00437
00438
00439
00440
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00778
00779
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00878
00879
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00978
00979
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009
01009
01010
01011
01012
01013
01014
01015
01016
01017
01018
01019
01019
01020
01021
01022
01023
01024
01025
01026
01027
01028
01029
01029
01030
01031
01032
01033
01034
01035
01036
01037
01038
01039
01039
01040
01041
01042
01043
01044
01045
01046
01047
01048
01049
01049
01050
01051
01052
01053
01054
01055
01056
01057
01058
01059
01059
01060
01061
01062
01063
01064
01065
01066
01067
01068
01069
01069
01070
01071
01072
01073
01074
01075
01076
01077
01078
01079
01079
01080
01081
01082
01083
01084
01085
01086
01087
01088
01089
01089
01090
01091
01092
01093
01094
01095
01096
01097
01098
01099
01099
01100
01101
01102
01103
01104
01105
01106
01107
01108
01109
01109
01110
01111
01112
01113
01114
01115
01116
01117
01118
01119
01119
01120
01121
01122
01123
01124
01125
01126
01127
01128
01129
01129
01130
01131
01132
01133
01134
01135
01136
01137
01138
01139
01139
01140
01141
01142
01143
01144
01145
01146
01147
01148
01149
01149
01150
01151
01152
01153
01154
01155
01156
01157
01158
01159
01159
01160
01161
01162
01163
01164
01165
01166
01167
01168
01169
01169
01170
01171
01172
01173
01174
01175
01176
01177
01178
01179
01179
01180
01181
01182
01183
01184
01185
01186
01187
01188
01189
01189
01190
01191
01192
01193
01194
01195
01196
01197
01198
01199
01199
01200
01201
01202
01203
01204
01205
01206
01207
01208
01209
01209
01210
01211
01212
01213
01214
01215
01216
01217
01218
01219
01219
01220
01221
01222
01223
01224
01225
01226
01227
01228
01229
01229
01230
01231
01232
01233
01234
01235
01236
01237
01238
01239
01239
01240
01241
01242
01243
01244
01245
01246
01247
01248
01249
01249
01250
01251
01252
01253
01254
01255
01256
01257
01258
01259
01259
01260
01261
01262
01263
01264
01265
01266
01267
01268
01269
01269
01270
01271
01272
01273
01274
01275
01276
01277
01278
01279
01279
01280
01281
01282
01283
01284
01285
01286
01287
01288
01289
01289
01290
01291
01292
01293
01294
01295
01296
01297
01298
01299
01299
01300
01301
01302
01303
01304
01305
01306
01307
01308
01309
01309
01310
01311
01312
01313
01314
01315
01316
01317
01318
01319
01319
01320
01321
01322
01323
01324
01325
01326
01327
01328
01329
01329
01330
01331
01332
01333
01334
01335
01336
01337
01338
01339
01339
01340
01341
01342
01343
01344
01345
01346
01347
01348
01349
01349
01350
01351
01352
01353
01354
01355
01356
01357
01358
01359
01359
01360
01361
01362
01363
01364
01365
01366
01367
01368
01369
01369
01370
01371
01372
01373
01374
01375
01376
01377
01378
01379
01379
01380
01381
01382
01383
01384
01385
01386
01387
01388
01389
01389
01390
01391
01392
01393
01394
01395
01396
01397
01398
01399
01399
01400
01401
01402
01403
01404
01405
01406
01407
01408
01409
01409
01410
01411
01412
01413
01414
01415
01416
01417
01418
01419
01419
01420
01421
01422
01423
01424
01425
01426
01427
01428
01429
01429
01430
01431
01432
01433
01434
01435
01436
01437
01438
01439
01439
01440
01441
01442
01443
01444
01445
01446
01447
01448
01449
01449
01450
01451
01452
01453
01454
01455
01456
01457
01458
01459
01459
01460
01461
01462
01463
01464
01465
01466
01467
01468
01469
01469
01470
01471
01472
01473
01474
01475
01476
01477
01478
01479
01479
01480
01481
01482
01483
01484
01485
01486
01487
01488
01489
01489
01490
01491
01492
01493
01494
01495
01496
01497
01498
01499
01499
01500
01501
01502
01503
01504
01505
01506
01507
01508
01509
01509
01510
01511
01512
01513
01514
01515
01516
01517
01518
01519
01519
01520
01521
01522
01523
01524
01525
01526
01527
01528
01529
01529
01530
01531
01532
01533
01534
01535
01536
01537
01538
01539
01539
01540
01541
01542
01543
01544
01545
01546
01547
01548
01549
01549
01550
01551
01552
01553
01554
01555
01556
01557
01558
01559
01559
01560
01561
01562
01563
01564
01565
01566
01567
01568
01569
01569
01570
01571
01572
01573
01574
01575
01576
01577
01578
01579
01579
01580
01581
01582
01583
01584
01585
01586
01587
01588
01589
01589
01590
01591
01592
01593
01594
01595
01596
01597
01598
01599
01599
01600
01601
01602
01603
01604
01605
01606
01607
01608
01609
01609
01610
01611
01612
01613
01614
01615
01616
01617
01618
01619
01619
01620
01621
01622
01623
01624
01625
01626
01627
01628
01629
01629
01630
01631
01632
01633
01634
01635
01636
01637
01638
01639
01639
01640
01641
01642
01643
01644
01645
01646
01647
01648
01649
01649
01650
01651
01652
01653
01654
01655
01656
01657
01658
01659
01659
01660
01661
01662
01663
01664
01665
01666
01667
01668
01669
01669
01670
01671
01672
01673
01674
01675
01676
01677
01678
01679
01679
01680
01681
01682
01683
01684
01685
01686
01687
01688
01689
01689
01690
01691
01692
01693
01694
01695
01696
01697
01698
01699
01699
01700
01701
01702
01703
01704
01705
01706
01707
01708
01709
01709
01710
01711
01712
01713
01714
01715
01716
01717
0
```

```

00149         producedItems[1] = MakeBee(BeeType.DRONE, queen.queenBee);
00150
00151     var repeats = UnityEngine.Random.Range(0, queen.queenBee.
00152     queen.pFertility);
00153
00154     /* produces as many other children as the bee stats will allow
00155     for (int i = 0; i < repeats; i++)
00156     {
00157         producedItems[i + 2] = MakeBee(queen.queenBee.queen.
00158         pFertility > 6 ? (BeeType)UnityEngine.Random.Range(1, 3) :
00159         BeeType.DRONE, queen.queenBee);
00160
00161         if (producedItems[i + 2] is Bee b && b.beeType !=
00162         BeeType.PRINCESS)
00163             producedItems[i + 2].itemStackCount =
00164             UnityEngine.Random.Range(1, (int)queen.queenBee.queen.
00165             pFertility + 1);
00166
00167         }
00168
00169         /* gets the produced items
00170         var beeProduce = BeeDictionaries.GetBeeProduce(queen.
00171         queenBee.queen.pSpecies);
00172
00173         /* changes the stack count of the produced items to the correct number
00174         for (int i = 0; i < beeProduce.Length; i++)
00175         {
00176             beeProduce[i].itemStackCount += UnityEngine.Random.Range(1, (int)
00177             queen.queenBee.queen.sProdSpeed + 1);
00178
00179         /* adds the items that the bee species produces into the produced item array
00180         for (int i = (int)queen.queenBee.queen.pFertility + 2, prod = 0; prod <
00181         beeProduce.Length; i++, prod++)
00182         {
00183             producedItems[i] = beeProduce[prod];
00184
00185             /* puts the items into the inventory
00186             for (int i = 0; i < 9; i++)
00187             {
00188                 if (inventory[i + 2] != null)
00189                 {
00190                     /* if the slot has the same item in it and it won't be more than the max stack count
00191                     /* but the new item into it
00192                     if (producedItems[i] == inventory[i + 2] && inventory[i + 2].itemStackCount + 1 <=
00193                     inventory[i + 2].maxStackCount)
00194                         inventory[i + 2].itemStackCount++;
00195                     else
00196                         /* otherwise find a new slot to put the item into
00197                         for (int j = i; j < (9 - i); j++)
00198                         {
00199                             if (inventory[j + 2] == null)
00200                             {
00201                                 inventory[j + 2] = producedItems[i];
00202                                 break;
00203                             }
00204                             else if (producedItems[i] == inventory[j + 2] && inventory[j + 2].
00205                             itemStackCount + 1 <= inventory[j + 2].maxStackCount)
00206                                 {
00207                                     inventory[j + 2].itemStackCount++;
00208                                     break;
00209                                 }
00210                         }
00211                     /* if the slot is empty put the item into it
00212                     else
00213                         inventory[i + 2] = producedItems[i];
00214
00215         public Bee MakeBee(BeeType beeType, QueenBee queen)
00216         {
00217             /* gives all of the primary and secondary stats to the bee
00218             NormalBee nb = new NormalBee()
00219             {
00220                 pSpecies = CombineSpecies(queen.queen.sSpecies, queen.
00221                 drone.sSpecies),
00222                 sSpecies = CombineSpecies(queen.queen.sSpecies, queen.
00223                 drone.sSpecies),
00224                 pEffect = CombineEffect(queen.queen.sEffect, queen.
00225                 drone.sEffect),
00226                 sEffect = CombineEffect(queen.queen.sEffect, queen.
00227                 drone.sEffect),
00228                 pFertility = CombineFertility(queen.queen.sFertility, queen.
00229                 drone.sFertility),
00230             }
00231         }
00232     }
00233 }
```

```
00225             sFertility = CombineFertility(queen.queen.sFertility, queen.
00226             drone.sFertility),
00227             pLifespan = CombineLifespan(queen.queen.sLifespan, queen.
00228             drone.sLifespan),
00229             sLifespan = CombineLifespan(queen.queen.sLifespan, queen.
00230             drone.sLifespan),
00231             pProdSpeed = CombineProductionSpeed(queen.queen.sProdSpeed, queen.
00232             drone.sProdSpeed),
00233             sProdSpeed = CombineProductionSpeed(queen.queen.sProdSpeed, queen.
00234             drone.sProdSpeed)
00235         };
00236         //QuestEvents.CallBeeCraftedEvent(nb.pSpecies);
00237         /* returns the new bee
00238         return new Bee(beeType, nb);
00239     }
00240
00241     private BeeSpecies CombineSpecies(BeeSpecies s1,
00242     BeeSpecies s2)
00243     {
00244         BeeSpecies[] possibleSpecies = BeeDictionaries.
00245         GetCombinations(s1, s2);
00246         float[] weights = possibleSpecies.Length > 2 ? BeeDictionaries.
00247         GetWeights(possibleSpecies) : new float[] { 0.5f, 0.5f };
00248
00249         var randomNum = Rand(weights);
00250         var weightsSum = 0f;
00251
00252         /* when the number generated is less than the current sum of the weights return that bee
00253         for (int i = 0; i < weights.Length; i++)
00254         {
00255             if(randomNum <= weightsSum)
00256             {
00257                 return possibleSpecies[i];
00258             }
00259
00260             weightsSum += weights[i];
00261         }
00262
00263         /* if for some reason the weights cannot work return the first bee in the combination list
00264         return possibleSpecies[0];
00265     }
00266
00267     private float Rand(float[] weights)
00268     {
00269         var totalWeights = 0f;
00270
00271         /* sums the weights
00272         for (int i = 0; i < weights.Length; i++)
00273         {
00274             totalWeights += weights[i];
00275         }
00276
00277         return (float)Math.Round(UnityEngine.Random.Range(0, totalWeights), 2);
00278     }
00279
00280     private BeeLifeSpan CombineLifespan(
00281     BeeLifeSpan b1, BeeLifeSpan b2)
00282     {
00283         return (BeeLifeSpan)ReturnChange((int)b1, (int)b2, (int)
00284         BeeLifeSpan.SEATURTLE);
00285     }
00286
00287     private uint CombineFertility(uint b1, uint b2)
00288     {
00289         return (uint)ReturnChange((int)b1, (int)b2, 5, 1);
00290     }
00291
00292     private BeeEffect CombineEffect(BeeEffect b1,
00293     BeeEffect b2)
00294     {
00295         return (BeeEffect)ReturnChange((int)b1, (int)b2, (int)
00296         BeeEffect.POISON);
00297     }
00298
00299     public BeeProductionSpeed CombineProductionSpeed(
00300     BeeProductionSpeed b1, BeeProductionSpeed b2)
00301     {
00302         return (BeeProductionSpeed)ReturnChange((int)b1, (int)b2, (int)
00303         BeeProductionSpeed.FAST);
00304     }
00305
00306     private int ReturnChange(int b1, int b2, int maxChange, int minChange = 0)
00307     {
```

```

00344     /* b1 and b2 are checked for which one is larger here as the
00345     /* queen may have a lower stat than the drone as the drone is always passed in second
00346     var change = UnityEngine.Random.Range(b1 < b2 ? b1 : b2, (b2 > b1 ? b2 : b1) + 2);
00347
00348     /* this will make it possible for the bees to mutate during combination of the stats are the
00349     /* same
00350     // it will also cause more random mutation more mimicking nature
00351     change += UnityEngine.Random.Range(-mutationMultipler, mutationMultipler);
00352
00353     /* as all of the stats are enums they have a min/max value so need to check that this is not
00354     exceeded
00355     if (change > maxChange)
00356         change = maxChange;
00357     else if (minChange > change)
00358         change = minChange;
00359
00360     return change;
00361 }
00362 #endregion
00363 }
```

7.5 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Bedrock.cs File Reference

Classes

- class BeeGame.Blocks.Bedrock
Bedrock Block

Namespaces

- namespace BeeGame.Blocks

7.6 Bedrock.cs

```

00001 using System;
00002 using BeeGame.Core.Enums;
00003 using BeeGame.Items;
00004 using BeeGame.Core;
00005
00006 namespace BeeGame.Blocks
00007 {
00008     [Serializable]
00009     public class Bedrock : Block
00010     {
00011         #region Data
00012         public new static int ID => -1;
00013         #endregion
00014
00015         #region Constructor
00016         public Bedrock() : base("Bedrock")
00017         {
00018             breakable = false;
00019         }
00020         #endregion
00021
00022         #region Break Block
00023         public override void BreakBlock(THVector3 pos)
00024         {
00025             return;
00026         }
00027         #endregion
00028
00029         #region Mesh
00030         public override Tile TexturePosition(Direction direction)
00031         {
00032             return new Tile() { x = 0, y = 0};
00033         }
00034         #endregion
00035 }
```

```

00051     #region Overrides
00052     public override int GetHashCode()
00053     {
00054         return ID;
00055     }
00056
00057     public override string ToString()
00058     {
00059         return $"{itemName} \nID: {GetItemID()}";
00060     }
00061     #endregion
00062 }
00063 }
```

7.7 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Block.cs File Reference

Classes

- class [BeeGame.Blocks.Block](#)

Base class for blocks

Namespaces

- namespace [BeeGame.Blocks](#)

7.8 Block.cs

```

00001 using UnityEngine;
00002 using BeeGame.Terrain.Chunks;
00003 using BeeGame.Core.Enums;
00004 using BeeGame.Items;
00005 using BeeGame.Core;
00006 using BeeGame.Core.Dictionaries;
00007
00008 namespace BeeGame.Blocks
00009 {
00010     [System.Serializable]
00011     public class Block : Item
00012     {
00013         #region Data
00014         public new static int ID = 1;
00015         public bool breakable = true;
00016         public bool changed = true;
00017         public override bool placeable => true;
00018         #endregion
00019
00020         #region Constructor
00021         public Block() : base()
00022         {
00023             itemName = "Stone";
00024         }
00025
00026         public Block(string name) : base(name)
00027         {
00028         }
00029         #endregion
00030
00031         #region Item Stuff
00032         public override Sprite GetItemSprite()
00033         {
00034             return SpriteDictionary.GetSprite("Stone");
00035         }
00036         #endregion
00037
00038         #region Update/Break Block
00039         public virtual void BreakBlock(THVector3 pos)
00040         {
00041             GameObject go = Object.Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject, pos, Quaternion.identity) as GameObject;
00042             go.GetComponent<ItemGameObject>().item = this;
00043         }
00044     }
```

```

00067
00075     public virtual void UpdateBlock(int x, int y, int z, Chunk chunk) { }
00076
00081     public virtual bool InteractWithBlock(BeeGame.
00082         Inventory.Inventory inv)
00083     {
00084         return false;
00085     }
00086 #endregion
00087
00088     #region Mesh
00089     public virtual MeshData BlockData(Chunk chunk, int x, int y, int z,
00090         MeshData meshData, bool addToRenderMesh = true)
00091     {
00092         /* Adds the Top face of the block
00093         if (!chunk.GetBlock(x, y + 1, z, false).IsSolid(Direction.DOWN))
00094         {
00095             meshData = FaceDataUp(x, y, z, meshData, addToRenderMesh);
00096         }
00097
00098         /* Adds the Bottom face of the block
00099         if (!chunk.GetBlock(x, y - 1, z, false).IsSolid(Direction.UP))
00100         {
00101             meshData = FaceDataDown(x, y, z, meshData, addToRenderMesh);
00102         }
00103
00104         /* Adds the North face of the block
00105         if (!chunk.GetBlock(x, y, z + 1, false).IsSolid(Direction.SOUTH))
00106         {
00107             meshData = FaceDataNorth(x, y, z, meshData, addToRenderMesh);
00108         }
00109
00110         /* Adds the South face of the block
00111         if (!chunk.GetBlock(x, y, z - 1, false).IsSolid(Direction.NORTH))
00112         {
00113             meshData = FaceDataSouth(x, y, z, meshData, addToRenderMesh);
00114         }
00115
00116         /* Adds the East face of the block
00117         if (!chunk.GetBlock(x + 1, y, z, false).IsSolid(Direction.WEST))
00118         {
00119             meshData = FaceDataEast(x, y, z, meshData, addToRenderMesh);
00120         }
00121
00122         /* Adds the West face of the block
00123         if (!chunk.GetBlock(x - 1, y, z, false).IsSolid(Direction.EAST))
00124         {
00125             meshData = FaceDataWest(x, y, z, meshData, addToRenderMesh);
00126         }
00127
00128         return meshData;
00129     }
00130
00131     public virtual bool IsSolid(Direction direction)
00132     {
00133         return true;
00134     }
00135 #endregion
00136
00137     #region Overrides
00138     public override int GetHashCode()
00139     {
00140         return ID;
00141     }
00142
00143     public override string ToString()
00144     {
00145         return $"{itemName} \nID: {GetHashCode()}";
00146     }
00147 #endregion
00148 }
00149 }
```

7.9 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Chest.cs File Reference

Classes

- class BeeGame.Blocks.Chest
Chest Block

Namespaces

- namespace BeeGame.Blocks

7.10 Chest.cs

```

00001 using System;
00002 using UnityEngine;
00003 using BeeGame.Core;
00004 using BeeGame.Terrain.Chunks;
00005 using BeeGame.Core.Enums;
00006 using BeeGame.Items;
00007 using BeeGame.Inventory;
00008 using BeeGame.Core.Dictionaries;
00009
00010 namespace BeeGame.Blocks
00011 {
00015     [Serializable]
00016     public class Chest : Block
00017     {
00018         #region Data
00019         [NonSerialized]
00023         private GameObject myGameObject;
00024
00025         public new static int ID => 8;
00026         #endregion
00027
00028         #region Constructors
00029         public Chest() : base("Chest")
00033         {
00034             usesGameObject = true;
00035         }
00036         #endregion
00037
00038         #region Block Overrides
00039         public override GameObject GetGameObject()
00044         {
00045             return PrefabDictionary.GetPrefab("Chest");
00046         }
00047
00056         public override Tile TexturePosition(Direction direction)
00057         {
00058             return new Tile() { x = 0, y = 9 };
00059         }
00060
00074         public override MeshData BlockData(Chunk chunk, int x, int y, int z,
00075             MeshData meshData, bool addToRenderMesh = true)
00076         {
00077             if (myGameObject == null)
00078             {
00079                 myGameObject = UnityEngine.Object.Instantiate(
00080                     PrefabDictionary.GetPrefab("Chest"), new THVector3(x, y, z) + chunk.
00081                     chunkWorldPos, Quaternion.identity, chunk.transform);
00082                 myGameObject.GetComponent<ChestInventory>().inventoryPosition = new
00083                     THVector3(x, y, z) + chunk.chunkWorldPos;
00084                 myGameObject.GetComponent<ChestInventory>().SetChestInventory();
00085             }
00086             return base.BlockData(chunk, x, y, z, meshData, true);
00087         }
00088
00089         public override void BreakBlock(THVector3 pos)
00090         {
00091             /* removes the blocks blocks inventory save file and destroys the game object
00092             Serialization.Serialization.DeleteFile(myGameObject.GetComponent<
00093                 ChestInventory>().inventoryName);
00094             UnityEngine.Object.Destroy(myGameObject);
00095             /* removes the collision mesh from the chunk
00096             base.BreakBlock(pos);
00097         }
00098
00102         public override Sprite GetItemSprite()
00103         {
00104             return SpriteDictionary.GetSprite("Chest");
00105         }
00106         #endregion
00107
00108         #region Inventory Suff
00109         public override bool InteractWithBlock(BeeGame.Inventory.
00110             Inventory inv)
00115         {
00116             myGameObject.GetComponent<ChestInventory>().ToggleInventory(inv);
00117             return true;

```

```

00118     }
00119     #endregion
00120
00121     #region Overrides
00122     public override int GetHashCode()
00123     {
00124         return ID;
00125     }
00126
00127     public override string ToString()
00128     {
00129         return $"{itemName}\nID{GetItemID()}";
00130     }
00131     #endregion
00132 }
00133 }
```

7.11 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/CraftingTable.cs

File Reference

Classes

- class [BeeGame.Blocks.CraftingTable](#)
The Workbench Block class

Namespaces

- namespace [BeeGame.Blocks](#)

7.12 CraftingTable.cs

```

00001 using System;
00002 using UnityEngine;
00003 using BeeGame.Core;
00004 using BeeGame.Items;
00005 using BeeGame.Core.Enums;
00006 using BeeGame.Terrain.Chunks;
00007 using BeeGame.Core.Dictionaries;
00008
00009 namespace BeeGame.Blocks
00010 {
00011     [Serializable]
00012     public class CraftingTable : Block
00013     {
00014         #region Data
00015         [NonSerialized]
00016         private GameObject myGameObject;
00017
00018         public new static int ID => 9;
00019         #endregion
00020
00021         #region Constructor
00022         public CraftingTable() : base("Workbench")
00023         {
00024             usesGameObject = true;
00025         }
00026         #endregion
00027
00028         #region Crafting
00029         public Item ReturnShapedRecipieItem(Item[] items)
00030         {
00031             var recipe = "";
00032
00033             for (int i = 0; i < items.Length; i++)
00034             {
00035                 if (items[i] == null)
00036                 {
00037                     recipe += "0:";
00038                     continue;
00039                 }
00040
00041                 recipe += $"{items[i].GetItemID()}:";
```

```

00059         }
00060
00061     return ReturnShapedRecipieItem(recipe);
00062 }
00063
00064     public virtual Item ReturnShapelessRecipieItem(
00065     Item[] items)
00066     {
00067         return CraftingRecipies.GetShaplessRecipieResult(items)
00068     }
00069
00070     public virtual Item ReturnShapedRecipieItem(string recipe)
00071     {
00072         return CraftingRecipies.GetShapedRecipeItem(recipe);
00073     }
00074 #endregion
00075
00076 #region Block Overrides
00077     public override bool InteractWithBlock(Inventory.Inventory inv)
00078     {
00079         myGameObject.GetComponent<Inventory.BlockInventory.CraftingTableInventory>().myblock = this;
00080         myGameObject.GetComponent<Inventory.BlockInventory.CraftingTableInventory>().ToggleInventory(
00081             inv);
00082         return true;
00083     }
00084
00085
00086     public override GameObject GetGameObject()
00087     {
00088         return PrefabDictionary.GetPrefab("CraftingTable");
00089     }
00090
00091
00092     public override MeshData BlockData(Chunk chunk, int x, int y, int z,
00093     MeshData meshData, bool addToRenderMesh = true)
00094     {
00095         if (myGameObject == null)
00096         {
00097             myGameObject = UnityEngine.Object.Instantiate(
00098                 PrefabDictionary.GetPrefab("CraftingTable"), new
00099                 THVector3(x, y, z) + chunk.chunkWorldPos, Quaternion.identity, chunk.transform);
00100         }
00101         return base.BlockData(chunk, x, y, z, meshData, true);
00102     }
00103
00104
00105     public override bool IsSolid(Direction direction)
00106     {
00107         return false;
00108     }
00109
00110
00111     public override void BreakBlock(THVector3 pos)
00112     {
00113         /* removes the game object
00114         UnityEngine.Object.Destroy(myGameObject);
00115         /* removes the collision mesh from the chunk
00116         base.BreakBlock(pos);
00117     }
00118
00119
00120     public override Sprite GetItemSprite()
00121     {
00122         return SpriteDictionary.GetSprite("CraftingTable");
00123     }
00124
00125
00126     public override Tile TexturePosition(Direction direction)
00127     {
00128         return new Tile() { x = 0, y = 9 };
00129     }
00130 #endregion
00131
00132
00133 #region Overrides
00134     public override int GetHashCode()
00135     {
00136         return ID;
00137     }
00138 #endregion
00139 }
00140
00141 }
```

7.13 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Dirt.cs File Reference

Classes

- class BeeGame.Blocks.Dirt

Dirt Block

Namespaces

- namespace BeeGame.Blocks

7.14 Dirt.cs

```

00001 using System;
00002 using BeeGame.Core.Enums;
00003 using BeeGame.Items;
00004 using BeeGame.Core.Dictionaries;
00005 using UnityEngine;
00006
00007 namespace BeeGame.Blocks
00008 {
00012     [Serializable]
00013     public class Dirt : Block
00014     {
00015         public new static int ID => 3;
00016
00017         #region Constructor
00018         public Dirt() : base("Dirt"){}
00022         #endregion
00023
00024         #region Item Stuff
00025         public override Sprite GetItemSprite()
00026         {
00027             return SpriteDictionary.GetSprite("Dirt");
00028         }
00029         #endregion
00030
00031         #region Mesh
00032         public override Tile TexturePosition(Direction direction)
00038         {
00039             return new Tile { x = 2, y = 9 };
00040         }
00041         #endregion
00042
00043         #region Overrides
00044         public override int GetHashCode()
00049         {
00050             return ID;
00051         }
00052
00057         public override string ToString()
00058         {
00059             return $"{itemName} \nID: {GetItemID()}";
00060         }
00061         #endregion
00062     }
00063 }
```

7.15 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Grass.cs File Reference

Classes

- class BeeGame.Blocks.Grass

Grass Block

Namespaces

- namespace BeeGame.Blocks

7.16 Grass.cs

```

00001 using System;
00002 using UnityEngine;
00003 using BeeGame.Core.Enums;
00004 using BeeGame.Terrain.Chunks;
00005 using BeeGame.Core.Dictionaries;
00006 using BeeGame.Items;
00007
00008 namespace BeeGame.Blocks
00009 {
00013     [Serializable]
00014     public class Grass : Block
00015     {
00016         public new static int ID => 4;
00017
00018         #region Constructor
00019         public Grass() : base("Grass") {}
00023         #endregion
00024
00025         #region Item Stuff
00026         public override Sprite GetItemSprite()
00027         {
00028             return SpriteDictionary.GetSprite("Grass");
00029         }
00030         #endregion
00031
00032         #region Mesh
00033         public override void UpdateBlock(int x, int y, int z, Chunk chunk)
00041         {
00042             if (chunk.GetBlock(x, y + 1, z, false).IsSolid(Direction.DOWN))
00043                 chunk.blocks[x, y, z] = new Dirt() { changed = changed };
00044         }
00045
00051         public override Tile TexturePosition(Direction direction)
00052         {
00053             //All textures are on the same Y value for the texture atlas so Y can be set
00054             Tile tile = new Tile()
00055             {
00056                 y = 9
00057             };
00059             switch (direction)
00060             {
00061                 //if we want the top face return the full grass texture
00062                 case Direction.UP:
00063                     tile.x = 3;
00064                     return tile;
00065                 //if we want the bottom face return the dirt texture
00066                 case Direction.DOWN:
00067                     tile.x = 2;
00068                     return tile;
00069                 //return the 1/2 grass texture if a side face is wanted
00070                 default:
00071                     tile.x = 4;
00072                     return tile;
00073             }
00074         }
00075         #endregion
00076
00077         #region Overrides
00078         public override int GetHashCode()
00083         {
00084             return ID;
00085         }
00086
00091         public override string ToString()
00092         {
00093             return $"{itemName} \nID: {GetItemID()}";
00094         }
00095         #endregion
00096     }
00097 }

```

7.17 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Leaves.cs File Reference

Classes

- class BeeGame.Blocks.Leaves

Namespaces

- namespace BeeGame.Blocks

7.18 Leaves.cs

```

00001 using System;
00002 using UnityEngine;
00003 using BeeGame.Core.Dictionaries;
00004 using BeeGame.Core.Enums;
00005 using BeeGame.Items;
00006
00007 namespace BeeGame.Blocks
00008 {
00009     [Serializable]
00010     public class Leaves : Block
00011     {
00012         public new static int ID => 6;
00013
00014         public Leaves() : base("Leaves")
00015         {
00016
00017         }
00018
00019         #region Item Stuff
00020         public override Sprite GetItemSprite()
00021         {
00022             return SpriteDictionary.GetSprite("Leaves");
00023         }
00024         #endregion
00025
00026         public override Tile TexturePosition(Direction direction)
00027         {
00028             return new Tile() { x = 5, y = 9 };
00029         }
00030
00031         public override bool IsSolid(Direction direction)
00032         {
00033             return false;
00034         }
00035
00036         #region Overrides
00037         public override int GetHashCode()
00038         {
00039             return ID;
00040         }
00041
00042         public override string ToString()
00043         {
00044             return $"{itemName} \nID: {GetItemID()}";
00045         }
00046         #endregion
00047     }
00048 }
```

7.19 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Planks.cs File Reference

Classes

- class BeeGame.Blocks.Planks

Planks Block

Namespaces

- namespace BeeGame.Blocks

7.20 Planks.cs

```

00001 using System;
00002 using BeeGame.Core.Enums;
00003 using BeeGame.Items;
00004 using BeeGame.Core.Dictionaries;
00005 using UnityEngine;
00006
00007 namespace BeeGame.Blocks
00008 {
00012     [Serializable]
00013     public class Planks : Block
00014     {
00015         public new static int ID => 7;
00016
00017         #region Constructor
00018         public Planks() : base("Planks") {}
00022         #endregion
00023
00024         #region Item Stuff
00025         public override Sprite GetItemSprite()
00026         {
00027             return SpriteDictionary.GetSprite("Planks");
00028         }
00029         #endregion
00030
00031         #region Mesh
00032         public override Tile TexturePosition(Direction direction)
00038         {
00039             return new Tile { x = 2, y = 9 };
00040         }
00041         #endregion
00042
00043         #region Overrides
00044         public override int GetHashCode()
00049         {
00050             return ID;
00051         }
00052
00057         public override string ToString()
00058         {
00059             return $"{itemName} \nID: {GetItemID()}";
00060         }
00061         #endregion
00062     }
00063 }
```

7.21 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Wood.cs File Reference

Classes

- class BeeGame.Blocks.Wood

Namespaces

- namespace BeeGame.Blocks

7.22 Wood.cs

```

00001 using System;
00002 using UnityEngine;
00003 using System.Collections.Generic;
00004 using System.Linq;
00005 using System.Text;
00006 using BeeGame.Core.Dictionaries;
00007 using BeeGame.Core.Enums;
00008 using BeeGame.Items;
00009
00010 namespace BeeGame.Blocks
00011 {
```

```

00012     [Serializable]
00013     public class Wood : Block
00014     {
00015         public new static int ID => 5;
00016
00017         public Wood() : base("Wood")
00018     {
00019
00020     }
00021
00022     #region Item Stuff
00023     public override Sprite GetItemSprite()
00024     {
00025         return SpriteDictionary.GetSprite("Wood");
00026     }
00027     #endregion
00028
00029     public override Tile TexturePosition(Direction direction)
00030     {
00031         return new Tile() { x = 7, y = 9 };
00032     }
00033
00034     #region Overrides
00035     public override int GetHashCode()
00036     {
00037         return ID;
00038     }
00039
00040     public override string ToString()
00041     {
00042         return $"{itemName} \nID: {GetItemID()}";
00043     }
00044     #endregion
00045 }
00046 }
```

7.23 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/BeeDictionaries.cs File Reference

Classes

- class BeeGame.Core.Dictionarys.BeeDictionarys

Namespaces

- namespace BeeGame.Core.Dictionarys

7.24 BeeDictionarys.cs

```

00001 using System.Collections.Generic;
00002 using System.Linq;
00003 using BeeGame.Core.Enums;
00004 using UnityEngine;
00005
00006 namespace BeeGame.Core.Dictionarys
00007 {
00008     public static class BeeDictionarys
00009     {
00010         #region Bee Combination Weights
00011         private static Dictionary<BeeSpecies, float> beeCombinationWeights = new Dictionary<BeeSpecies,
00012             float>()
00013         {
00014             {BeeSpecies.COMMON, 0.15f },
00015             {BeeSpecies.HEROIC, 0.06f }
00016         };
00017         public static float[] GetWeights(BeeSpecies[] species)
00018         {
00019             var returnArray = new float[species.Length];
00020             for (int i = 0; i < species.Length; i++)
00021             {
00022                 returnArray[i] = beeCombinationWeights[species[i]];
00023             }
00024         }
00025     }
00026 }
```

```

00023             if(beeCombinationWeights.ContainsKey(species[i]))
00024                 returnArray[i] = beeCombinationWeights[species[i]];
00025             else
00026                 returnArray[i] = 0.5f;
00027         }
00028
00029     return returnArray;
00030 }
00031 #endregion
00032
00033 #region Bee Combinations
00034 public static Dictionary<BeeSpecies[], BeeSpecies[]> beeCombinations = new Dictionary<BeeSpecies[], BeeSpecies[]>(new BeeCombinationDictionaryEqualityComparer())
00035 {
00036     { new BeeSpecies[6] { BeeSpecies.FOREST,
00037         BeeSpecies.MEADOWS, BeeSpecies.TROPICAL, BeeSpecies.WINTRY,
00038         BeeSpecies.MODEST, BeeSpecies.MARSHY }, new BeeSpecies[1] {
00039             BeeSpecies.COMMON } }
00040 };
00041
00042     public static BeeSpecies[] GetCombinations(
00043         BeeSpecies s1, BeeSpecies s2)
00044     {
00045         var beeSpecies = new BeeSpecies[2] { s1, s2 };
00046         var returnBeeList = new List<BeeSpecies>();
00047
00048         var keys = beeCombinations.Keys.ToArray();
00049         var comparor = new BeeCombinationDictionaryEqualityComparer
00050     ();
00051
00052         for (int i = 0; i < keys.Length; i++)
00053         {
00054             if(comparor.Equals(keys[i], beeSpecies))
00055             {
00056                 var temp = beeCombinations[keys[i]];
00057
00058                 for (int j = 0; j < temp.Length; j++)
00059                 {
00060                     returnBeeList.Add(temp[i]);
00061                 }
00062             }
00063
00064         }
00065     #endregion
00066
00067     #region Bee Produce
00068     private static Dictionary<BeeSpecies, Items.Item[]> beeProduce = new Dictionary<
00069         BeeSpecies, Items.Item[]>()
00070     {
00071         {BeeSpecies.FOREST, new Items.Item[]{new Items.HoneyComb(
00072             HoneyCombType.HONEY) } },
00073         {BeeSpecies.COMMON, new Items.Item[]{new Items.HoneyComb(
00074             HoneyCombType.HONEY) } }
00075     };
00076
00077     public static Items.Item[] GetBeeProduce(BeeSpecies species)
00078     {
00079         beeProduce.TryGetValue(species, out Items.Item[] produce);
00080
00081         /* if the produce can't be found, then return a honey comb as it is probably a bug
00082         return produce ?? new Items.Item[1] { new Items.HoneyComb(
00083             HoneyCombType.HONEY) }; */
00084     }
00085     #endregion
00086
00087     #region Bee Colours
00088     private static Dictionary<BeeSpecies, Color> beeColour = new Dictionary<BeeSpecies, Color>()
00089     {
00090         {BeeSpecies.FOREST, CombColour(0, 255, 0) },
00091         {BeeSpecies.COMMON, CombColour(255, 0, 0) }
00092     };
00093
00094     public static Color GetBeeColour(BeeSpecies species)
00095     {
00096         beeColour.TryGetValue(species, out Color colour);
00097
00098         return colour != null ? colour : new Color();
00099     }
00100 #endregion
00101
00102 #region Comb Colours
00103 private static Dictionary<HoneyCombType, Color> honeyCombColour = new Dictionary<HoneyCombType,

```

```

        Color>()
00103    {
00104        {HoneyCombType.HONEY, CombColour(255, 164, 56) },
00105        {HoneyCombType.ICEY, CombColour(78, 231, 231) }
00106    };
00107
00117    private static Color CombColour(float r, float g, float b, float a = 255f)
00118    {
00119        return new Color(r / 255f, g / 255f, b / 255f);
00120    }
00121
00127    public static Color GetCombColour(HoneyCombType type)
00128    {
00129        honeyCombColour.TryGetValue(type, out var temp);
00130
00131        if (temp == null)
00132            return new Color(1, 0, 0);
00133
00134        return temp;
00135    }
00136    #endregion
00137}
00138}

```

7.25 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/← CraftingRecipies.cs File Reference

Classes

- class BeeGame.Core.Dictionaries.CraftingRecipies

Namespaces

- namespace BeeGame.Core.Dictionaries

7.26 CraftingRecipies.cs

```

00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using System.Text;
00005 using BeeGame.Items;
00006 using BeeGame.Exceptions;
00007
00008 namespace BeeGame.Core.Dictionaries
00009 {
00010     public static class CraftingRecipies
00011     {
00012         #region Shaped Crafting
00013         private static Dictionary<string, Item> shapedCraftingRecipies = new Dictionary<string, Item>();
00017
00032         public static void AddShapedRecipie(object[] recipie,
00033             Item result)
00034         {
00035             /* converts the given blocks of 3 characters to a 9 character string
00036             var stringRecipie = "";
00037
00038             for (int i = 0; i < 3; i++)
00039             {
00040                 stringRecipie += recipie[i] as string;
00041
00042             /* gets what character represents which item
00043             for (int i = 3; i < recipie.Length; i += 2)
00044             {
00045                 var character = (string)recipie[i];
00046                 var itemID = (int)recipie[i + 1];
00047
00048                 /* replaces the character with the items id
00049                 stringRecipie = stringRecipie.Replace(character, $"{itemID.ToString()}:");
00050             }
00051

```

```
00052     /* converts empty sets " " into "0:" */
00053     stringRecipie = stringRecipie.Replace(" ", "0:");
00054
00055     /* if the recipe exists an exception is thrown as two recipies cannot be the same
00056     if (shapedCraftingRecipies.ContainsKey(stringRecipie))
00057         throw new CraftingRecipeAdditionException($"Shaped Recipe
already exists: {stringRecipie}");
00058
00059     result.itemStackCount = 1;
00060
00061     /* adds the recipe to the dictionary
00062     shapedCraftingRecipies.Add(stringRecipie, result);
00063 }
00064
00065 public static Item GetShapedRecipeItem(string recipe)
00066 {
00067     shapedCraftingRecipies.TryGetValue(recipe, out var item);
00068
00069     if (item != null)
00070         item.itemStackCount = 1;
00071
00072     return item;
00073 }
#endregion
00074
00075 #endregion
00076
00077 #region Shapless Crafting
00078 private static Dictionary<string, Item> shaplessRecipies = new Dictionary<string, Item>()
00079 {
00080     {"5:", new Blocks.Planks() }
00081 };
00082
00083 public static void AddShaplessRecipie(object[] recipe,
00084 Item result)
00085 {
00086     var itemList = new List<int>();
00087     var stringRecipie = "";
00088
00089     for (int i = 0; i < recipe.Length; i+=2)
00090     {
00091         for (int j = 0; j < (int)recipe[i+1]; j++)
00092         {
00093             itemList.Add(int.Parse(((Item)recipe[i]).GetItemID()));
00094         }
00095     }
00096
00097     itemList.Sort();
00098
00099     for (int i = 0; i < itemList.Count; i++)
00100     {
00101         stringRecipie += $"{itemList[i]}:";
00102     }
00103
00104     if (shaplessRecipies.ContainsKey(stringRecipie))
00105         throw new CraftingRecipeAdditionException($"Shaped Recipe
already exists: {stringRecipie}");
00106
00107     result.itemStackCount = 1;
00108     shaplessRecipies.Add(stringRecipie, result);
00109 }
00110
00111 public static string GetShaplessRecipieString(
00112 Item[] recipe)
00113 {
00114     var IDList = new List<int>();
00115     var stringRecipie = "";
00116
00117     /* converts the given item list to an ID list so it can be sorted
00118     for (int i = 0; i < recipe.Length; i++)
00119     {
00120         if(recipe[i] != null)
00121             IDList.Add(recipe[i].GetHashCode());
00122     }
00123
00124     IDList.Sort();
00125
00126     /* converts the sorted ID list to a string so it can be used as a dictionary key
00127     for (int i = 0; i < IDList.Count; i++)
00128     {
00129         /* : after each ID as it is possible for ID clashes without eg ID: 11 can be seen as 2 *
00130         ID: 1
00131             stringRecipie += $"{IDList[i]}:";
00132     }
00133
00134     return stringRecipie;
00135 }
00136
00137 public static Item GetShaplessRecipieResult(int[] recipe)
```

```

00173     {
00174         var list = recipe.ToList();
00175         list.Sort();
00176
00177         var stringRecipe = "";
00178
00179         for (int i = 0; i < list.Count; i++)
00180         {
00181             stringRecipe += $"{list[i]}:";
00182         }
00183
00184         return GetShaplessRecipieResult(stringRecipe);
00185     }
00186
00187     public static Item GetShaplessRecipieResult(string recipe)
00188     {
00189         shaplessRecipes.TryGetValue(recipe, out var item);
00190
00191         return item;
00192     }
00193
00194     public static Item GetShaplessRecipieResult(
00195         Item[] recipe)
00196     {
00197         shaplessRecipes.TryGetValue(GetShaplessRecipieString(recipe), out var item);
00198
00199         if (item != null)
00200             item.itemStackCount = 1;
00201
00202         return item;
00203     }
00204 #endregion
00205 }
00206 }
```

7.27 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/EqualityComperors.cs File Reference

Classes

- class BeeGame.Core.Dictionaries.BeeCombinationDictionaryEqualityComparer

Namespaces

- namespace BeeGame.Core.Dictionaries

7.28 EqualityComperors.cs

```

00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using System.Text;
00005 using BeeGame.Core.Enums;
00006
00007 namespace BeeGame.Core.Dictionaries
00008 {
00009     public class BeeCombinationDictionaryEqualityComparer :
00010         IEqualityComparer<BeeSpecies[]>
00011     {
00012         public bool Equals(BeeSpecies[] x, BeeSpecies[] y)
00013         {
00014             if (x.Contains(y[0]) && x.Contains(y[1]))
00015             {
00016                 /* if the x length is greater than 2 this means that the combination can have duplicate
00017                 bees for a product
00018                 if (x.Length > 2)
00019                     return true;
00020
00021                 /* if 1 means both y elements are the same so no combination has been found
00022                 if(y.Intersect(x).Count() <= 1)
00023                     return false;
00024             }
00025         }
00026     }
00027 }
```

```
00023         return true;
00024     }
00025
00026     return false;
00027 }
00028
00029 public int GetHashCode(BeeSpecies[] obj)
00030 {
00031     unchecked
00032     {
00033         int hashCode = 13;
00034
00035         for (int i = 0; i < obj.Length; i++)
00036         {
00037             hashCode += (int)obj[i];
00038         }
00039
00040         return hashCode;
00041     }
00042 }
00043 }
00044 }
```

7.29 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/↔ PrefabDictionary.cs File Reference

Classes

- class [BeeGame.Core.Dictionaries.PrefabDictionary](#)
The prefabs available to the game

Namespaces

- namespace [BeeGame.Core.Dictionaries](#)

7.30 PrefabDictionary.cs

```
00001 using System.Collections.Generic;
00002 using UnityEngine;
00003
00004 namespace BeeGame.Core.Dictionaries
00005 {
00009     public static class PrefabDictionary
0010     {
0014         private static Dictionary<string, GameObject> prefabDictionary = new Dictionary<string, GameObject>();
0015
0019         public static void LoadPrefabs()
0020         {
0021             prefabDictionary = Resources.Resources.GetPrefabs();
0022         }
0023
0029         public static GameObject GetPrefab(string prefab)
0030         {
0031             return prefabDictionary[prefab];
0032         }
0033     }
0034 }
```

7.31 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/Sprite↔ Dictionary.cs File Reference

Classes

- class [BeeGame.Core.Dictionaries.SpriteDictionary](#)
All of the sprites available to the game

Namespaces

- namespace BeeGame.Core.Dictionaries

7.32 SpriteDictionary.cs

```

00001 using System.Collections.Generic;
00002 using UnityEngine;
00003
00004 namespace BeeGame.Core.Dictionaries
00005 {
00009     public static class SpriteDictionary
0010     {
0014         private static Dictionary<string, Sprite> itemSpriteDictionary = new Dictionary<string, Sprite>();
0015
0021         public static Sprite GetSprite(string spriteName)
0022         {
0023             itemSpriteDictionary.TryGetValue(spriteName, out Sprite sprite);
0024
0025             if (sprite == null)
0026                 return new Sprite();
0027
0028             return sprite;
0029         }
0030
0034         public static void LoadSprites()
0035         {
0036             itemSpriteDictionary = Resources.Resources.GetSprites();
0037         }
0038     }
0039 }
```

7.33 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Enums/Enums.cs File Reference

Namespaces

- namespace BeeGame.Core.Enums

Enumerations

- enum BeeGame.Core.Enums.HoneyCombType { BeeGame.Core.Enums.HoneyCombType.HONEY, BeeGame.Core.Enums.HoneyCombType.ICEY }

Honey Comb Types
- enum BeeGame.Core.Enums.BeeSpecies {
 BeeGame.Core.Enums.BeeSpecies.FOREST, BeeGame.Core.Enums.BeeSpecies.MEADOWS, BeeGame.Core.Enums.BeeSpecies.WINTRY,
 BeeGame.Core.Enums.BeeSpecies.MODEST, BeeGame.Core.Enums.BeeSpecies.MARSHY, BeeGame.Core.Enums.BeeSpecies.ENDER,
 BeeGame.Core.Enums.BeeSpecies.MONASTIC,
 BeeGame.Core.Enums.BeeSpecies.STEADFAST, BeeGame.Core.Enums.BeeSpecies.VALIANT, BeeGame.Core.Enums.BeeSpecies.COMMON,
 BeeGame.Core.Enums.BeeSpecies.CULTIVATED,
 BeeGame.Core.Enums.BeeSpecies.DILIGENT, BeeGame.Core.Enums.BeeSpecies.RURAL, BeeGame.Core.Enums.BeeSpecies.FARMERLY,
 BeeGame.Core.Enums.BeeSpecies.AGRARIAN,
 BeeGame.Core.Enums.BeeSpecies.UNWEARY, BeeGame.Core.Enums.BeeSpecies.INDUSTRIOUS,
 BeeGame.Core.Enums.BeeSpecies.ICY, BeeGame.Core.Enums.BeeSpecies.GLACIAL,
 BeeGame.Core.Enums.BeeSpecies.NOBLE, BeeGame.Core.Enums.BeeSpecies.IMPERIAL, BeeGame.Core.Enums.BeeSpecies.MAJESTIC,
 BeeGame.Core.Enums.BeeSpecies.MIRY,
 BeeGame.Core.Enums.BeeSpecies.BOGGY, BeeGame.Core.Enums.BeeSpecies.HERIOC,
 BeeGame.Core.Enums.BeeSpecies.PHANTASMAL, BeeGame.Core.Enums.BeeSpecies.SPECTRAL,
 BeeGame.Core.Enums.BeeSpecies.HERMETIC, BeeGame.Core.Enums.BeeSpecies.SECLUDED,
 BeeGame.Core.Enums.BeeSpecies.SINISTER, BeeGame.Core.Enums.BeeSpecies.FIENDISH,

```
BeeGame.Core.Enums.BeeSpecies.DEMONIC, BeeGame.Core.Enums.BeeSpecies.FRUGAL, BeeGame.Core.Enums.BeeSpecies.AUSTER, BeeGame.Core.Enums.BeeSpecies.VINDICTIVE, BeeGame.Core.Enums.BeeSpecies.EXOTIC, BeeGame.Core.Enums.BeeSpecies.ENDEMIC, BeeGame.Core.Enums.BeeSpecies.VENGEFUL, BeeGame.Core.Enums.BeeSpecies.AVENGING, BeeGame.Core.Enums.BeeSpecies.SETADFAST, BeeGame.Core.Enums.BeeSpecies.HEROIC }
```

The different possible bee Species

- enum BeeGame.Core.Enums.BeeType { BeeGame.Core.Enums.BeeType.QUEEN, BeeGame.Core.Enums.BeeType.DRONE, BeeGame.Core.Enums.BeeType.PRINCESS }

The different bee types

- enum BeeGame.Core.Enums.BeeTempPreference {
 BeeGame.Core.Enums.BeeTempPreference.FROZEN, BeeGame.Core.Enums.BeeTempPreference.COLD, BeeGame.Core.Enums.BeeTempPreference.TEMPERATE, BeeGame.Core.Enums.BeeTempPreference.HOT, BeeGame.Core.Enums.BeeTempPreference.HELL }

The different bee temp preferences

- enum BeeGame.Core.Enums.BeeLifeSpan {
 BeeGame.Core.Enums.BeeLifeSpan.HUMMINGBIRD, BeeGame.Core.Enums.BeeLifeSpan.SHORTEST, BeeGame.Core.Enums.BeeLifeSpan.SHORT, BeeGame.Core.Enums.BeeLifeSpan.NORMAL, BeeGame.Core.Enums.BeeLifeSpan.LONG, BeeGame.Core.Enums.BeeLifeSpan.LONGEST, BeeGame.Core.Enums.BeeLifeSpan.SEATURTLE }

The lifespan of the bee

- enum BeeGame.Core.Enums.BeeProductionSpeed { BeeGame.Core.Enums.BeeProductionSpeed.SLOW, BeeGame.Core.Enums.BeeProductionSpeed.NORMAL, BeeGame.Core.Enums.BeeProductionSpeed.FAST }

How fast the bee produces items

- enum BeeGame.Core.Enums.BeeEffect { BeeGame.Core.Enums.BeeEffect.NONE, BeeGame.Core.Enums.BeeEffect.POISON }

Any effects of the bee

- enum BeeGame.Core.Enums.BeeHumidityPreference {
 BeeGame.Core.Enums.BeeHumidityPreference.ARID, BeeGame.Core.Enums.BeeHumidityPreference.DRY, BeeGame.Core.Enums.BeeHumidityPreference.TEMPERATE, BeeGame.Core.Enums.BeeHumidityPreference.MOIST, BeeGame.Core.Enums.BeeHumidityPreference.HUMID }

Humidity preferences of the bee

- enum BeeGame.Core.Enums.Direction {
 BeeGame.Core.Enums.Direction.NORTH, BeeGame.Core.Enums.Direction.EAST, BeeGame.Core.Enums.Direction.SOUTH, BeeGame.Core.Enums.Direction.WEST, BeeGame.Core.Enums.Direction.UP, BeeGame.Core.Enums.Direction.DOWN }

Direction in the game

7.34 Enums.cs

```
00001 namespace BeeGame.Core.Enums
00002 {
00006     public enum HoneyCombType
00007     {
00008         HONEY, ICY
00009     };
00010
00011     #region BeeStuff
00012     public enum BeeSpecies
00016     {
00017         FOREST, MEADOWS, TROPICAL, WINTRY, MODEST,
MARSHY, ENDER, MONASTIC, STEADFAST, VALIANT,
COMMON, CULTIVATED, DILIGENT, RURAL, FARMERLY,
AGRARIAN, UNWEARY, INDUSTRIOS, ICY, GLACIAL,
NOBLE, IMPERIAL, MAJESTIC, MIRY, BOGGY, HEROIC,
PHANTASMAL, SPECTRAL, HERMETIC, SECLUDED,
SINISTER, FIENDISH, DEMONIC, FRUGAL, AUSTER,
VINDICTIVE, EXOTIC, ENDEMIC, VENGEFUL, AVENGING,
```

```

00018     SETADFAST, HEROIC
00019 };
00020
00023     public enum BeeType
00024 {
00025     QUEEN, DRONE, PRINCESS
00026 };
00027
00031     public enum BeeTempPreference
00032 {
00033     FROZEN, COLD, TEMPERATE, HOT, HELL
00034 };
00035
00039     public enum BeeLifeSpan
00040 {
00041     HUMMINGBIRD, SHORTEST, SHORT, NORMAL, LONG,
00042     LONGEST, SEATURTLE
00043 };
00044
00047     public enum BeeProductionSpeed
00048 {
00049     SLOW, NORMAL, FAST
00050 };
00051
00055     public enum BeeEffect
00056 {
00057     NONE, POISON
00058 };
00059
00063     public enum BeeHumidityPreference
00064 {
00065     ARID, DRY, TEMPERATE, MOIST, HUMID
00066 };
00067 #endregion BeeStuff
00068
00072     public enum Direction
00073 {
00074     NORTH, EAST, SOUTH, WEST, UP, DOWN
00075 };
00076 }

```

7.35 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Extensions.cs File Reference

Classes

- class [BeeGame.Core.Extensions](#)

Namespaces

- namespace [BeeGame.Core](#)

7.36 Extensions.cs

```

00001 using System;
00002 using System.Reflection;
00003 using UnityEngine;
00004 using BeeGame.Items;
00005
00006 namespace BeeGame.Core
00007 {
00008     public static class Extensions
00009     {
00010         public static T CloneObject<T>(this T obj)
00011         {
00012             /* gets the type of the given object
00013             Type typeSource = obj.GetType();
00014
00015             /* makes a new object of type T
00016             T objTarget = (T)Activator.CreateInstance(typeSource);
00017
00018             /* gets the properties in T
00019             PropertyInfo[] properties = obj.GetType().GetProperties();
00020
00021             foreach (PropertyInfo prop in properties)
00022             {
00023                 if (prop.CanRead && prop.CanWrite)
00024                 {
00025                     object value = prop.GetValue(obj);
00026
00027                     if (value != null)
00028                     {
00029                         PropertyInfo targetProp = objTarget.GetType().GetProperty(prop.Name);
00030
00031                         if (targetProp != null)
00032                         {
00033                             targetProp.SetValue(objTarget, value);
00034                         }
00035                     }
00036                 }
00037             }
00038         }
00039     }
00040 }

```

```
00027         PropertyInfo[] propertyInfo = typeSource.GetProperties(BindingFlags.Public | BindingFlags.
00028             NonPublic | BindingFlags.Instance);
00029
00030         /* applies the properties in T to the new type T object
00031         foreach (var property in propertyInfo)
00032         {
00033             if (property.CanWrite)
00034             {
00035                 /* if the property is a value just set it
00036                 if (property.PropertyType.IsValueType || property.PropertyType.IsEnum || property.
00037                     PropertyType.Equals(typeof(string)))
00038                 {
00039                     property.SetValue(objTarget, property.GetValue(obj, null), null);
00040                 }
00041
00042                 /* if the property is not a value type this function will need to be called
00043                 recursively as it could also have non value type variables
00044                 object PropertyValue = property.GetValue(obj, null);
00045
00046                 if (PropertyValue == null)
00047                 {
00048                     property.SetValue(objTarget, null, null);
00049                 }
00050                 else
00051                 {
00052                     property.SetValue(objTarget, PropertyValue.CloneObject(), null);
00053                 }
00054             }
00055
00056             /* gets all of the field in T
00057             FieldInfo[] fieldInfo = typeSource.GetFields();
00058
00059             /* applies all of the fields of T to the new object if type T in the same manor that the
00060             properties are applied
00061             foreach (var field in fieldInfo)
00062             {
00063                 if(field.FieldType.IsValueType || field.FieldType.IsEnum || field.FieldType.Equals(typeof(
00064                     string)))
00065                 {
00066                     field.SetValue(objTarget, field.GetValue(obj));
00067                 }
00068                 else
00069                 {
00070                     object fieldValue = field.GetValue(obj);
00071
00072                     if(fieldValue == null)
00073                     {
00074                         field.SetValue(objTarget, null);
00075                     }
00076                     else
00077                     {
00078                         field.SetValue(objTarget, fieldValue.CloneObject());
00079                     }
00080                 }
00081             }
00082
00083         }
00084
00085         public static Sprite ColourSprite(this Sprite sprite, Color colour, Color[]
00086             coloursToAvoid = null, bool setTransparentToWhite = false)
00087         {
00088             Texture2D tex = new Texture2D((int)sprite.rect.width, (int)sprite.rect.height)
00089             {
00090                 filterMode = FilterMode.Point,
00091                 wrapMode = TextureWrapMode.Clamp
00092             };
00093
00094             /* sets the texture pixels to the pixels of the sprite so the original sprite is not modified
00095             tex.SetPixels(sprite.texture.GetPixels());
00096
00097             for (int x = 0; x < tex.width; x++)
00098             {
00099                 for (int y = 0; y < tex.height; y++)
00100                 {
00101                     /* if we dont have to avoid any colours set the pixel
00102                     if (coloursToAvoid == null)
00103                     {
00104                         tex.SetPixel(x, y, tex.GetPixel(x, y) * colour);
00105                     }
00106                     else
00107                     {
00108                         for (int i = 0; i < coloursToAvoid.Length; i++)
00109                         {
00110                             if (coloursToAvoid[i] == colour)
00111                             {
00112                                 tex.SetPixel(x, y, tex.GetPixel(x, y));
00113                             }
00114                         }
00115                     }
00116                 }
00117             }
00118         }
00119     }
00120 }
```

```

00116             /* if this colour should be avoided skip this iteration of the loop and move
00117             on
00118                 if (tex.GetPixel(x, y) == coloursToAvoid[i])
00119                     goto Skip;
00120
00121             tex.SetPixel(x, y, tex.GetPixel(x, y) * colour);
00122         }
00123
00124         /* if transparent pixels should be set to white do that
00125         if (setTransparentToWhite && tex.GetPixel(x, y).a == 0)
00126             tex.SetPixel(x, y, Color.white);
00127
00128         Skip:
00129             continue;
00130         }
00131     }
00132
00133     /* apply the new texture with its colours
00134     tex.Apply();
00135
00136     /* return the Texture2D as a sprite
00137     return Sprite.Create(tex, new Rect(0, 0, tex.width, tex.height), new
00138     THVector2(0.5f, 0.5f));
00139   }
00140
00141   public static void SpawnItem(this Item item, THVector3 position, Quaternion
00142   rotation = new Quaternion())
00143   {
00144       GameObject go = MonoBehaviour.Instantiate(UnityEngine.Resources.Load("
00145       Prefabs/ItemGameObject") as GameObject, position, rotation) as GameObject;
00146       go.GetComponent<ItemGameObject>().item = item;
00147   }
00148 }
```

7.37 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityTypeReplacements/← THInput.cs File Reference

Classes

- class BeeGame.Core.THInput

My implementation of the unity input system. Acts as a buffer layer to the unity system so that the input keys can be changed at runtime

Namespaces

- namespace BeeGame.Core

7.38 THInput.cs

```

00001 using System;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004 using BeeGame.Exceptions;
00005
00006 namespace BeeGame.Core
00007 {
00011   public static class THInput
00012   {
00016       private static Dictionary<string, object> inputButtons = new Dictionary<string, object>()
00017       {
00018           {"Forward", KeyCode.W},
00019           {"Backward", KeyCode.S },
00020           {"Right", KeyCode.D },
00021           {"Left", KeyCode.A },
00022           {"Player Inventory", KeyCode.E },
00023           {"Quest Book", KeyCode.Mouse1 },
00024           {"Interact", KeyCode.Mouse1 },
00025           {"Place", KeyCode.Mouse1 },
00026           {"Break Block", KeyCode.Mouse0 },
```

```
00027         {"Close Menu/Inventory", new KeyCode[2] { KeyCode.Escape, KeyCode.E } },
00028         {"Jump", KeyCode.Space }
00029     };
00030
00034     public static bool isAnotherInventoryOpen;
00035
00039     public static bool blockInventoryJustClosed;
00043     internal static bool chestOpen;
00044
00050     public static bool GetButtonDown(string button)
00051     {
00052         if (!inputButtons.ContainsKey(button))
00053         {
00054             throw new InputException($"Key input name not defined: {button}");
00055         }
00056
00057         switch (inputButtons[button])
00058         {
00059             case KeyCode[] arry:
00060                 /*for each possible key, check if it was pressed and if it was return that it was; if
none of them was pressed, return false
00061                 foreach (var item in arry)
00062                 {
00063                     if (Input.GetKeyDown(item))
00064                     {
00065                         return true;
00066                     }
00067                 }
00068
00069                 return false;
00070             default:
00071                 return Input.GetKeyDown((KeyCode)inputButtons[button]);
00072         }
00073     }
00074
00080     public static bool GetButton(string button)
00081     {
00082         if (!inputButtons.ContainsKey(button))
00083         {
00084             throw new InputException($"Key input name not defined: {button}");
00085         }
00086
00087         switch (inputButtons[button])
00088         {
00089             case KeyCode[] arry:
00090                 /*for each possible key, check if it was pressed and if it was return that it was; if
none of them was pressed return false
00091                 foreach (var item in arry)
00092                 {
00093                     if (Input.GetKey(item))
00094                     {
00095                         return true;
00096                     }
00097                 }
00098
00099                 return false;
00100             default:
00101                 return Input.GetKey((KeyCode)inputButtons[button]);
00102         }
00103     }
00104
00110     public static bool GetButtonUp(string button)
00111     {
00112         if (!inputButtons.ContainsKey(button))
00113         {
00114             throw new InputException($"Key input name not defined: {button}");
00115         }
00116
00117         switch (inputButtons[button])
00118         {
00119             case KeyCode[] arry:
00120                 /*for each possible key, check if it was pressed and if it was return that it was; if
none of them was pressed return false
00121                 foreach (var item in arry)
00122                 {
00123                     if (Input.GetKeyUp(item))
00124                     {
00125                         return true;
00126                     }
00127                 }
00128
00129                 return false;
00130             default:
00131                 return Input.GetKeyUp((KeyCode)inputButtons[button]);
00132         }
00133     }
00134 }
```

```

00140     public static int GetAxis(string axis)
00141     {
00142         int returnAxis = 0;
00143
00144         if (axis == "Horizontal")
00145         {
00146             if (GetButton("Right"))
00147             {
00148                 returnAxis += 1;
00149             }
00150
00151             if (GetButton("Left"))
00152             {
00153                 returnAxis -= 1;
00154             }
00155         }
00156         else if (axis == "Vertical")
00157         {
00158             if (GetButton("Forward"))
00159             {
00160                 returnAxis += 1;
00161             }
00162
00163             if (GetButton("Backward"))
00164             {
00165                 returnAxis -= 1;
00166             }
00167         }
00168
00169         return returnAxis;
00170     }
00171 }
00172 }
```

7.39 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityTypeReplacements/THQuaternion.cs File Reference

Classes

- struct [BeeGame.Core.UnityTypeReplacements.THQuaternion](#)

Namespaces

- namespace [BeeGame.Core.UnityTypeReplacements](#)

7.40 THQuaternion.cs

```

00001 using System;
00002 using UnityEngine;
00003
00004 namespace BeeGame.Core.UnityTypeReplacements
00005 {
00006     [Serializable]
00007     public struct THQuaternion
00008     {
00009         public float x;
00010         public float y;
00011         public float z;
00012         public float w;
00013
00014         public static Quaternion identity => new Quaternion(0, 0, 0, 1);
00015
00016         public THQuaternion(float x, float y, float z, float w)
00017         {
00018             this.x = x;
00019             this.y = y;
00020             this.z = z;
00021             this.w = w;
00022         }
00023
00024         public THQuaternion(THVector3 vector, float w)
00025         {
```

```
00026         x = vector.x;
00027         y = vector.y;
00028         z = vector.z;
00029         this.w = w;
00030     }
00031
00032     #region Overrides
00033     public static THQuaternion operator +(THQuaternion a,
00034         THQuaternion b)
00035     {
00036         a.x += b.x;
00037         a.y += b.y;
00038         a.z += b.z;
00039         a.w += b.w;
00040
00041         return a;
00042     }
00043     public static THQuaternion operator +(THQuaternion a, float b)
00044     {
00045         a.x += b;
00046         a.y += b;
00047         a.z += b;
00048         a.w += b;
00049
00050         return a;
00051     }
00052
00053     public static THQuaternion operator -(THQuaternion a,
00054         THQuaternion b)
00055     {
00056         a.x -= b.x;
00057         a.y -= b.y;
00058         a.z -= b.z;
00059         a.w -= b.w;
00060
00061         return a;
00062     }
00063     public static THQuaternion operator -(THQuaternion a, float b)
00064     {
00065         a.x -= b;
00066         a.y -= b;
00067         a.z -= b;
00068         a.w -= b;
00069
00070         return a;
00071     }
00072
00073     public static THQuaternion operator *(THQuaternion a,
00074         THQuaternion b)
00075     {
00076         a.x *= b.x;
00077         a.y *= b.y;
00078         a.z *= b.z;
00079         a.w *= b.w;
00080
00081         return a;
00082     }
00083     public static THQuaternion operator *(THQuaternion a, float b)
00084     {
00085         a.x *= b;
00086         a.y *= b;
00087         a.z *= b;
00088         a.w *= b;
00089
00090         return a;
00091     }
00092
00093     public static THQuaternion operator /(THQuaternion a,
00094         THQuaternion b)
00095     {
00096         a.x /= b.x;
00097         a.y /= b.y;
00098         a.z /= b.z;
00099         a.w /= b.w;
00100
00101         return a;
00102     }
00103     public static THQuaternion operator /(THQuaternion a, float b)
00104     {
00105         a.x /= b;
00106         a.y /= b;
00107         a.z /= b;
00108         a.w /= b;
```

```

00109         return a;
00110     }
00111 }
00112 #endregion
00113
00114 #region Implicit Operators
00115 public static implicit operator THQuaternion(Quaternion q)
00116 {
00117     return new THQuaternion(q.x, q.y, q.z, q.w);
00118 }
00119
00120 public static implicit operator Quaternion(THQuaternion q)
00121 {
00122     return new Quaternion(q.x, q.y, q.z, q.w);
00123 }
00124 #endregion
00125 }
00126 }
```

7.41 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityTypeReplacements/← THVector2.cs File Reference

Classes

- struct **BeeGame.Core.THVector2**
Serializable version of Vector2

Namespaces

- namespace **BeeGame.Core**

7.42 THVector2.cs

```

00001 using System;
00002 using UnityEngine;
00003
00004 namespace BeeGame.Core
00005 {
00009     [Serializable]
0010 public struct THVector2
0011 {
0012     #region Data
0013     public float x;
0020     public float y;
0021     #endregion
0022
0023     #region Constructor
0024     public THVector2(float x, float y)
0030     {
0031         this.x = x;
0032         this.y = y;
0033     }
0034
0039     public THVector2(THVector2 vec2)
0040     {
0041         this = vec2;
0042     }
0043
0048     public THVector2(Vector2 vec2)
0049     {
0050         this = vec2;
0051     }
0052     #endregion
0053
0054     #region Overrides
0055     public override bool Equals(object obj)
0056     {
0057         if (!(obj is THVector2))
0058             return false;
0059         if (obj.GetHashCode() == GetHashCode())
0060             return true;
0061         return false;
0062     }
0063 }
```

```
00062         }
00063
00064     public override int GetHashCode()
00065     {
00066         unchecked
00067         {
00068             int hash = 13;
00069
00070             hash *= 443 * x.GetHashCode();
00071             hash *= 373 * y.GetHashCode();
00072
00073             return hash;
00074         }
00075     }
00076
00077     public override string ToString()
00078     {
00079         return $"{x}, {y}";
00080     }
00081
00082     public static bool operator ==(THVector2 a, THVector2 b)
00083     {
00084         return a.Equals(b);
00085     }
00086     public static bool operator !=(THVector2 a, THVector2 b)
00087     {
00088         return !(a == b);
00089     }
00090
00091     public static THVector2 operator +(THVector2 a,
00092         THVector2 b)
00093     {
00094         a.x += b.x;
00095         a.y += b.y;
00096
00097         return a;
00098     }
00099     public static THVector2 operator +(THVector2 a, float b)
00100    {
00101        a.x += b;
00102        a.y += b;
00103
00104        return a;
00105    }
00106    public static THVector2 operator +(float a, THVector2 b)
00107    {
00108        return new THVector2(a + b.x, a + b.y);
00109    }
00110    public static THVector2 operator -(THVector2 a,
00111        THVector2 b)
00112    {
00113        a.x -= b.x;
00114        a.y -= b.y;
00115
00116        return a;
00117    }
00118    public static THVector2 operator -(THVector2 a, float b)
00119    {
00120        a.x += b;
00121        a.y += b;
00122
00123        return a;
00124    }
00125    public static THVector2 operator -(float a, THVector2 b)
00126    {
00127        return new THVector2(a - b.x, a - b.y);
00128    }
00129    public static THVector2 operator *(THVector2 a,
00130        THVector2 b)
00131    {
00132        a.x *= b.x;
00133        a.y *= b.y;
00134
00135        return a;
00136    }
00137    public static THVector2 operator *(THVector2 a, float b)
00138    {
00139        a.x *= b;
00140        a.y *= b;
00141
00142        return a;
00143    }
00144    public static THVector2 operator *(float a, THVector2 b)
00145    {
00146        return new THVector2(a * b.x, a * b.y);
00147    }
00148    public static THVector2 operator /(THVector2 a,
```

```

    THVector2 b)
00146     {
00147         a.x /= b.x;
00148         a.y /= b.y;
00149
00150         return a;
00151     }
00152     public static THVector2 operator /(THVector2 a, float b)
00153     {
00154         a.x /= b;
00155         a.y /= b;
00156
00157         return a;
00158     }
00159     public static THVector2 operator /(float a, THVector2 b)
00160     {
00161         return new THVector2(a / b.x, a / b.y);
00162     }
00163 #endregion
00164
00165 #region Implicit Operators
00166 public static implicit operator Vector2(THVector2 vec2)
00167 {
00168     return new Vector2(vec2.x, vec2.y);
00169 }
00170
00171 public static implicit operator THVector2(Vector2 vec2)
00172 {
00173     return new THVector2(vec2.x, vec2.y);
00174 }
00175 #endregion
00176 }
00177 }
```

7.43 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityTypeReplacements/← THVector3.cs File Reference

Classes

- struct [BeeGame.Core.THVector3](#)
Serializable version of Vector3

Namespaces

- namespace [BeeGame.Core](#)

7.44 THVector3.cs

```

00001 using System;
00002 using UnityEngine;
00003
00004 namespace BeeGame.Core
00005 {
00009     [Serializable]
0010     public struct THVector3
0011     {
0012         #region Data
0013         public float x;
0020         public float y;
0024         public float z;
0025     #endregion
0026
0027     #region Constructors
0028     public THVector3(float x, float y, float z)
0035     {
0036         this.x = x;
0037         this.y = y;
0038         this.z = z;
0039     }
0040
0045     public THVector3(THVector3 vec3)
0046     {
```

```
00047         this = vec3;
00048     }
00049
00050     public THVector3(Vector3 vec3)
00051     {
00052         this = vec3;
00053     }
00054
00055     public THVector3(Terrain.ChunkWorldPos vec3)
00056     {
00057         this = vec3;
00058     }
00059     #endregion
00060
00061     #region Methods
00062     public static float Distance(THVector3 a, THVector3 b)
00063     {
00064         return (float)Math.Sqrt(Math.Pow((a.x - b.x), 2) + Math.Pow((a.y - b.
00065         y), 2) + Math.Pow((a.z - b.z), 2));
00066     }
00067     #endregion
00068
00069     #region Overrides
00070     public override bool Equals(object obj)
00071     {
00072         if (!(obj is THVector3))
00073             return false;
00074         if (obj.GetHashCode() == GetHashCode())
00075             return true;
00076         return false;
00077     }
00078
00079     public override int GetHashCode()
00080     {
00081         unchecked
00082         {
00083             int hash = 13;
00084
00085             hash *= 443 * x.GetHashCode();
00086             hash *= 373 * y.GetHashCode();
00087             hash *= 127 * z.GetHashCode();
00088
00089             return hash;
00090         }
00091     }
00092
00093     public override string ToString()
00094     {
00095         return $"{x}, {y}, {z}";
00096     }
00097
00098     public static bool operator ==(THVector3 a, THVector3 b)
00099     {
00100         return a.Equals(b);
00101     }
00102     public static bool operator !=(THVector3 a, THVector3 b)
00103     {
00104         return !(a == b);
00105     }
00106
00107     public static THVector3 operator +(THVector3 a,
00108                                         THVector3 b)
00109     {
00110         a.x += b.x;
00111         a.y += b.y;
00112         a.z += b.z;
00113
00114         return a;
00115     }
00116     public static THVector3 operator +(THVector3 a, float b)
00117     {
00118         a.x += b;
00119         a.y += b;
00120         a.z += b;
00121
00122         return a;
00123     }
00124     public static THVector3 operator +(float a, THVector3 b)
00125     {
00126         return new THVector3(a + b.x, a + b.y, a + b.z);
00127     }
00128
00129     public static THVector3 operator -(THVector3 a,
00130                                         THVector3 b)
00131     {
00132         a.x -= b.x;
00133         a.y -= b.y;
00134         a.z -= b.z;
00135
00136     }
```

```

00194         return a;
00195     }
00196 }
00203     public static THVector3 operator -(THVector3 a, float b)
00204 {
00205     a.x += b;
00206     a.y += b;
00207     a.z += b;
00208
00209     return a;
00210 }
00217     public static THVector3 operator -(float a, THVector3 b)
00218 {
00219     return new THVector3(a - b.x, a - b.y, a - b.z);
00220 }
00227     public static THVector3 operator *(THVector3 a,
00228     THVector3 b)
00229 {
00230     a.x *= b.x;
00231     a.y *= b.y;
00232     a.z *= b.z;
00233
00234     return a;
00241     public static THVector3 operator *(THVector3 a, float b)
00242 {
00243     a.x *= b;
00244     a.y *= b;
00245     a.z *= b;
00246
00247     return a;
00248 }
00255     public static THVector3 operator *(float a, THVector3 b)
00256 {
00257     return new THVector3(a * b.x, a * b.y, a * b.z);
00258 }
00265     public static THVector3 operator /(THVector3 a,
00266     THVector3 b)
00267 {
00268     a.x /= b.x;
00269     a.y /= b.y;
00270     a.z /= b.z;
00271
00272     return a;
00279     public static THVector3 operator /(THVector3 a, float b)
00280 {
00281     a.x /= b;
00282     a.y /= b;
00283     a.z /= b;
00284
00285     return a;
00286 }
00293     public static THVector3 operator /(float a, THVector3 b)
00294 {
00295     return new THVector3(a / b.x, a / b.y, a / b.z);
00296 }
00297 #endregion
00298
00299 #region Implicit Operators
00300     public static implicit operator Vector3(THVector3 vec3)
00301 {
00302     return new Vector3(vec3.x, vec3.y, vec3.z);
00303 }
00313     public static implicit operator THVector3(Vector3 vec3)
00314 {
00315     return new THVector3(vec3.x, vec3.y, vec3.z);
00316 }
00317 #endregion
00318
00319 #region Explicit Operators
00320     public static explicit operator Quaternion(THVector3 vec3)
00321 {
00322     return new Quaternion(vec3.x, vec3.y, vec3.z, 0);
00323 }
00331 #endregion
00332 }
00333 }

```

7.45 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Exceptions/Crafting← RecipeAdditionException.cs File Reference

Classes

- class BeeGame.Exceptions.CraftingRecipeAdditionException

Namespaces

- namespace BeeGame.Exceptions

7.46 CraftingRecipeAdditionException.cs

```
00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using System.Text;
00005
00006 namespace BeeGame.Exceptions
00007 {
00008     public class CraftingRecipeAdditionException : Exception
00009     {
00010         public CraftingRecipeAdditionException() : base()
00011         {
00012         }
00013
00014         public CraftingRecipeAdditionException(string message) : base(
00015             message)
00016         {
00017         }
00018
00019         public CraftingRecipeAdditionException(string message, Exception
00020             innerException) : base(message, innerException)
00021         {
00022         }
00023
00024     }
00025 }
```

7.47 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Exceptions/InputException.cs File Reference**Classes**

- class BeeGame.Exceptions.InputException

Namespaces

- namespace BeeGame.Exceptions

7.48 InputException.cs

```

00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using System.Text;
00005
00006 namespace BeeGame.Exceptions
00007 {
00008     public class InputException : Exception
00009     {
00010         public InputException() : base()
00011         {
00012         }
00013
00014
00015         public InputException(string message) : base(message)
00016         {
00017
00018
00019
00020         public InputException(string message, Exception innerException) : base(message,
00021             innerException)
00022         {
00023
00024     }
00025 }
```

7.49 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Exceptions/Quest← AlreadyExistsException.cs File Reference

Classes

- class [BeeGame.Exceptions.QuestAlreadyExistsException](#)

Namespaces

- namespace [BeeGame.Exceptions](#)

7.50 QuestAlreadyExistsException.cs

```

00001 using System;
00002
00003 namespace BeeGame.Exceptions
00004 {
00005     public class QuestAlreadyExistsException : Exception
00006     {
00007         public QuestAlreadyExistsException() : base()
00008         {
00009
00010
00011
00012         public QuestAlreadyExistsException(string message) : base(message)
00013         {
00014
00015
00016
00017         public QuestAlreadyExistsException(string message, Exception
00018             innerException) : base(message, innerException)
00019         {
00020
00021     }
00022 }
```

7.51 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Apiary
CraftingOutputSlot.cs File Reference

Classes

- class BeeGame.Inventory.ApiaryCraftingOutputSlot

Overrides the

Namespaces

- namespace BeeGame.Inventory

7.52 ApiaryCraftingOutputSlot.cs

```
00001 using UnityEngine.EventSystems;
00002 using BeeGame.Items;
00003 using BeeGame.Inventory.BlockInventory;
00004 using BeeGame.Core;
00005 using BeeGame.Quest;
00006 using System;
00007
00008 namespace BeeGame.Inventory
00009 {
00013     public class ApiaryCraftingOutputSlot :
00014         InventorySlot, IPointerClickHandler, IPointerEnterHandler, IPointerExitHandler
00018     {
00019         protected new void Update()
00020         {
00021             CheckItem();
00022             UpdateIcon();
00023         }
00028         public override void OnPointerClick(PointerEventData eventData)
00029         {
00030             /* record what item was in the slot before it is moved
00031             Item before = item;
00032
00033             base.OnPointerClick(eventData);
00034
00035             /* if the item is different now then the crafting result must have been removed so call the
00036             event
00037             if (before != item && before != null)
00038                 ((CraftingTableInventory)myInventory).craftingResultRemoved.Invoke();
00039
00040             if (before is Bee b)
00041                 QuestEvents.CallBeeCraftedEvent(b.normalBee?.pSpecies ?? b.
00042                     queenBee.queen.pSpecies);
00041             else
00042                 QuestEvents.CallItemCraftedEvent(before.
00043                     GetHashCode());
00044         }
00045 }
```

7.53 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/BlockInventory/←
ApiaryInventory.cs File Reference

Classes

- class BeeGame.Inventory.ApiaryInventory

Inventory for Apriarys Apairy

Namespaces

- namespace BeeGame.Inventory

7.54 ApiaryInventory.cs

```

00001 using UnityEngine;
00002 using UnityEngine.UI;
00003 using BeeGame.Blocks;
00004
00005 namespace BeeGame.Inventory
00006 {
00013     public class ApiaryInventory : ChestInventory
00014     {
00015         #region Data
00016         private bool beesCombineing;
00020
00024         public float combinationTime = 0;
00025
00029         public Slider timerSlideer;
00030     #endregion
00031
00032         #region Unity Methods
00033         private void Update()
00037         {
00038             /* Updates the base class as unity Update function does not run on parent classes
00039             UpdateChestInventory();
00040
00041             /* if the apiary is not an item on the ground and bees are not currently combineing check is
00042             bees should be combineing
00043             if(items.itemsInInventory.Length > 0 && !beesCombineing)
00044                 CheckforBees();
00045
00046             /* if the currently combineing bees has finished combineing
00047             if (combinationTime < 0 && beesCombineing)
00048                 {
00049                     /* make the items that the bees should make and destroy the spent queen
00050                     (Apiary)myblock.MakeBees(items.itemsInInventory[0] as Items.Bee, ref items.
00051                     itemsInInventory);
00052                     beesCombineing = false;
00053                     items.itemsInInventory[0] = null;
00054
00055                     /* save the changes to the inventory
00056                     SaveInv();
00057                 }
00058
00061         private void FixedUpdate()
00062         {
00063             /* if bees are combineing reduce the combination time
00064             if (beesCombineing)
00065                 timerSlideer.value = combinationTime -= 0.1f;
00066         }
00067     #endregion
00068
00069         #region Apiary Stuff
00070         private void CheckforBees()
00074         {
00075             Items.Item posOneItem = items.itemsInInventory[0];
00076             Items.Item posTwoItem = items.itemsInInventory[1];
00077
00078             /* the item is checkd if it is a bee and if it is then a new variable is made for convenience
00079             /* if it is a queen then just set the combination time and go
00080             if (posOneItem is Items.Bee b && b.beeType == Core.Enums.BeeType.QUEEN)
00081             {
00082                 combinationTime = ((float)b.queenBee.queen.pLifespan + 1) * 2;
00083                 beesCombineing = true;
00084                 SaveInv();
00085
00086                 timerSlideer.maxValue = combinationTime;
00087
00088                 return;
00089             }
00090
00091             /* of one bee is a princess and another is a drone in the correct slots combine them
00092             if(posOneItem is Items.Bee b1 && posTwoItem is Items.Bee b2 && b1.beeType == Core.Enums.BeeType
00093 .PRINCESS && b2.beeType == Core.Enums.BeeType.DRONE)
00093             {
00094                 /* convert the princess to a queen with the paired drone
00095                 Items.Bee.ConvertToQueen(ref b1, b2.normalBee);
00096
00097                 /* reduce number of drones in slot by 1 and check it is a valid stack number
00098                 items.itemsInInventory[1].itemStackCount -= 1;
00099                 slots[0].item = b1;
00100
00101                 if (items.itemsInInventory[1].itemStackCount <= 0)
00102                     items.itemsInInventory[1] = null;
00103
00104                 /* set the combination time
00105                 combinationTime = ((float)b1.queenBee.queen.pLifespan + 1) * 2;

```

```

00106         beesCombining = true;
00107
00108         SaveInv();
00109
00110         /* set the slider max to the combination time
00111         timerSlideer.MaxValue = combinationTime;
00112     }
00113 }
00114 #endregion
00115
00116 #region Overrides
00117 public override void SetChestInventory(string invName = "Apiary")
00118 {
00119     base.SetChestInventory("Apiary");
00120 }
00121 #endregion
00122 }
00123 }
```

7.55 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/BlockInventory/ChestInventory.cs File Reference

Classes

- class [BeeGame.Inventory.ChestInventory](#)
Inventory for the chests

Namespaces

- namespace [BeeGame.Inventory](#)

7.56 ChestInventory.cs

```

00001 using BeeGame.Core;
00002 using BeeGame.Terrain;
00003 using UnityEngine;
00004 using static BeeGame.Core.ThInput;
00005
00006 namespace BeeGame.Inventory
00007 {
00011     public class ChestInventory : Inventory
00012     {
00013         #region Data
00014         public THVector3 inventoryPosition;
00021         public Inventory playerInventory;
00025         public GameObject inventory;
00026
00030         public int inventorySize;
00031     #endregion
00032
00033         #region Unity Methods
00034         void Update()
00038         {
00039             UpdateChestInventory();
00040         }
00041
00045         public void UpdateChestInventory()
00046         {
00047             /* the chest should always have a player inventory when it does this but checks just in case
00048             if (playerInventory != null)
00049                 UpdateBase();
00050
00051             /* checks if the inventory should be closed
00052             if (GetButtonDown("Player Inventory") && thisInventoryOpen && floatingItem == null)
00053                 ToggleInventory(playerInventory);
00054         }
00055     #endregion
00056
00060         public virtual void SetChestInventory(string invName = "Chest")
00061         {
00062             SetInventorySize(inventorySize);
00063             /* sets the UI to not be seen as inventories cannot start open

```

```

00064     inventory.SetActive(false);
00065
00066     /* sets the name and position if this inventory used during serialization and deserialization
00067     inventoryName = $"{invName} @ {(ChunkWorldPos)inventoryPosition}";
00068
00069     /** loads the inventory if it had had items put in it last time it existed
00070     Serialization.DeserializeInventory(this, inventoryName);
00071 }
00072
00073 #region Player Inventory
00074 void SetPlayerItems()
00075 {
00076     for (int i = 0; i < playerinventory.items.itemsInInventory.Length; i++)
00077     {
00078         items.itemsInInventory[i + (inventorySize - 36)] = playerinventory.
00079         items.itemsInInventory[i];
00080     }
00081 }
00082
00083 void ApplyPlayerItems()
00084 {
00085     for (int i = 0; i < playerinventory.items.itemsInInventory.Length; i++)
00086     {
00087         playerinventory.items.itemsInInventory[i] = items.itemsInInventory[i +
00088         (inventorySize - 36)];
00089     }
00090
00091     playerinventory.SaveInv();
00092 }
00093 #endregion
00094
00103 public override void ToggleInventory(Inventory inv)
00104 {
00105     /* sets the player inventory
00106     playerinventory = inv;
00107
00108     thisInventoryOpen = !thisInventoryOpen;
00109
00110     isAnotherInventoryOpen = thisInventoryOpen;
00111
00112     inventory.SetActive(!inventory.activeInHierarchy);
00113
00114     if (inventory.activeInHierarchy)
00115     {
00116         chestOpen = true;
00117
00118         /* stops the player inventory from being opened immidiately after this is closed
00119         blockInventoryJustClosed = true;
00120         SetPlayerItems();
00121         /* hides and locks the cursor
00122         Cursor.lockState = CursorLockMode.None;
00123         Cursor.visible = true;
00124     }
00125     else
00126     {
00127         chestOpen = false;
00128
00129         /* puts the items into the chest
00130         /* shows and unlocks the cursor
00131         ApplyPlayerItems();
00132         Cursor.lockState = CursorLockMode.Locked;
00133         Cursor.visible = false;
00134     }
00135 }
00136 }
00137 }

```

7.57 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/BlockInventory/← CraftingTableInventory.cs File Reference

Classes

- class BeeGame.Inventory.BlockInventory.CraftingTableInventory
Invenetory for the CraftingTable Block

Namespaces

- namespace BeeGame.Inventory.BlockInventory

7.58 CraftingTableInventory.cs

```
00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using System.Text;
00005 using BeeGame.Core;
00006 using BeeGame.Blocks;
00007 using BeeGame.Items;
00008 using BeeGame.Quest;
00009
00010 namespace BeeGame.Inventory.BlockInventory
00011 {
00015     public class CraftingTableInventory : ChestInventory
00016     {
00017         #region Data
00018         public delegate void ItemRemovedFromResult();
00025         public ItemRemovedFromResult craftingResultRemoved;
00026         #endregion
00027
00028         #region Unity Methods
00029         protected void Start()
00030         {
00034             SetChestInventory();
00035             craftingResultRemoved = CraftedItemRemoved;
00036         }
00037
00038
00042         protected void Update()
00043         {
00044             UpdateChestInventory();
00045
00046             if (inventory.activeInHierarchy)
00047             {
00048                 var shaped = CheckShapedRecipie();
00049                 var shapless = CheckShapelessRecipie();
00050
00051                 if (shaped != null)
00052                 {
00053                     items.itemsInInventory[9] = shaped;
00054                     return;
00055                 }
00056                 /* checks for shapless recipies second
00057                 else if(shapless != null)
00058                 {
00059                     items.itemsInInventory[9] = shapless;
00060                     return;
00061                 }
00062
00063                 items.itemsInInventory[9] = null;
00064             }
00065         }
00066
00070         protected void OnDestroy()
00071         {
00072             /* just ensures no memory leaks occur
00073             craftingResultRemoved -= CraftedItemRemoved;
00074         }
00075         #endregion
00076
00077         #region Crafting Stuff
00078         public virtual Item CheckShapedRecipie()
00079         {
00083             var items = new Item[9];
00084
00085             for (int i = 0; i < items.Length; i++)
00086             {
00087                 items[i] = base.items.itemsInInventory[i];
00088             }
00089
00090             /* if it is a recipe put the result into the crafting result slot
00091             return ((CraftingTable)myblock).ReturnShapedRecipieItem(items);
00092         }
00093
00097         public virtual Item CheckShapelessRecipie()
00098         {
00099             var items = new Item[9];
00100
00101             for (int i = 0; i < items.Length; i++)
00102             {
00103                 items[i] = base.items.itemsInInventory[i];
00104             }
00105
00106             return ((CraftingTable)myblock).ReturnShapelessRecipieItem(items);
00107         }
00108 }
```

```

00112     public void CraftedItemRemoved()
00113     {
00114         if (items.itemsInInventory[9] != null)
00115         {
00116             QuestEvents.CallItemCraftedEvent(items.itemsInInventory[9].
GetHashCode());
00117             for (int i = 0; i < 9; i++)
00118             {
00119                 if (items.itemsInInventory[i] != null)
00120                     items.itemsInInventory[i].itemStackCount -= 1;
00121             }
00122         }
00123     }
#endregion
00125
00126 #region Inventory Stuff
00127 public virtual void DropItemsFromInventory()
00128 {
00129     /* looks at every item in the crafting grid
00130     for (int i = 0; i < 9; i++)
00131     {
00132         if (items.itemsInInventory[i] != null)
00133         {
00134             /* spawns it and removes it from the inventory if an items exists within
00135             for (int j = 0; j < items.itemsInInventory[i].itemStackCount; j++)
00136             {
00137                 items.itemsInInventory[i].SpawnItem((THVector3)this.transform.position +
00138 new THVector3(0, 1, 0));
00139             }
00140             items.itemsInInventory[i] = null;
00141         }
00142     }
#endregion
00143
00144
00145
00146
00147
00148
00149
00150
00151 #region Overrides
00152 public override void ToggleInventory(Inventory inv)
00153 {
00154     base.ToggleInventory(inv);
00155
00156     /* if the inventory was closed drop the items within
00157     if (!inventory.activeInHierarchy)
00158         DropItemsFromInventory();
00159
00160
00161
00162
00163
00164
00165
00166
00167
00168
00169
00170
00171
00172     public override void SetChestInventory(string invName = "Workbench")
00173 {
00174     SetInventorySize(inventorySize);
00175     /* sets the UI to not be seen as inventories cannot start open
00176     inventory.SetActive(false);
00177
00178
00179
00180
00181
00182
00183
00184
00185
00186
00187     public override void AddItemToSlots(int slotIndex, Item item)
00188 {
00189     items.AddItem(slotIndex, item);
00190
00191
00192
00193
00194
00195     public override void SaveInv()
00196 {
00197     /* does not need to be saved so overrided to do nothing
00198
00199
#endregion
00200 }
00201 }

```

7.59 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/CraftingOutputSlot.cs File Reference

Classes

- class BeeGame.Inventory.CraftingOutputSlot
Overrides the

Namespaces

- namespace BeeGame.Inventory

7.60 CraftingOutputSlot.cs

```

00001 using UnityEngine.EventSystems;
00002 using BeeGame.Items;
00003 using BeeGame.Inventory.BlockInventory;
00004 using BeeGame.Core;
00005 using System;
00006
00007 namespace BeeGame.Inventory
00008 {
00012     public class CraftingOutputSlot : InventorySlot, IPointerClickHandler,
00013         IPointerEnterHandler, IPointerExitHandler
00017     {
00018         protected new void Update()
00019         {
00020             CheckItem();
00021             UpdateIcon();
00022         }
00027         public override void OnPointerClick(PointerEventData eventData)
00028         {
00029             /* record what item was in the slot before it is moved
00030             Item before = item;
00031
00032             base.OnPointerClick(eventData);
00033
00034             /* if the item is different now then the crafting result must have been removed so call the
00035             event
00036             if (before != item && before != null)
00037                 ((CraftingTableInventory)myInventory).craftingResultRemoved.Invoke();
00038
00039             Quest.QuestEvents.CallItemCraftedEvent(before.GetHashCode());
00040         }
00041 }

```

7.61 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Inventory.cs File Reference

Classes

- class [BeeGame.Inventory.Inventory](#)
Base class for all inventorys in the game

Namespaces

- namespace [BeeGame.Inventory](#)

7.62 Inventory.cs

```

00001 using System;
00002 using UnityEngine;
00003 using BeeGame.Items;
00004 using BeeGame.Core.Dictionaries;
00005
00006 namespace BeeGame.Inventory
00007 {
0011     public class Inventory : MonoBehaviour
0012     {
0013         #region Data
0014         public ItemsInInventory items;
0021         public InventorySlot[] slots;
0025         internal Item floatingItem;
0029         public string inventoryName = "";
0033         protected bool thisInventoryOpen = false;
0037         private GameObject spriteAtCursor;
0044         public Blocks.Block myblock;
0045         #endregion
0046         #region Init

```

```

00048     public bool InventorySet()
00049     {
00050         if (items == null)
00051             return true;
00052
00053         return false;
00054     }
00055
00056
00057
00058
00059
00060     public void SetInventorySize(int inventorySize)
00061     {
00062         items = new ItemsInInventory(slots.Length);
00063     }
00064
00065
00066
00067
00068
00069     public void SetAllItems(ItemsInInventory items)
00070     {
00071         this.items = items;
00072     }
00073
00074 #endregion
00075
00076
00077
00078
00079
00080
00081
00082 #region Update
00083     protected void UpdateBase()
00084     {
00085         PutItemsInSlots();
00086         DrawItemAtCursor();
00087     }
00088
00089
00090
00091
00092     private void DrawItemAtCursor()
00093     {
00094         if (floatingItem != null)
00095         {
00096             if (spriteAtCursor == null)
00097             {
00098                 spriteAtCursor = Instantiate(PrefabDictionary.
00099                     GetPrefab("ItemIcon"));
00100                 spriteAtCursor.GetComponentInChildren<UnityEngine.UI.Image>().sprite =
00101                     floatingItem.GetItemSprite();
00102                 /* will update a the sprite of in item is swapped between a slot and teh floating item if
00103                 the previous item wasnt put into a slot first
00104                 else if(spriteAtCursor != null)
00105                 {
00106                     spriteAtCursor.GetComponentInChildren<UnityEngine.UI.Image>().sprite =
00107                         floatingItem.GetItemSprite();
00108                 }
00109
00110                 spriteAtCursor.transform.GetChild(0).position = Input.mousePosition;
00111             }
00112             else
00113             {
00114                 Destroy(spriteAtCursor);
00115             }
00116         }
00117     }
00118 #endregion
00119
00120 #region Edit Inventory
00121     public virtual void ToggleInventory(Inventory inv)
00122     {
00123         throw new NotImplementedException();
00124     }
00125
00126
00127
00128
00129
00130
00131     public virtual void SaveInv()
00132     {
00133         Serialization.Serialization.SerializeInventory(this, inventoryName);
00134     }
00135
00136
00137     public virtual void SetItemInventory(Item[] items)
00138     {
00139
00140
00141
00142     public virtual void PutItemsInSlots()
00143     {
00144         /* goes through all of the items in the array setting then all to a slot
00145         for (int i = 0; i < slots.Length; i++)
00146         {
00147             slots[i].slotIndex = i;
00148             slots[i].myInventory = this;
00149             slots[i].item = items.itemsInInventory[i];
00150         }
00151     }
00152
00153
00154
00155     public ItemsInInventory GetAllItems()
00156     {
00157         return items;
00158     }
00159
00160
00161
00162
00163
00164     public virtual void AddItemToSlots(int slotIndex, Item item)
00165
00166
00167
00168
00169
00170
00171
00172
00173
00174
00175
00176
00177
00178
00179
00180
00181
00182
00183
00184
00185
00186
00187
00188
00189
00190
00191
00192
00193
00194
00195
00196
00197
00198
00199
00200
00201
00202
00203
00204
00205
00206
00207
00208
00209
00210
00211
00212
00213
00214
00215
00216
00217
00218
00219
00220
00221
00222
00223
00224
00225
00226
00227
00228
00229
00230
00231
00232
00233
00234
00235
00236
00237
00238
00239
00240
00241
00242
00243
00244
00245
00246
00247
00248
00249
00250
00251
00252
00253
00254
00255
00256
00257
00258
00259
00260
00261
00262
00263
00264
00265
00266
00267
00268
00269
00270
00271
00272
00273
00274
00275
00276
00277
00278
00279
00280
00281
00282
00283
00284
00285
00286
00287
00288
00289
00290
00291
00292
00293
00294
00295
00296
00297
00298
00299
00300
00301
00302
00303
00304
00305
00306
00307
00308
00309
00310
00311
00312
00313
00314
00315
00316
00317
00318
00319
00320
00321
00322
00323
00324
00325
00326
00327
00328
00329
00330
00331
00332
00333
00334
00335
00336
00337
00338
00339
00340
00341
00342
00343
00344
00345
00346
00347
00348
00349
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378
00379
00380
00381
00382
00383
00384
00385
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402
00403
00404
00405
00406
00407
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417
00418
00419
00420
00421
00422
00423
00424
00425
00426
00427
00428
00429
00430
00431
00432
00433
00434
00435
00436
00437
00438
00439
00440
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009
01010
01011
01012
01013
01014
01015
01016
01017
01018
01019
01020
01021
01022
01023
01024
01025
01026
01027
01028
01029
01030
01031
01032
01033
01034
01035
01036
01037
01038
01039
01040
01041
01042
01043
01044
01045
01046
01047
01048
01049
01050
01051
01052
01053
01054
01055
01056
01057
01058
01059
01060
01061
01062
01063
01064
01065
01066
01067
01068
01069
01070
01071
01072
01073
01074
01075
01076
01077
01078
01079
01080
01081
01082
01083
01084
01085
01086
01087
01088
01089
01090
01091
01092
01093
01094
01095
01096
01097
01098
01099
01100
01101
01102
01103
01104
01105
01106
01107
01108
01109
01110
01111
01112
01113
01114
01115
01116
01117
01118
01119
01120
01121
01122
01123
01124
01125
01126
01127
01128
01129
01130
01131
01132
01133
01134
01135
01136
01137
01138
01139
01140
01141
01142
01143
01144
01145
01146
01147
01148
01149
01150
01151
01152
01153
01154
01155
01156
01157
01158
01159
01160
01161
01162
01163
01164
01165
01166
01167
01168
01169
01170
01171
01172
01173
01174
01175
01176
01177
01178
01179
01180
01181
01182
01183
01184
01185
01186
01187
01188
01189
01190
01191
01192
01193
01194
01195
01196
01197
01198
01199
01200
01201
01202
01203
01204
01205
01206
01207
01208
01209
01210
01211
01212
01213
01214
01215
01216
01217
01218
01219
01220
01221
01222
01223
01224
01225
01226
01227
01228
01229
01230
01231
01232
01233
01234
01235
01236
01237
01238
01239
01240
01241
01242
01243
01244
01245
01246
01247
01248
01249
01250
01251
01252
01253
01254
01255
01256
01257
01258
01259
01260
01261
01262
01263
01264
01265
01266
01267
01268
01269
01270
01271
01272
01273
01274
01275
01276
01277
01278
01279
01280
01281
01282
01283
01284
01285
01286
01287
01288
01289
01290
01291
01292
01293
01294
01295
01296
01297
01298
01299
01300
01301
01302
01303
01304
01305
01306
01307
01308
01309
01310
01311
01312
01313
01314
01315
01316
01317
01318
01319
01320
01321
01322
01323
01324
01325
01326
01327
01328
01329
01330
01331
01332
01333
01334
01335
01336
01337
01338
01339
01340
01341
01342
01343
01344
01345
01346
01347
01348
01349
01350
01351
01352
01353
01354
01355
01356
01357
01358
01359
01360
01361
01362
01363
01364
01365
01366
01367
01368
01369
01370
01371
01372
01373
01374
01375
01376
01377
01378
01379
01380
01381
01382
01383
01384
01385
01386
01387
01388
01389
01390
01391
01392
01393
01394
01395
01396
01397
01398
01399
01400
01401
01402
01403
01404
01405
01406
01407
01408
01409
01410
01411
01412
01413
01414
01415
01416
01417
01418
01419
01420
01421
01422
01423
01424
01425
01426
01427
01428
01429
01430
01431
01432
01433
01434
01435
01436
01437
01438
01439
01440
01441
01442
01443
01444
01445
01446
01447
01448
01449
01450
01451
01452
01453
01454
01455
01456
01457
01458
01459
01460
01461
01462
01463
01464
01465
01466
01467
01468
01469
01470
01471
01472
01473
01474
01475
01476
01477
01478
01479
01480
01481
01482
01483
01484
01485
01486
01487
01488
01489
01490
01491
01492
01493
01494
01495
01496
01497
01498
01499
01500
01501
01502
01503
01504
01505
01506
01507
01508
01509
01510
01511
01512
01513
01514
01515
01516
01517
01518
01519
01520
01521
01522
01523
01524
01525
01526
01527
01528
01529
01530
01531
01532
01533
01534
01535
01536
01537
01538
01539
01540
01541
01542
01543
01544
01545
01546
01547
01548
01549
01550
01551
01552
01553
01554
01555
01556
01557
01558
01559
01560
01561
01562
01563
01564
01565
01566
01567
01568
01569
01570
01571
01572
01573
01574
01575
01576
01577
01578
01579
01580
01581
01582
01583
01584
01585
01586
01587
01588
01589
01590
01591
01592
01593
01594
01595
01596
01597
01598
01599
01600
01601
01602
01603
01604
01605
01606
01607
01608
01609
01610
01611
01612
01613
01614
01615
01616
01617
01618
01619
01620
01621
01622
01623
01624
01625
01626
01627
01628
01629
01630
01631
01632
01633
01634
01635
01636
01637
01638
01639
01640
01641
01642
01643
01644
01645
01646
01647
01648
01649
01650
01651
01652
01653
01654
01655
01656
01657
01658
01659
01660
01661
01662
01663
01664
01665
01666
01667
01668
01669
01670
01671
01672
01673
01674
01675
01676
01677
01678
01679
01680
01681
01682
01683
01684
01685
01686
01687
01688
01689
01690
01691
01692
01693
01694
01695
01696
01697
01698
01699
01700
01701
01702
01703
01704
01705
01706
01707
01708
01709
01710
01711
01712
01713
01714
01715
01716
01717
01718
01719
01720
01721
01722
01723
01724
01725
01726
01727
01728
01729
01730
01731
01732
01733
01734
01735
01736
01737
01738
01739
01740
01741
01742
01743
01744
01745
01746
01747
01748
01749
01750
01751
01752
01753
01754
01755
01756
01757
01758
01759
01760
01761
01762
01763
01764
01765
01766
01767
01768
01769
01770
01771
01772
01773
01774
01775
01776
01777
01778
01779
01780
01781
01782
01783
01784
01785
01786
01787
01788
01789
01790
01791
01792
01793
01794
01795
01796
01797
01798
01799
01800
01801
01802
01803
01804
01805
01806
01807
01808
01809
01810
01811
01812
01813
01814
01815
01816
01817
01818
01819
01820
01821
01822
01823
01824
01825
01826
01827
01828
01829
01830
01831
01832
01833
018
```

```

00170         {
00171             items.AddItem(slotIndex, item);
00172             /* saves the inventory changes
00173             Serialization.Serialization.SerializeInventory(this, inventoryName);
00174         }
00175
00181     public bool AddItemToInventory(Item item)
00182     {
00183         return items.AddItem(item);
00184     }
00185 #endregion
00186 }
00187 }
```

7.63 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/InventorySlot.cs File Reference

Classes

- class BeeGame.Inventory.InventorySlot

Namespaces

- namespace BeeGame.Inventory

7.64 InventorySlot.cs

```

00001 using UnityEngine;
00002 using UnityEngine.UI;
00003 using UnityEngine.EventSystems;
00004 using BeeGame.Items;
00005 using BeeGame.Core;
00006 using BeeGame.Core.Dictionaries;
00007
00008 namespace BeeGame.Inventory
00009 {
00010     public class InventorySlot : MonoBehaviour, IPointerClickHandler, IPointerEnterHandler,
00011     IPointerExitHandler
00011     {
00012         #region Data
00013         internal int slotIndex;
00020         public Item item;
00024         public Inventory myInventory;
00028         public GameObject itemText;
00032         public bool selectedSlot = false;
00036         public bool itemsCanBeInserted = true;
00037     #endregion
00038
00042     protected void Update()
00043     {
00044         CheckItem();
00045         UpdateIcon();
00046     }
00047
00051     internal void UpdateIcon()
00052     {
00053         if(item == null)
00054         {
00055             GetComponent<Image>().sprite = null;
00056         }
00057         else
00058         {
00059             if(!item.Equals(new Item()))
00060                 GetComponent<Image>().sprite = item.GetItemSprite();
00061         }
00062
00063         /* if the slot is selected in the hotbar give the player some indication by colouring it grey
00064         if (selectedSlot)
00065         {
00066             GetComponent<Image>().color = Color.gray;
00067         }
00068         else
```

```

00069         {
00070             GetComponent<Image>().color = Color.white;
00071         }
00072     }
00073
00074     #region Interact With Slot
00075     public virtual void OnPointerClick(PointerEventData eventData)
00076     {
00077         if (myInventory.floatingItem != null)
00078         {
00079             /* Left click moves whole stacks of items
00080             if (eventData.button == PointerEventData.InputButton.Left)
00081             {
00082                 /* If the item in the slot is empty put the floating item into it then clear it and
00083                 the slot can have items inserted
00084                 if (item == null && itemsCanBeInserted)
00085                 {
00086                     item = myInventory.floatingItem;
00087                     myInventory.floatingItem = null;
00088                     myInventory.AddItemToSlots(slotIndex, item);
00089                     return;
00090                 }
00091                 /* if the items are the same
00092                 if (myInventory.floatingItem == item && itemsCanBeInserted)
00093                 {
00094                     /* if the item in the inventoys stack count + the floating items stack count is
00095                     less than the max stack count
00096                     if (myInventory.floatingItem.itemStackCount + item.
00097                         itemStackCount <= item.maxStackCount)
00098                     {
00099                         AddToSlot(myInventory.floatingItem.
00100                             itemStackCount);
00101                         return;
00102                     }
00103                     /* if the item stack added is larger than the max count add as many as you can and
00104                     move on
00105                     else
00106                     {
00107                         AddToSlot(item.maxStackCount - item.
00108                             itemStackCount);
00109                         return;
00110                     }
00111                 }
00112             }
00113             /* if the tiems are the same but items cannot be inserted into the slot add as many
00114             items as you
00115             /* can from the slot to the floating item
00116             else if (myInventory.floatingItem == item && !itemsCanBeInserted)
00117             {
00118                 AddToFloatingItem();
00119                 return;
00120             }
00121             /* If the items were not == swap them
00122             else
00123             {
00124                 /* only if items can be inserted into the slot
00125                 if (itemsCanBeInserted)
00126                     SwapItems();
00127                 return;
00128             }
00129         }
00130         else if (eventData.button == PointerEventData.InputButton.Right)
00131         {
00132             /* if the item in slot is null add 1 from the floating item to it
00133             if (item == null && itemsCanBeInserted)
00134             {
00135                 AddToSlot(1);
00136                 return;
00137             }
00138             /* if the items are the same add 1 from the floating item to this item
00139             else if (item == myInventory.floatingItem && itemsCanBeInserted)
00140             {
00141                 AddToSlot(1);
00142                 return;
00143             }
00144         }
00145         /* if the floating item is null
00146         else
00147         {
00148             /* add 1/2 of the stack into the floating item if right click was pressed
00149             if (eventData.button == PointerEventData.InputButton.Right)
00150             {
00151                 SplitStack();
00152
00153                 return;
00154             }
00155

```

```
00156         if (item == null)
00157             return;
00158
00159         /* otherwise add the items into the floating item slot
00160         myInventory.floatingItem = item.CloneObject();
00161
00162         item.itemStackCount -= item.itemStackCount;
00163
00164         return;
00165     }
00166 }
00167 }
00168 }
00169
00170 void AddToFloatingItem()
00171 {
00172     /* if the whole stack can be added do it and move on
00173     if(myInventory.floatingItem.itemStackCount + item.
00174     itemStackCount <= item.maxStackCount)
00175     {
00176         myInventory.floatingItem.itemStackCount += item.
00177         itemStackCount;
00178
00179         item = null;
00180
00181         myInventory.AddItemToSlots(slotIndex, item);
00182
00183         return;
00184     }
00185 }
00186
00187 /* if the whole stack cannot be added calculate how many need to be removed from the slots
00188 item stack
00189     item.itemStackCount -= (item.maxStackCount - myInventory.
00190 floatingItem.itemStackCount);
00191     /* set the floating item to the max stack count
00192     myInventory.floatingItem.itemStackCount = item.
00193     maxStackCount;
00194
00195     myInventory.AddItemToSlots(slotIndex, item);
00196 }
00197
00198 void AddToSlot(int numerToAdd)
00199 {
00200     /* if the item in the slot is null create it
00201     if (item == null)
00202     {
00203         item = myInventory.floatingItem.CloneObject();
00204         item.itemStackCount = 0;
00205     }
00206
00207     /* add to number to add to the stack count
00208     item.itemStackCount += numerToAdd;
00209
00210     /* if the stack count is now larger than it should be dont let it be
00211     if (item.itemStackCount > item.maxStackCount)
00212     {
00213         item.itemStackCount = item.maxStackCount;
00214     }
00215
00216     /* remove the numebr if items form the floating item then check the floating item is not null
00217     myInventory.floatingItem.itemStackCount -= numerToAdd;
00218     CheckFloatingItem();
00219     /* save the inventory changes
00220     myInventory.AddItemToSlots(slotIndex, item);
00221 }
00222
00223
00224 void SplitStack()
00225 {
00226     myInventory.floatingItem = item.CloneObject();
00227     int give = (item.itemStackCount + 1) / 2;
00228     myInventory.floatingItem.itemStackCount = give;
00229     item.itemStackCount -= give;
00230
00231     if (item.itemStackCount <= 0)
00232         item = null;
00233
00234     myInventory.AddItemToSlots(slotIndex, item);
00235     Destroy(itemText);
00236 }
00237
00238
00239 void SwapItems()
00240 {
00241     /* temp copy of the item
00242     Item temp = myInventory.floatingItem;
00243     /* sets the floating item
00244     myInventory.floatingItem = item;
00245     /* sets the item that was in the floating item to the item in the the slot
```

```

00254         item = temp;
00255         /* Saves the changes to the inventory
00256         myInventory.AddItemToSlots(slotIndex, item);
00257         /* destroys the text as it is not needed anymore
00258         Destroy(itemText);
00259     }
00260
00261     void CheckFloatingItem()
00262     {
00263         if(myInventory.floatingItem.itemStackCount <= 0)
00264         {
00265             myInventory.floatingItem = null;
00266         }
00267     }
00268 #endregion
00269
00270     internal void CheckItem()
00271     {
00272         if (item != null && myInventory != null)
00273         {
00274             if (item.itemStackCount == 0 || item.itemName == "TestItem")
00275             {
00276                 myInventory.items.itemsInInventory[slotIndex] = null;
00277                 Destroy(itemText);
00278             }
00279         }
00280     }
00281
00282 #region Display Item On Hover
00283     public void OnPointerEnter(PointerEventData eventData)
00284     {
00285         /* if the item is null or the floating item has something in it dont display the item text as
00286         it is not necessary
00287         if (item != null && myInventory.floatingItem == null)
00288         {
00289             itemText = Instantiate(PrefabDictionary.
00290             GetPrefab("ItemDetails"));
00291             /* sets the text to the correct position
00292             itemText.transform.GetChild(0).position = Input.mousePosition;
00293             /* puts the correct text in the box
00294             itemText.transform.GetChild(0).GetComponent<Text>().text = $""
00295             {item.GetItemName()}\nStack: {item.itemStackCount}";
00296         }
00297     }
00298
00299     public void OnPointerExit(PointerEventData eventData)
00300     {
00301         Destroy(itemText);
00302     }
00303
00304     void OnDisable()
00305     {
00306         Destroy(itemText);
00307     }
00308 #endregion
00309 }
00310 }
```

7.65 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/ItemInventory/← BeeAlyzerInventory.cs File Reference

Classes

- class BeeGame.Inventory.BeeAlyzerInventory

Incentory for the chests

Namespaces

- namespace BeeGame.Inventory

7.66 BeeAlyzerInventory.cs

```

00001 using BeeGame.Core;
00002 using UnityEngine;
00003 using UnityEngine.UI;
00004 using static BeeGame.Core.THInput;
00005 using BeeGame.Items;
00006 using BeeGame.Quest;
00007
00008 namespace BeeGame.Inventory
00009 {
00013     public class BeeAlyzerInventory : Inventory
00014     {
00015         #region Data
00016         public Text infoText;
00023         private Inventory playerInventory;
00027         internal BeeAlyzer myItem;
00028         #endregion
00029
00030         protected new void Update()
00031         {
00032             if (GetButtonDown("Close Menu/Inventory"))
00033                 ToggleInventory(playerInventory);
00034             UpdateBase();
00035             PutItemsInSlots();
00036
00037             CheckForBeeAndHoney();
00038         }
00039
00043         public override void SaveInv()
00044         {
00045             return;
00046         }
00047
00048         #region Open/Close Inventory
00049         public override void ToggleInventory(Inventory inv)
00050         {
00055             thisInventoryOpen = !thisInventoryOpen;
00056
00057             isAnotherInventoryOpen = thisInventoryOpen;
00058
00059             if (this.gameObject.activeInHierarchy && !thisInventoryOpen)
00060             {
00061                 chestOpen = false;
00062
00063                 /* removes all of the items from thsi inventory
00064                 DropItemsFromInventory();
00065
00066                 /* tells item that inventory has been closed
00067                 myItem.OpenItemInvnetory();
00068                 Cursor.lockState = CursorLockMode.Locked;
00069                 Cursor.visible = false;
00070                 /* applies the chanegs to the players inventory
00071                 ApplyPlayerItems();
00072                 /* destroys this as it is not needed
00073                 Destroy(this.gameObject);
00074             }
00075             else
00076             {
00077                 chestOpen = true;
00078
00079                 SetInventorySize(slots.Length);
00080
00081                 playerInventory = inv;
00082
00083                 SetPlayerItems();
00084
00085                 PutItemsInSlots();
00086                 /* hides and locks the cursor
00087                 Cursor.lockState = CursorLockMode.None;
00088                 Cursor.visible = true;
00089             }
00090         }
00091
00095         public virtual void DropItemsFromInventory()
00096         {
00097             /* looks at every item in the crafting grid
00098             for (int i = 0; i < 3; i++)
00099             {
00100                 if (items.itemsInInventory[i] != null)
00101                 {
00102                     /* spawns it and removes it from the inventory if an items exists within
00103                     for (int j = 0; j < items.itemsInInventory[i].itemStackCount; j++)
00104                     {
00105                         MonoBehaviour.print(GameObject.FindGameObjectWithTag("Player").transform.position);
00106                         items.itemsInInventory[i].SpawnItem((THVector3)GameObject.

```

```

00107         }
00108         items.itemsInInventory[i] = null;
00109     }
00110 }
00111 }
00112 #endregion
00113
00114 #region Inventory Function
00115 public void CheckForBeeAndHoney()
00116 {
00117     if(slots[0].item == null)
00118     {
00119         //if (slots[1].item.GetHashCode() == Honey.ID && slots[1].item.itemStackCount >= 1)
00120         if (slots[2].item is Bee b)
00121         {
00122             b.canSeeBeeData = true;
00123             items.itemsInInventory[0] = slots[2].item;
00124             items.itemsInInventory[2] = null;
00125             //items.itemsInInventory[1].itemStackCount -= 1;
00126
00127             infoText.text = ReturnData(b);
00128         }
00129     }
00130     else
00131     {
00132         infoText.text = "";
00133     }
00134 }
00135 }
00136 }
00137 }
00138
00144 private string ReturnData(Bee b)
00145 {
00146     string returnString = "";
00147
00148     if (b.beeType == Core.Enums.BeeType.QUEEN)
00149     {
00150
00151     }
00152
00153     /* calls to check if a pure bread bee quest should be called
00154     QuestEvents.CallPureBeeCraftedEvent(b.
normalBee.pSpecies, b.normalBee.sSpecies);
00155
00156     returnString += $"Primary Species: {b.normalBee.pSpecies}\nSecondary Species:
{b.normalBee.sSpecies}\nPrimary Fertility: {b.normalBee.pFertility}\nSecondary Fertility: {b.normalBee.sFertility}\nPr
Lifespan: {b.normalBee.pLifespan}\nSecondary Lifespan: {b.normalBee.sLifespan}\nPrimary Production Speed:
{b.normalBee.pProdSpeed}\nSecondary Production Speed: {b.normalBee.sProdSpeed}";
00157
00158     return returnString;
00159 }
00160 #endregion
00161
00162 #region Set inventory
00163 void SetPlayerItems()
00164 {
00165     for (int i = 0; i < playerInventory.items.itemsInInventory.Length; i++)
00166     {
00167         items.itemsInInventory[i + (slots.Length - 36)] = playerInventory.
items.itemsInInventory[i];
00168     }
00169 }
00170
00171 void ApplyPlayerItems()
00172 {
00173     for (int i = 0; i < playerInventory.items.itemsInInventory.Length; i++)
00174     {
00175         playerInventory.items.itemsInInventory[i] = items.itemsInInventory[i +
(slots.Length - 36)];
00176     }
00177
00178     playerInventory.SaveInv();
00179 }
00180
00181 #endregion
00182 }
00183
00184 }
00185 }
00186
00187 }
00188 }

```

7.67 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/ItemInventory/QuestBookInventory.cs File Reference

Classes

- class BeeGame.Inventory.QuestBookInventory

Namespaces

- namespace BeeGame.Inventory

7.68 QuestBookInventory.cs

```

00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using System.Text;
00005 using System.Threading.Tasks;
00006 using BeeGame.Quest;
00007 using BeeGame.Items;
00008 using UnityEngine;
00009 using UnityEngine.UI;
00010 using static BeeGame.Core.THInput;
00011
00012 namespace BeeGame.Inventory
00013 {
00014     public class QuestBookInventory : Inventory
00015     {
00016         public Button questButton;
00017         public GameObject scrollObject;
00018         internal QuestBook myItem;
00019
00020         protected void Start()
00021         {
00022             AddQuests();
00023         }
00024
00025         protected void Update()
00026         {
00027             if (GetButtonDown("Close Menu/Inventory"))
00028                 ToggleInventory(this);
00029         }
00030
00031         public void AddQuests()
00032         {
00033             for (int i = 0; i < scrollObject.transform.childCount; i++)
00034             {
00035                 Destroy(scrollObject.transform.GetChild(i).gameObject);
00036             }
00037
00038             var compleatedQuests = Quests.ReturnCompleatedQuests();
00039             var currentQuests = Quests.ReturnCurrentQuests();
00040             var compleatClaimedQuests = Quests.
00041                 ReturnCompleatedClaimedQuests();
00042
00043             for (int i = 0; i < compleatedQuests.Count; i++)
00044             {
00045                 var go = Instantiate(questButton);
00046
00047                 go.transform.SetParent(scrollObject.transform, false);
00048
00049                 go.gameObject.SetActive(true);
00050
00051                 go.GetComponentInChildren<Text>().text = (string)compleatedQuests[compleatedQuests.Keys.
00052                     ElementAt(i)][compleatedQuests[compleatedQuests.Keys.ElementAt(i)].Length - 1];
00053                 go.GetComponentInChildren<Text>().color = Color.yellow;
00054
00055                 var key = compleatedQuests.Keys.ElementAt(i);
00056                 go.GetComponent<Button>().onClick.AddListener(() => QuestAchieved(key));
00057
00058             for (int i = 0; i < currentQuests.Count; i++)
00059             {
00060                 var go = Instantiate(questButton);
00061
00062                 go.transform.SetParent(scrollObject.transform, false);
00063
00064                 go.gameObject.SetActive(true);
00065
00066                 go.GetComponentInChildren<Text>().text = (string)currentQuests[currentQuests.Keys.ElementAt(
00067                     i)][currentQuests[currentQuests.Keys.ElementAt(i)].Length - 1];
00068                 go.GetComponentInChildren<Text>().color = Color.red;
00069
00070             for (int i = 0; i < compleatClaimedQuests.Count; i++)
00071             {
00072                 var go = Instantiate(questButton);
00073

```

```

00074         go.transform.SetParent(scrollObject.transform, false);
00075
00076         go.gameObject.SetActive(true);
00077
00078         go.GetComponentInChildren<Text>().text = (string)compleatClaimedQuests[
00079             compleatClaimedQuests.Keys.ElementAt(i)][compleatClaimedQuests[compleatClaimedQuests.Keys.ElementAt(i)].Length - 1];
00080         go.GetComponentInChildren<Text>().color = Color.green;
00081     }
00082 }
00083
00084     private void QuestAchieved(string key)
00085     {
00086         Quests.ClaimQuest(key);
00087         AddQuests();
00088         Serialization.Serialization.SaveQuests();
00089     }
00090
00091     #region Open/Close Inventory
00092     public override void ToggleInventory(Inventory inv)
00093     {
00094         thisInventoryOpen = !thisInventoryOpen;
00095
00096         isAnotherInventoryOpen = thisInventoryOpen;
00097
00098         if (this.gameObject.activeInHierarchy && !thisInventoryOpen)
00099         {
00100             chestOpen = false;
00101
00102             /* removes all of the items from thsi inventory
00103
00104             /* tells item that inventory has been closed
00105             myItem.OpenItemInvenetory();
00106             Cursor.lockState = CursorLockMode.Locked;
00107             Cursor.visible = false;
00108             /* destroys this as it is not needed
00109             Destroy(this.gameObject);
00110         }
00111         else
00112         {
00113             chestOpen = true;
00114
00115             /* hides and locks the cursor
00116             Cursor.lockState = CursorLockMode.None;
00117             Cursor.visible = true;
00118         }
00119     }
00120     #endregion
00121 }
00122 }
```

7.69 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/ItemsInInventory.cs File Reference

Classes

- class BeeGame.Inventory.ItemsInInventory

Class that holds all of the items in the inventory. Can be serialized so inventory may be saved

Namespaces

- namespace BeeGame.Inventory

7.70 ItemsInInventory.cs

```

00001 using System;
00002 using BeeGame.Items;
00003
00004 namespace BeeGame.Inventory
00005 {
00006     [Serializable]
00007     public class ItemsInInventory
```

```

00011     {
00015         public Item[] itemsInInventory;
00016
00021         public ItemsInInventory(int numberOfInventorySlots)
00022         {
00023             itemsInInventory = new Item[numberOfInventorySlots];
00024         }
00025
00031         public void AddItem(int index, Item item)
00032         {
00033             itemsInInventory[index] = item;
00034         }
00035
00041         public bool AddItem(Item item)
00042         {
00043             for (int i = 0; i < itemsInInventory.Length; i++)
00044             {
00045                 if (itemsInInventory[i] == null)
00046                 {
00047                     itemsInInventory[i] = item;
00048                     return true;
00049                 }
00050                 if (itemsInInventory[i] == item && itemsInInventory[i].itemStackCount + 1 <=
itemsInInventory[i].maxStackCount)
00051                 {
00052                     itemsInInventory[i].itemStackCount++;
00053                     return true;
00054                 }
00055             }
00056
00057             return false;
00058         }
00059     }
00060 }
```

7.71 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Player Inventory/← PlayerInventory.cs File Reference

Classes

- class BeeGame.Inventory.Player_Inventory.PlayerInventory
Controls the player inventory

Namespaces

- namespace BeeGame.Inventory.Player_Inventory

7.72 PlayerInventory.cs

```

00001 using UnityEngine;
00002 using BeeGame.Items;
00003 using BeeGame.Core;
00004 using BeeGame.Quest;
00005
00006 namespace BeeGame.Inventory.Player_Inventory
00007 {
00011     public class PlayerInventory : Inventory
00012     {
00013         #region Data
00014         public GameObject playerInventory;
00018         #endregion
00019
00020         #region Init
00021         protected void Awake()
00025         {
00026             SetPlayerInventory();
00027             inventoryName = "PlayerInventory";
00028             Serialization.Serialization.DeSerializeInventory(this, inventoryName);
00029         }
00030
00034         public virtual void SetPlayerInventory(int size = 36)
00035         {
```

```

00036         if (!InventorySet())
00037             SetInventorySize(size);
00038     }
00039 #endregion
00040
00041     protected void Update()
00042     {
00043         UpdateBase();
00044
00045         /* checks if the inventory should be opened/closed
00046         if ((thisInventoryOpen || !playerInventory.activeInHierarchy) && !
00047             THInput.chestOpen && THInput.GetButtonDown("Player Inventory"))
00048         {
00049             if (THInput.blockInventoryJustClosed)
00050             {
00051                 THInput.blockInventoryJustClosed = false;
00052                 return;
00053             }
00054             else
00055             {
00056                 OpenPlayerInventory();
00057             }
00058         }
00059
00060         /* dont pickup items if the inventory is open
00061         if (THInput.isAnotherInventoryOpen)
00062             return;
00063
00064         /* checks if something should be picked up and put into the inventory
00065         RaycastHit[] hit = Physics.SphereCastAll(transform.position, 1f, transform.forward);
00066
00067         for (int i = hit.Length - 1; i >= 0; i--)
00068         {
00069             if (hit[i].collider.GetComponent<ItemGameObject>())
00070                 PickupItem(hit[i].collider.GetComponent<ItemGameObject>());
00071         }
00072
00073     }
00074
00075     #region Hotbar
00076     public void SelectedSlot(int index)
00077     {
00078         for (int i = 0; i < slots.Length; i++)
00079         {
00080             slots[i].selectedSlot = false;
00081         }
00082
00083         slots[index].selectedSlot = true;
00084     }
00085
00086     public bool GetItemFromHotBar(int slotIndex, out Item outItem)
00087     {
00088         /* get the item
00089         outItem = GetAllItems().itemsInInventory[slotIndex];
00090
00091         if (outItem == null)
00092             return false;
00093
00094         /* if the item is placeable and is not null remove 1 from the inventory as it is assumed it is
00095         about to be placed in the world
00096         if(outItem.placeable)
00097             RemoveItemFromInventory(slotIndex);
00098
00099         return outItem.placeable;
00100     }
00101 #endregion
00102
00103     #region Interact With Inventory
00104     internal void OpenPlayerInventory()
00105     {
00106         if (floatingItem != null)
00107             return;
00108         thisInventoryOpen = !thisInventoryOpen;
00109         playerInventory.SetActive(!playerInventory.activeInHierarchy);
00110         THInput.isAnotherInventoryOpen = !
00111         THInput.isAnotherInventoryOpen;
00112
00113         /* hides/shows the mouse depending on if the inventory is open or not
00114         if (playerInventory.activeInHierarchy)
00115         {
00116             Cursor.lockState = CursorLockMode.None;
00117             Cursor.visible = true;
00118         }
00119         else
00120         {
00121             Cursor.visible = false;
00122             Cursor.lockState = CursorLockMode.Locked;
00123
00124     }
00125
00126     /* hides/shows the mouse depending on if the inventory is open or not
00127     if (playerInventory.activeInHierarchy)
00128     {
00129         Cursor.lockState = CursorLockMode.None;
00130         Cursor.visible = true;
00131     }
00132     else
00133     {
00134         Cursor.visible = false;
00135         Cursor.lockState = CursorLockMode.Locked;
00136
00137     }
00138
00139     #endregion
00140
00141     protected void OnDrawGizmos()
00142     {
00143         if (thisInventoryOpen)
00144             Gizmos.DrawWireSphere(transform.position, 1f);
00145     }
00146
00147     protected void OnGUI()
00148     {
00149         if (thisInventoryOpen)
00150             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00151     }
00152
00153     protected void OnGUI()
00154     {
00155         if (thisInventoryOpen)
00156             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00157     }
00158
00159     protected void OnGUI()
00160     {
00161         if (thisInventoryOpen)
00162             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00163     }
00164
00165     protected void OnGUI()
00166     {
00167         if (thisInventoryOpen)
00168             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00169     }
00170
00171     protected void OnGUI()
00172     {
00173         if (thisInventoryOpen)
00174             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00175     }
00176
00177     protected void OnGUI()
00178     {
00179         if (thisInventoryOpen)
00180             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00181     }
00182
00183     protected void OnGUI()
00184     {
00185         if (thisInventoryOpen)
00186             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00187     }
00188
00189     protected void OnGUI()
00190     {
00191         if (thisInventoryOpen)
00192             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00193     }
00194
00195     protected void OnGUI()
00196     {
00197         if (thisInventoryOpen)
00198             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00199     }
00200
00201     protected void OnGUI()
00202     {
00203         if (thisInventoryOpen)
00204             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00205     }
00206
00207     protected void OnGUI()
00208     {
00209         if (thisInventoryOpen)
00210             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00211     }
00212
00213     protected void OnGUI()
00214     {
00215         if (thisInventoryOpen)
00216             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00217     }
00218
00219     protected void OnGUI()
00220     {
00221         if (thisInventoryOpen)
00222             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00223     }
00224
00225     protected void OnGUI()
00226     {
00227         if (thisInventoryOpen)
00228             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00229     }
00230
00231     protected void OnGUI()
00232     {
00233         if (thisInventoryOpen)
00234             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00235     }
00236
00237     protected void OnGUI()
00238     {
00239         if (thisInventoryOpen)
00240             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00241     }
00242
00243     protected void OnGUI()
00244     {
00245         if (thisInventoryOpen)
00246             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00247     }
00248
00249     protected void OnGUI()
00250     {
00251         if (thisInventoryOpen)
00252             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00253     }
00254
00255     protected void OnGUI()
00256     {
00257         if (thisInventoryOpen)
00258             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00259     }
00260
00261     protected void OnGUI()
00262     {
00263         if (thisInventoryOpen)
00264             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00265     }
00266
00267     protected void OnGUI()
00268     {
00269         if (thisInventoryOpen)
00270             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00271     }
00272
00273     protected void OnGUI()
00274     {
00275         if (thisInventoryOpen)
00276             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00277     }
00278
00279     protected void OnGUI()
00280     {
00281         if (thisInventoryOpen)
00282             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00283     }
00284
00285     protected void OnGUI()
00286     {
00287         if (thisInventoryOpen)
00288             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00289     }
00290
00291     protected void OnGUI()
00292     {
00293         if (thisInventoryOpen)
00294             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00295     }
00296
00297     protected void OnGUI()
00298     {
00299         if (thisInventoryOpen)
00300             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00301     }
00302
00303     protected void OnGUI()
00304     {
00305         if (thisInventoryOpen)
00306             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00307     }
00308
00309     protected void OnGUI()
00310     {
00311         if (thisInventoryOpen)
00312             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00313     }
00314
00315     protected void OnGUI()
00316     {
00317         if (thisInventoryOpen)
00318             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00319     }
00320
00321     protected void OnGUI()
00322     {
00323         if (thisInventoryOpen)
00324             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00325     }
00326
00327     protected void OnGUI()
00328     {
00329         if (thisInventoryOpen)
00330             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00331     }
00332
00333     protected void OnGUI()
00334     {
00335         if (thisInventoryOpen)
00336             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00337     }
00338
00339     protected void OnGUI()
00340     {
00341         if (thisInventoryOpen)
00342             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00343     }
00344
00345     protected void OnGUI()
00346     {
00347         if (thisInventoryOpen)
00348             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00349     }
00350
00351     protected void OnGUI()
00352     {
00353         if (thisInventoryOpen)
00354             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00355     }
00356
00357     protected void OnGUI()
00358     {
00359         if (thisInventoryOpen)
00360             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00361     }
00362
00363     protected void OnGUI()
00364     {
00365         if (thisInventoryOpen)
00366             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00367     }
00368
00369     protected void OnGUI()
00370     {
00371         if (thisInventoryOpen)
00372             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00373     }
00374
00375     protected void OnGUI()
00376     {
00377         if (thisInventoryOpen)
00378             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00379     }
00380
00381     protected void OnGUI()
00382     {
00383         if (thisInventoryOpen)
00384             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00385     }
00386
00387     protected void OnGUI()
00388     {
00389         if (thisInventoryOpen)
00390             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00391     }
00392
00393     protected void OnGUI()
00394     {
00395         if (thisInventoryOpen)
00396             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00397     }
00398
00399     protected void OnGUI()
00400     {
00401         if (thisInventoryOpen)
00402             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00403     }
00404
00405     protected void OnGUI()
00406     {
00407         if (thisInventoryOpen)
00408             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00409     }
00410
00411     protected void OnGUI()
00412     {
00413         if (thisInventoryOpen)
00414             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00415     }
00416
00417     protected void OnGUI()
00418     {
00419         if (thisInventoryOpen)
00420             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00421     }
00422
00423     protected void OnGUI()
00424     {
00425         if (thisInventoryOpen)
00426             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00427     }
00428
00429     protected void OnGUI()
00430     {
00431         if (thisInventoryOpen)
00432             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00433     }
00434
00435     protected void OnGUI()
00436     {
00437         if (thisInventoryOpen)
00438             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00439     }
00440
00441     protected void OnGUI()
00442     {
00443         if (thisInventoryOpen)
00444             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00445     }
00446
00447     protected void OnGUI()
00448     {
00449         if (thisInventoryOpen)
00450             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00451     }
00452
00453     protected void OnGUI()
00454     {
00455         if (thisInventoryOpen)
00456             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00457     }
00458
00459     protected void OnGUI()
00460     {
00461         if (thisInventoryOpen)
00462             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00463     }
00464
00465     protected void OnGUI()
00466     {
00467         if (thisInventoryOpen)
00468             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00469     }
00470
00471     protected void OnGUI()
00472     {
00473         if (thisInventoryOpen)
00474             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00475     }
00476
00477     protected void OnGUI()
00478     {
00479         if (thisInventoryOpen)
00480             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00481     }
00482
00483     protected void OnGUI()
00484     {
00485         if (thisInventoryOpen)
00486             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00487     }
00488
00489     protected void OnGUI()
00490     {
00491         if (thisInventoryOpen)
00492             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00493     }
00494
00495     protected void OnGUI()
00496     {
00497         if (thisInventoryOpen)
00498             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00499     }
00500
00501     protected void OnGUI()
00502     {
00503         if (thisInventoryOpen)
00504             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00505     }
00506
00507     protected void OnGUI()
00508     {
00509         if (thisInventoryOpen)
00510             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00511     }
00512
00513     protected void OnGUI()
00514     {
00515         if (thisInventoryOpen)
00516             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00517     }
00518
00519     protected void OnGUI()
00520     {
00521         if (thisInventoryOpen)
00522             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00523     }
00524
00525     protected void OnGUI()
00526     {
00527         if (thisInventoryOpen)
00528             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00529     }
00530
00531     protected void OnGUI()
00532     {
00533         if (thisInventoryOpen)
00534             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00535     }
00536
00537     protected void OnGUI()
00538     {
00539         if (thisInventoryOpen)
00540             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00541     }
00542
00543     protected void OnGUI()
00544     {
00545         if (thisInventoryOpen)
00546             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00547     }
00548
00549     protected void OnGUI()
00550     {
00551         if (thisInventoryOpen)
00552             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00553     }
00554
00555     protected void OnGUI()
00556     {
00557         if (thisInventoryOpen)
00558             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00559     }
00560
00561     protected void OnGUI()
00562     {
00563         if (thisInventoryOpen)
00564             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00565     }
00566
00567     protected void OnGUI()
00568     {
00569         if (thisInventoryOpen)
00570             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00571     }
00572
00573     protected void OnGUI()
00574     {
00575         if (thisInventoryOpen)
00576             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00577     }
00578
00579     protected void OnGUI()
00580     {
00581         if (thisInventoryOpen)
00582             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00583     }
00584
00585     protected void OnGUI()
00586     {
00587         if (thisInventoryOpen)
00588             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00589     }
00590
00591     protected void OnGUI()
00592     {
00593         if (thisInventoryOpen)
00594             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00595     }
00596
00597     protected void OnGUI()
00598     {
00599         if (thisInventoryOpen)
00600             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00601     }
00602
00603     protected void OnGUI()
00604     {
00605         if (thisInventoryOpen)
00606             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00607     }
00608
00609     protected void OnGUI()
00610     {
00611         if (thisInventoryOpen)
00612             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00613     }
00614
00615     protected void OnGUI()
00616     {
00617         if (thisInventoryOpen)
00618             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00619     }
00620
00621     protected void OnGUI()
00622     {
00623         if (thisInventoryOpen)
00624             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00625     }
00626
00627     protected void OnGUI()
00628     {
00629         if (thisInventoryOpen)
00630             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00631     }
00632
00633     protected void OnGUI()
00634     {
00635         if (thisInventoryOpen)
00636             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00637     }
00638
00639     protected void OnGUI()
00640     {
00641         if (thisInventoryOpen)
00642             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00643     }
00644
00645     protected void OnGUI()
00646     {
00647         if (thisInventoryOpen)
00648             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00649     }
00650
00651     protected void OnGUI()
00652     {
00653         if (thisInventoryOpen)
00654             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00655     }
00656
00657     protected void OnGUI()
00658     {
00659         if (thisInventoryOpen)
00660             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00661     }
00662
00663     protected void OnGUI()
00664     {
00665         if (thisInventoryOpen)
00666             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00667     }
00668
00669     protected void OnGUI()
00670     {
00671         if (thisInventoryOpen)
00672             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00673     }
00674
00675     protected void OnGUI()
00676     {
00677         if (thisInventoryOpen)
00678             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00679     }
00680
00681     protected void OnGUI()
00682     {
00683         if (thisInventoryOpen)
00684             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00685     }
00686
00687     protected void OnGUI()
00688     {
00689         if (thisInventoryOpen)
00690             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00691     }
00692
00693     protected void OnGUI()
00694     {
00695         if (thisInventoryOpen)
00696             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00697     }
00698
00699     protected void OnGUI()
00700     {
00701         if (thisInventoryOpen)
00702             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00703     }
00704
00705     protected void OnGUI()
00706     {
00707         if (thisInventoryOpen)
00708             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00709     }
00710
00711     protected void OnGUI()
00712     {
00713         if (thisInventoryOpen)
00714             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00715     }
00716
00717     protected void OnGUI()
00718     {
00719         if (thisInventoryOpen)
00720             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00721     }
00722
00723     protected void OnGUI()
00724     {
00725         if (thisInventoryOpen)
00726             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00727     }
00728
00729     protected void OnGUI()
00730     {
00731         if (thisInventoryOpen)
00732             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00733     }
00734
00735     protected void OnGUI()
00736     {
00737         if (thisInventoryOpen)
00738             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00739     }
00740
00741     protected void OnGUI()
00742     {
00743         if (thisInventoryOpen)
00744             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00745     }
00746
00747     protected void OnGUI()
00748     {
00749         if (thisInventoryOpen)
00750             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00751     }
00752
00753     protected void OnGUI()
00754     {
00755         if (thisInventoryOpen)
00756             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00757     }
00758
00759     protected void OnGUI()
00760     {
00761         if (thisInventoryOpen)
00762             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00763     }
00764
00765     protected void OnGUI()
00766     {
00767         if (thisInventoryOpen)
00768             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00769     }
00770
00771     protected void OnGUI()
00772     {
00773         if (thisInventoryOpen)
00774             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00775     }
00776
00777     protected void OnGUI()
00778     {
00779         if (thisInventoryOpen)
00780             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00781     }
00782
00783     protected void OnGUI()
00784     {
00785         if (thisInventoryOpen)
00786             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00787     }
00788
00789     protected void OnGUI()
00790     {
00791         if (thisInventoryOpen)
00792             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00793     }
00794
00795     protected void OnGUI()
00796     {
00797         if (thisInventoryOpen)
00798             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00799     }
00800
00801     protected void OnGUI()
00802     {
00803         if (thisInventoryOpen)
00804             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00805     }
00806
00807     protected void OnGUI()
00808     {
00809         if (thisInventoryOpen)
00810             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00811     }
00812
00813     protected void OnGUI()
00814     {
00815         if (thisInventoryOpen)
00816             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00817     }
00818
00819     protected void OnGUI()
00820     {
00821         if (thisInventoryOpen)
00822             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00823     }
00824
00825     protected void OnGUI()
00826     {
00827         if (thisInventoryOpen)
00828             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00829     }
00830
00831     protected void OnGUI()
00832     {
00833         if (thisInventoryOpen)
00834             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00835     }
00836
00837     protected void OnGUI()
00838     {
00839         if (thisInventoryOpen)
00840             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00841     }
00842
00843     protected void OnGUI()
00844     {
00845         if (thisInventoryOpen)
00846             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00847     }
00848
00849     protected void OnGUI()
00850     {
00851         if (thisInventoryOpen)
00852             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00853     }
00854
00855     protected void OnGUI()
00856     {
00857         if (thisInventoryOpen)
00858             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00859     }
00860
00861     protected void OnGUI()
00862     {
00863         if (thisInventoryOpen)
00864             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00865     }
00866
00867     protected void OnGUI()
00868     {
00869         if (thisInventoryOpen)
00870             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00871     }
00872
00873     protected void OnGUI()
00874     {
00875         if (thisInventoryOpen)
00876             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00877     }
00878
00879     protected void OnGUI()
00880     {
00881         if (thisInventoryOpen)
00882             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00883     }
00884
00885     protected void OnGUI()
00886     {
00887         if (thisInventoryOpen)
00888             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00889     }
00890
00891     protected void OnGUI()
00892     {
00893         if (thisInventoryOpen)
00894             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00895     }
00896
00897     protected void OnGUI()
00898     {
00899         if (thisInventoryOpen)
00900             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00901     }
00902
00903     protected void OnGUI()
00904     {
00905         if (thisInventoryOpen)
00906             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00907     }
00908
00909     protected void OnGUI()
00910     {
00911         if (thisInventoryOpen)
00912             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00913     }
00914
00915     protected void OnGUI()
00916     {
00917         if (thisInventoryOpen)
00918             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00919     }
00920
00921     protected void OnGUI()
00922     {
00923         if (thisInventoryOpen)
00924             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00925     }
00926
00927     protected void OnGUI()
00928     {
00929         if (thisInventoryOpen)
00930             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00931     }
00932
00933     protected void OnGUI()
00934     {
00935         if (thisInventoryOpen)
00936             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00937     }
00938
00939     protected void OnGUI()
00940     {
00941         if (thisInventoryOpen)
00942             GUI.Box(new Rect(10, 10, 100, 100), "Inventory");
00943     }
00944
00945     protected void OnGUI()
00946     {
00
```

```

00136         }
00137     }
00138
00143     public void RemoveItemFromInventory(int index)
00144     {
00145         /* if the item is already null nothing needs to be removed
00146         if (GetAllItems().itemsInInventory[index] != null)
00147         {
00148             /* remove 1 item and if that was the last in the stack remove the item from the inventory
00149             GetAllItems().itemsInInventory[index].itemStackCount -= 1;
00150
00151             if (GetAllItems().itemsInInventory[index].itemStackCount <= 0)
00152                 GetAllItems().itemsInInventory[index] = null;
00153
00154             Serialization.Serialization.SerializeInventory(this, inventoryName);
00155         }
00156     }
00157
00162     void PickupItem(ItemGameObject item)
00163     {
00164         item.item.itemStackCount = 1;
00165
00166         /* if the item can be added to the inventory do that
00167         if (AddItemToInventory(item.item))
00168         {
00169             QuestEvents.CallItemPickupEvent(item.
item.GetHashCode());
00170             /* if the item was added destroy its gameobject and save the inventory
00171             Destroy(item.gameObject);
00172             Serialization.Serialization.SerializeInventory(this, inventoryName);
00173         }
00174     }
00175 #endregion
00176 }
00177 }
```

7.73 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/AbstractItem.cs **File Reference**

Classes

- class [BeeGame.Items.AbstractItem](#)

Does this need to exist?

Namespaces

- namespace [BeeGame.Items](#)

7.74 AbstractItem.cs

```

00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using System.Text;
00005
00006 namespace BeeGame.Items
00007 {
00011     [Serializable]
00012     public abstract class AbstractItem
00013     {
00014         public abstract string GetItemName();
00015         public abstract string GetItemID();
00016         public abstract override int GetHashCode();
00017     }
00018 }
```

7.75 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/ApplyColour.cs File Reference

Classes

- class [BeeGame.Items.ApplyColour](#)
Applies a given colour to a gameobject

Namespaces

- namespace [BeeGame.Items](#)

7.76 ApplyColour.cs

```

00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using System.Text;
00005 using UnityEngine;
00006
00007 namespace BeeGame.Items
00008 {
00012     public class ApplyColour : MonoBehaviour
00013     {
00014         #region Data
00015         public Color colour;
00025         public GameObject[] objects;
00026         #endregion
00027
00028         #region Unity Methods
00029         private void Start()
00033         {
00034             /* applies the correct colour to each object in the array
00035             for (int i = 0; i < objects.Length; i++)
00036             {
00037                 objects[i].GetComponent<Renderer>().material.SetColor("_OverlayColour", colour);
00038             }
00039         }
00040         #endregion
00041     }
00042 }
```

7.77 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/Bee.cs File Reference

Classes

- class [BeeGame.Items.Bee](#)
The bee item
- class [BeeGame.Items.QueenBee](#)
- class [BeeGame.Items.NormalBee](#)

Namespaces

- namespace [BeeGame.Items](#)

7.78 Bee.cs

```

00001 using System;
00002 using System.Globalization;
00003 using UnityEngine;
00004 using BeeGame.Core;
00005 using BeeGame.Core.Enums;
00006 using BeeGame.Core.Dictionaries;
00007
00008 namespace BeeGame.Items
00009 {
00013     [Serializable]
00014     public class Bee : Item
00015     {
00016         #region Data
00017         public bool canSeeBeeData = false;
00021
00025         public BeeType beeType { get; set; }
00029         private BeeType previousBeeType { get; set; }
00033         public override int maxStackCount { get { return maxStack; } }
00034         private int maxStack = 64;
00035
00039         [NonSerialized]
00040         private Sprite itemSprite;
00041
00048         public QueenBee queenBee { get; set; }
00052         public NormalBee normalBee { get; set; }
00053
00054         public new static int ID => 11;
00055     #endregion
00056
00057     #region Constructors
00058     public Bee()
00059     {
00060         usesGameObject = true;
00061         normalBee = new NormalBee();
00062     }
00063
00064
00070         public Bee(BeeType beeType, NormalBee normalBee) : base(new CultureInfo("en-US",
00071             false).TextInfo.ToTitleCase($"{normalBee.pSpecies} {beeType}.ToLower()))
00072     {
00073         usesGameObject = true;
00073         if (beeType == BeeType.PRINCESS || beeType == BeeType.QUEEN)
00074             maxStack = 1;
00075         this.beeType = beeType;
00076         this.normalBee = normalBee;
00077     }
00078
00084         public Bee(BeeType beeType, QueenBee queenBee) : base(new CultureInfo("en-US",
00085             false).TextInfo.ToTitleCase($"{queenBee.queen.pSpecies} {beeType}.ToLower()))
00086     {
00087         usesGameObject = true;
00087         if (beeType == BeeType.PRINCESS || beeType == BeeType.QUEEN)
00088             maxStack = 1;
00089         this.beeType = beeType;
00090         this.queenBee = queenBee;
00091     }
00092 #endregion
00093
00094     #region Item Overrides
00095     public override Sprite GetItemSprite()
00096     {
00101         /* if the bee has not change in any way dont rebuild the sprite as that takes time
00102         if(previousBeeType == beeType && itemSprite != null)
00103         {
00104             return itemSprite;
00105         }
00106
00107         previousBeeType = beeType;
00108
00109         /* set the correct sprite and colour
00110         if (beeType == BeeType.QUEEN)
00111         {
00112             /* avoids the crown, black body, yellow body, and both colours of the wings
00113             Color[] colorsToAvoid = { new Color(0, 0, 0), new Color(232f, 200f, 42f, 255f) / 255f, new
00113             Color(232f, 213f, 106f, 255f) / 255f, new Color(156f, 146f, 130f, 255f) / 255f, new Color(225f, 223f, 219f,
00113             255f) / 255f };
00114             return itemSprite = SpriteDictionary.GetSprite("Queen").
00114             ColourSprite(BeeDictionaries.GetBeeColour((BeeSpecies)(queenBee?.queen.pSpecies))
00114             , coloursToAvoid: colorsToAvoid);
00115         }
00116         else if (beeType == BeeType.PRINCESS)
00117         {
00118             /* avoids the tiara, black body, yellow body, and both colours of the wings
00119             Color[] colorsToAvoid = { new Color(0, 0, 0), new Color(191f, 195f, 45f, 255f) / 255f, new

```

```

        Color(191f, 195f, 44f, 255f) / 255f, new Color(156f, 146f, 130f, 255f) / 255f, new Color(225f, 223f, 219f, 2
00120      55f) / 255f, new Color(232f, 200, 42, 255f) / 255f };
00121      return itemSprite = SpriteDictionary.GetSprite("Princess");
00122  }
00123  {
00124      /* avoids the block body, yellow body, and both wing colours
00125      Color[] colorsToAvoid = { new Color(0, 0, 0), new Color(156f, 146f, 130f, 255f) / 255f, new
00126      Color(225f, 223f, 219f, 255f) / 255f, new Color(232f, 200, 42, 255f) / 255f };
00127      return itemSprite = SpriteDictionary.GetSprite("Drone");
00128  }
00129
00130  public override GameObject GetGameObject()
00131  {
00132      var go = PrefabDictionary.GetPrefab("Bee");
00133
00134      go.GetComponent<SetBeeGOColours>().colour =
00135      BeeDictionarys.GetBeeColour(normalBee?.pSpecies ?? queenBee.queen.pSpecies);
00136
00137      go.GetComponent<SetBeeGOColours>().beeType = beeType;
00138  }
00139
00140  public override string GetItemID()
00141  {
00142      return $"{GetHashCode()}\\{(int)beeType}{queenBee?.GetHashCode() ?? normalBee?.GetHashCode()}";
00143
00144  #endregion
00145
00146  #region Bee Stuff
00147  public static void ConvertToQueen(Bee princess, NormalBee drone)
00148  {
00149      ConvertToQueen(ref princess, drone);
00150
00151  }
00152
00153
00154  public static void ConvertToQueen(ref Bee princess,
00155  NormalBee drone)
00156  {
00157      princess.beeType = BeeType.QUEEN;
00158      princess.queenBee = new QueenBee(princess.normalBee, drone);
00159      princess.normalBee = null;
00160
00161      princess.itemName = new CultureInfo("en-US", false).TextInfo.ToTitleCase($""
00162      {princess.queenBee.queen.pSpecies} {princess.beeType}.ToLower());
00163  }
00164
00165  public Bee MakeBeeWithStats(BeeType beeType =
00166  BeeType.DRONE, BeeSpecies species = BeeSpecies.FOREST,
00167  BeeLifeSpan lifespan = BeeLifeSpan.NORMAL, uint fertility = 2,
00168  BeeEffect effect = BeeEffect.NONE, BeeProductionSpeed prodSpeed =
00169  BeeProductionSpeed.NORMAL)
00170  {
00171      NormalBee normBee = new NormalBee()
00172      {
00173          pSpecies = species,
00174          pLifespan = lifespan,
00175          pFertility = fertility,
00176          pProdSpeed = prodSpeed,
00177          pEffect = effect,
00178          sEffect = effect,
00179          sFertility = fertility,
00180          sLifespan = lifespan,
00181          sProdSpeed = prodSpeed,
00182          sSpecies = species
00183      };
00184
00185      switch (beeType)
00186      {
00187          case BeeType.QUEEN:
00188              return new Bee(beeType, new QueenBee(normBee, normBee));
00189          default:
00190              return new Bee(beeType, normBee);
00191      }
00192  }
00193
00194  #endregion
00195
00196  #region Overrides
00197  public override int GetHashCode()
00198  {
00199      return ID;
00200  }
00201
00202  #endregion

```

```

00220      }
00221
00222     [Serializable]
00223     public class QueenBee
00224     {
00225         /* Properties so that they can be copied by reflection as it does not copy variables only
00226         properties
00227         public NormalBee queen { get; set; }
00228         public NormalBee drone { get; set; }
00229
00230         public QueenBee() { }
00231
00232         public QueenBee(NormalBee princess, NormalBee drone)
00233         {
00234             this.princess = princess;
00235             this.drone = drone;
00236         }
00237
00238         public override int GetHashCode()
00239         {
00240             unchecked
00241             {
00242                 return (int)Int64.Parse($"{queen.GetHashCode() ^ drone.GetHashCode()}");
00243             }
00244         }
00245     }
00246
00247     [Serializable]
00248     public class NormalBee
00249     {
00250         #region Phenotype
00251         /* Currently shown traits of the bee
00252
00253         public BeeSpecies pSpecies;
00254         public BeeLifeSpan pLifespan;
00255         public uint pFertility;
00256         public BeeEffect pEffect;
00257         public BeeProductionSpeed pProdSpeed;
00258         #endregion
00259
00260         #region Secondary
00261         /* Traits of the bee used in the bees combination
00262
00263         public BeeSpecies sSpecies;
00264         public BeeLifeSpan sLifespan;
00265         public uint sFertility;
00266         public BeeEffect sEffect;
00267         public BeeProductionSpeed sProdSpeed;
00268         #endregion Secondary
00269
00270         public override int GetHashCode()
00271         {
00272             unchecked
00273             {
00274                 //int hashCode = 13;
00275
00276                 var temp = $"((int)pSpecies){(int)sSpecies}{(int)pLifespan}{(int)sLifespan}{(int)pFertility}{(int)sFertility}{(int)pEffect}{(int)sEffect}{(int)pProdSpeed}{(int)sProdSpeed}";
00277
00278                 var hashCode = (int)(Int64.Parse(temp) ^ (127 * 13) / 159);
00279
00280                 hashCode += ((int)pSpecies ^ (int)pLifespan ^ (int)pFertility ^ (int)pEffect ^ (int)pProdSpeed) * 127;
00281                 hashCode += ((int)sSpecies ^ (int)sLifespan ^ (int)sFertility ^ (int)sEffect ^ (int)sProdSpeed) * 307;
00282
00283                 return hashCode;
00284             }
00285         }
00286     }
00287 }
00288 }
```

7.79 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/BeeAlyzer.cs File Reference

Classes

- class BeeGame.Items.BeeAlyzer

Namespaces

- namespace BeeGame.Items

7.80 BeeAlyzer.cs

```

00001 using System;
00002 using BeeGame.Inventory;
00003 using UnityEngine;
00004 using BeeGame.Core.Dictionaries;
00005
00006 namespace BeeGame.Items
00007 {
00008     [Serializable]
00009     public class BeeAlyzer : Item
00010     {
00011         #region Data
00012         public new int ID = 13;
00016
00020         public override int maxStackCount
00021         {
00022             get
00023             {
00024                 return 1;
00025             }
00026         }
00027
00031         public override bool placeable
00032         {
00033             get
00034             {
00035                 return false;
00036             }
00037         }
00038
00042         [NonSerialized]
00043         public GameObject myInventory;
00044     #endregion
00045
00046     #region Constructors
00047     public BeeAlyzer() : base("BeeAlyzer")
00048     {
00049     }
00050
00055     public BeeAlyzer(string s) : base(s)
00056     {}
00057     #endregion
00058
00059     #region ItemInventory
00060     public virtual void OpenItemInvnetory(Inventory.Inventory playerInventory = null)
00065     {
00066         if (myInventory == null)
00067         {
00068             /* makes the inventory
00069             myInventory = (GameObject)UnityEngine.Object.Instantiate(
UnityEngine.Resources.Load("Prefabs/BeeAlyzerInventory"));
00070
00071             /* opens the inventory and gives it the players inventory
00072             myInventory.GetComponent<BeeAlyzerInventory>().ToggleInventory(
playerInventory);
00073             myInventory.GetComponent<BeeAlyzerInventory>().myItem = this;
00074         }
00075         else
00076         {
00077             myInventory = null;
00078         }
00079     }
00080     #endregion
00081
00082     #region Item Overrides
00083     public override void InteractWithItem(Inventory.Inventory playerInventory)
00088     {
00089         OpenItemInvnetory(playerInventory);
00090     }
00091
00096     public override bool InteractWithObject()
00097     {
00098         return true;
00099     }
00100
00105     public override Sprite GetItemSprite()
00106     {

```

```

00107         return SpriteDictionary.GetSprite("BeeAlyzer");
00108     }
00109
00114     public override string GetItemID()
00115     {
00116         return $"{GetHashCode()}";
00117     }
00118 #endregion
00119
00120     #region Overrides
00121     public override int GetHashCode()
00122     {
00123         return ID;
00124     }
00125 #endregion
00126 }
00127 }
```

7.81 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/GameObject.cs File Reference

Classes

- class BeeGame.Items.ItemGameObject
Interface between item and unity gameobjects

Namespaces

- namespace BeeGame.Items

7.82 ItemGameObject.cs

```

00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using System.Text;
00005 using BeeGame.Terrain.Chunks;
00006 using BeeGame.Blocks;
00007 using UnityEngine;
00008
00009 namespace BeeGame.Items
00010 {
00014     [RequireComponent(typeof(Rigidbody))]
00015     [RequireComponent(typeof(MeshFilter))]
00016     [RequireComponent(typeof(MeshRenderer))]
00017     [RequireComponent(typeof(BoxCollider))]
00018     public class ItemGameObject : MonoBehaviour
00019     {
00023         public Item item;
00027         public GameObject go;
00028
00032         private void Start()
00033         {
00034             if (!item.usesGameObject)
00035                 MakeMesh();
00036
00037             if (item.usesGameObject)
00038             {
00039                 Instantiate(item.GetGameObject(), transform, false);
00040                 transform.localScale = new Vector3(0.5f, 0.5f, 0.5f);
00041             }
00042         }
00043
00047         private void Update()
00048         {
00049             if (transform.position.y < -100)
00050             {
00051                 Destroy(gameObject);
00052             }
00053         }
00058         void MakeMesh()
```

```

00059      {
00060          MeshData meshData = new MeshData();
00061          if(item != null)
00062              meshData = item.ItemMesh(0, 0, 0, meshData);
00063
00064          Mesh mesh = new Mesh()
00065          {
00066              vertices = meshData.verts.ToArray(),
00067              triangles = meshData.tris.ToArray(),
00068              uv = meshData.uv.ToArray()
00069          };
00070
00071          mesh.RecalculateNormals();
00072
00073          GetComponent<MeshFilter>().mesh = mesh;
00074      }
00075  }
00076 }
```

7.83 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/GameObjectStuff/SetBeeGOColours.cs File Reference

Classes

- class BeeGame.Items.SetBeeGOColours

Namespaces

- namespace BeeGame.Items

7.84 SetBeeGOColours.cs

```

00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using System.Text;
00005 using BeeGame.Core.Enums;
00006 using BeeGame.Core.Dictionaries;
00007 using UnityEngine;
00008
00009 namespace BeeGame.Items
00010 {
00011     public class SetBeeGOColours : MonoBehaviour
00012     {
00013         public GameObject[] objects;
00014         public GameObject crown;
00015         public GameObject tiara;
00016         public BeeType beeType;
00017         public Color colour;
00018
00019         protected void Start()
00020         {
00021             switch (beeType)
00022             {
00023                 case BeeType.DRONE:
00024                     crown.SetActive(false);
00025                     tiara.SetActive(false);
00026                     break;
00027                 case BeeType.PRINCESS:
00028                     crown.SetActive(false);
00029                     break;
00030             }
00031
00032             foreach (var item in objects)
00033             {
00034                 item.GetComponent<Renderer>().materialSetColor("_Color", colour);
00035             }
00036         }
00037     }
00038 }
```

7.85 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/Honey.cs File Reference

Classes

- class [BeeGame.Items.Honey](#)

Namespaces

- namespace [BeeGame.Items](#)

7.86 Honey.cs

```
00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using System.Text;
00005
00006 namespace BeeGame.Items
00007 {
00008     public class Honey : Item
00009     {
00010         public new static int ID => 14;
00011
00012         public override string ToString()
00013         {
00014             return $"{itemName} \\" {GetItemID()}";
00015         }
00016
00017         public override string GetItemID()
00018         {
00019             return GetHashCode().ToString();
00020         }
00021
00022         public override int GetHashCode()
00023         {
00024             return ID;
00025         }
00026     }
00027 }
```

7.87 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/HoneyComb.cs File Reference

Classes

- class [BeeGame.Items.HoneyComb](#)
Honey comb item produced by bees

Namespaces

- namespace [BeeGame.Items](#)

7.88 HoneyComb.cs

```

00001 using System;
00002 using System.Globalization;
00003 using BeeGame.Core;
00004 using BeeGame.Core.Enums;
00005 using BeeGame.Core.Dictionaries;
00006 using UnityEngine;
00007
00008 namespace BeeGame.Items
00009 {
00013     [Serializable]
00014     public class HoneyComb : Item
00015     {
00016         #region Data
00017         public HoneyCombType type { get; set; }
00021
00025         public Color CombColour
00026         {
00027             get
00028             {
00029                 return BeeDictionaries.GetCombColour(type);
00030             }
00031         }
00032
00036         [NonSerialized]
00037         private Sprite itemSprite;
00038
00039         public new static int ID => 12;
00040         #endregion
00041
00042         #region Constructors
00043         public HoneyComb() : base(new CultureInfo("en-US", false).TextInfo.ToTitleCase($"HoneyCombType.HONEY") Comb".ToLower()))
00044         {
00045             usesGameObject = true;
00046             type = HoneyCombType.HONEY;
00047         }
00048
00056         public HoneyComb(HoneyCombType type) : base(new CultureInfo("en-US", false).
00057             TextInfo.ToTitleCase($"{type.ToString()} Comb".ToLower()))
00058         {
00059             usesGameObject = true;
00060             this.type = type;
00061         }
00062         #endregion
00063
00064         #region Item Overrides
00065         public override Sprite GetItemSprite()
00066         {
00067             return itemSprite ?? (itemSprite = SpriteDictionary.
00068             GetSprite("HoneyComb").ColourSprite(CombColour));
00069         }
00070
00077         public override GameObject GetGameObject()
00078         {
00079             GameObject obj = PrefabDictionary.GetPrefab("HoneyComb");
00080             /* cannot access the instance material from here have to do it on the object
00081             obj.GetComponent<ApplyColour>().colour = CombColour;
00082             return obj;
00083         }
00084
00089         public override string GetItemID()
00090         {
00091             return $"{GetHashCode()}\\{(int)type}";
00092         }
00093         #endregion
00094
00095         #region Overrides
00096         public override int GetHashCode()
00097         {
00098             return ID;
00099         }
00100         #endregion
00101     }
00102 }
```

7.89 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/Item.cs File Reference

Classes

- class BeeGame.Items.Item

Base class for all Items and Blocks in the game

- struct BeeGame.Items.Tile

Position of the items texture

Namespaces

- namespace BeeGame.Items

7.90 Item.cs

```
00001 using System;
00002 using System.IO;
00003 using UnityEngine;
00004 using BeeGame.Core;
00005 using BeeGame.Core.Enums;
00006 using BeeGame.Terrain.Chunks;
00007 using BeeGame.Core.Dictionaries;
00008 using System.Runtime.Serialization.Formatters.Binary;
00009
00010 namespace BeeGame.Items
00011 {
00015     [Serializable]
00016     public class Item : AbstractItem, ICloneable
00017     {
00018         #region Data
00019         internal string itemName { get; set; }
00026         public virtual bool placeable => false;
00030         public bool usesGameObject { get; set; }
00034         private const float tileSize = 0.1f;
00035
00039         public int itemStackCount = 1;
00040
00044         public virtual int maxStackCount => 64;
00045
00046         public static int ID => 0;
00047         #endregion
00048
00049         #region Constructors
00050         public Item()
00051         {
00052             itemName = "TestItem";
00053         }
00054
00055         public Item(string name)
00056         {
00057             itemName = name;
00058         }
00059         #endregion
00060
00061         #region Player Item Interactions
00062         public virtual bool InteractWithObject()
00063         {
00064             return false;
00065         }
00066
00067         public virtual void InteractWithItem(Inventory.Inventory playerInventory)
00068         {
00069             return;
00070         }
00071         #endregion
00072
00073         #region Item Stuff
00074         public virtual GameObject GetGameObject() { return null; }
00079
00084         public override string GetItemID()
00085         {
00086             return $"{GetHashCode()}";
00087         }
00088
00093         public virtual Sprite GetItemSprite()
00094         {
00095             return SpriteDictionary.GetSprite("TestSprite");
00096         }
00097
00102         public override string GetItemName()
00103         {
00104             return $"{itemName}";
00105         }
```

```

00106     #endregion
00107
00108     #region Item Mesh
00109     public virtual Tile TexturePosition(Direction direction)
00115     {
00116         return new Tile() { x = 1, y = 9 };
00117     }
00118
00127     public virtual MeshData ItemMesh(int x, int y, int z,
00128     MeshData meshData)
00128     {
00129         /* adds all faces of the item to the mesh as all faces could be seen at any time
00130         meshData = FaceDataUp(x, y, z, meshData, true, 0.25f);
00131         meshData = FaceDataDown(x, y, z, meshData, true, 0.25f);
00132         meshData = FaceDataNorth(x, y, z, meshData, true, 0.25f);
00133         meshData = FaceDataEast(x, y, z, meshData, true, 0.25f);
00134         meshData = FaceDataSouth(x, y, z, meshData, true, 0.25f);
00135         meshData = FaceDataWest(x, y, z, meshData, true, 0.25f);
00136
00137         return meshData;
00138     }
00139
00145     public virtual Vector2[] FaceUVs(Direction direction)
00146     {
00147         /* only 4 uvs per face
00148         Vector2[] UVs = new Vector2[4];
00149         Tile tilePos = TexturePosition(direction);
00150
00151         /* sets the UVs for each vertex
00152         UVs[0] = new THVector2(tileSize * tilePos.x + tileSize - 0.01f, tileSize * tilePos.
00153         y + 0.01f);
00153         UVs[1] = new THVector2(tileSize * tilePos.x + tileSize - 0.01f, tileSize * tilePos.
00154         y + tileSize - 0.01f);
00154         UVs[2] = new THVector2(tileSize * tilePos.x + 0.01f, tileSize * tilePos.
00155         y + tileSize - 0.01f);
00155         UVs[3] = new THVector2(tileSize * tilePos.x + 0.01f, tileSize * tilePos.
00156         y + 0.01f);
00157
00157         return UVs;
00158     }
00159
00170     protected virtual MeshData FaceDataUp(int x, int y, int z,
00171     MeshData meshData, bool addToRenderMesh = true, float blockSize = 0.5f)
00171     {
00172         /* Adds vertices in a anti-clockwise order
00173         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z +
00174         blockSize), addToRenderMesh, Direction.UP);
00174         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z +
00175         blockSize), addToRenderMesh, Direction.UP);
00175         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z -
00176         blockSize), addToRenderMesh, Direction.UP);
00176         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z -
00177         blockSize), addToRenderMesh, Direction.UP);
00177
00178         /* adds teh tirs for the quad
00179         meshData.AddQuadTriangles(addToRenderMesh);
00180
00181         /* if the data should be added to the render mesh also add the uvs to the mesh
00182         if (addToRenderMesh)
00183             meshData.uv.AddRange(FaceUVs(Direction.UP));
00184
00185         return meshData;
00186     }
00187
00198     protected virtual MeshData FaceDataDown(int x, int y, int z,
00198     MeshData meshData, bool addToRenderMesh = true, float blockSize = 0.5f)
00199     {
00200         /* Adds vertices in a anti-clockwise order
00201         meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z -
00201         blockSize), addToRenderMesh);
00202         meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z -
00203         blockSize), addToRenderMesh);
00203         meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z +
00204         blockSize), addToRenderMesh);
00204         meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z +
00205         blockSize), addToRenderMesh);
00205
00206         /* adds teh tirs for the quad
00207         meshData.AddQuadTriangles(addToRenderMesh);
00208
00209         /* if the data should be added to the render mesh also add the uvs to the mesh
00210         if (addToRenderMesh)
00211             meshData.uv.AddRange(FaceUVs(Direction.DOWN));
00212
00213         return meshData;
00214     }
00215

```

```
00226     protected virtual MeshData FaceDataNorth(int x, int y, int z,
00227         MeshData meshData, bool addToRenderMesh = true, float blockSize = 0.5f)
00228     {
00229         /* Adds vertices in a anti-clockwise order
00230         meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z +
00231             blockSize), addToRenderMesh);
00232         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z +
00233             blockSize), addToRenderMesh);
00234         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z +
00235             blockSize), addToRenderMesh);
00236         meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z +
00237             blockSize), addToRenderMesh);
00238
00239         /* adds teh tirs for the quad
00240         meshData.AddQuadTriangles(addToRenderMesh);
00241
00242         /* if the data should be added to the render mesh also add the uvs to the mesh
00243         if (addToRenderMesh)
00244             meshData.uv.AddRange(FaceUVs(Direction.NORTH));
00245
00246         return meshData;
00247     }
00248
00249     protected virtual MeshData FaceDataEast(int x, int y, int z,
00250         MeshData meshData, bool addToRenderMesh = true, float blockSize = 0.5f)
00251     {
00252         /* Adds vertices in a anti-clockwise order
00253         meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z -
00254             blockSize), addToRenderMesh);
00255         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z -
00256             blockSize), addToRenderMesh);
00257         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z +
00258             blockSize), addToRenderMesh);
00259         meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z +
00260             blockSize), addToRenderMesh);
00261
00262         /* adds teh tirs for the quad
00263         meshData.AddQuadTriangles(addToRenderMesh);
00264
00265         /* if the data should be added to the render mesh also add the uvs to the mesh
00266         if (addToRenderMesh)
00267             meshData.uv.AddRange(FaceUVs(Direction.EAST));
00268
00269         return meshData;
00270     }
00271
00272     protected virtual MeshData FaceDataSouth(int x, int y, int z,
00273         MeshData meshData, bool addToRenderMesh = true, float blockSize = 0.5f)
00274     {
00275         /* Adds vertices in a anti-clockwise order
00276         meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z -
00277             blockSize), addToRenderMesh);
00278         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z -
00279             blockSize), addToRenderMesh);
00280         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z -
00281             blockSize), addToRenderMesh);
00282         meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z -
00283             blockSize), addToRenderMesh);
00284
00285         /* adds teh tirs for the quad
00286         meshData.AddQuadTriangles(addToRenderMesh);
00287
00288         /* if the data should be added to the render mesh also add the uvs to the mesh
00289         if (addToRenderMesh)
00290             meshData.uv.AddRange(FaceUVs(Direction.SOUTH));
00291
00292         return meshData;
00293     }
00294
00295     protected virtual MeshData FaceDataWest(int x, int y, int z,
00296         MeshData meshData, bool addToRenderMesh = true, float blockSize = 0.5f)
00297     {
00298         /* Adds vertices in a anti-clockwise order
00299         meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z +
00300             blockSize), addToRenderMesh);
00301         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z +
00302             blockSize), addToRenderMesh);
00303         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z -
00304             blockSize), addToRenderMesh);
00305         meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z -
00306             blockSize), addToRenderMesh);
00307
00308         /* adds teh tirs for the quad
00309         meshData.AddQuadTriangles(addToRenderMesh);
00310
00311         /* if the data should be added to the render mesh also add the uvs to the mesh
00312         if (addToRenderMesh)
```

```

00323             meshData.uv.AddRange(FaceUVs(Direction.WEST));
00324
00325         return meshData;
00326     }
00327 #endregion
00328
00329 #region Interfaces
00330 public object Clone()
00331 {
00332     /* Saves this to a file then reads it back so that a copy and not a reference is passed
00333     BinaryFormatter bf = new BinaryFormatter();
00334     MemoryStream ms = new MemoryStream();
00335
00336     bf.Serialize(ms, this);
00337     ms.Seek(0, SeekOrigin.Begin);
00338
00339     return bf.Deserialize(ms);
00340 }
00341 #endregion
00342
00343 #region Overrides
00344 public override string ToString()
00345 {
00346     return $"{itemName} \nID: {GetItemID()}";
00347 }
00348
00349 public override int GetHashCode()
00350 {
00351     return ID;
00352 }
00353
00354 public override bool Equals(object obj)
00355 {
00356     if (!(obj is Item))
00357         return false;
00358
00359     return this == (obj as Item);
00360 }
00361
00362 public static bool operator ==(Item a, Item b)
00363 {
00364     if (ReferenceEquals(a, null) && ReferenceEquals(b, null))
00365         return true;
00366     if (ReferenceEquals(a, null) || ReferenceEquals(b, null))
00367         return false;
00368
00369     if(a.GetItemID() == b.GetItemID())
00370         return true;
00371
00372     return false;
00373 }
00374
00375 public static bool operator !=(Item a, Item b)
00376 {
00377     return !(a == b);
00378 }
00379 #endregion
00380 }
00381
00382 [Serializable]
00383 public struct Tile
00384 {
00385     public int x;
00386     public int y;
00387 }
00388 }
```

7.91 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/QuestBook.cs File Reference

Classes

- class BeeGame.Items.QuestBook

Namespaces

- namespace BeeGame.Items

7.92 QuestBook.cs

```

00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using System.Text;
00005 using System.Threading.Tasks;
00006 using UnityEngine;
00007 using BeeGame.Core.Dictionaries;
00008 using BeeGame.Inventory;
00009
00010 namespace BeeGame.Items
00011 {
00012     [System.Serializable]
00013     public class QuestBook : BeeAlyzer
00014     {
00015         public static new int ID = 15;
00016         public override int maxStackCount
00017         {
00018             get
00019             {
00020                 return 1;
00021             }
00022         }
00023         public override bool placeable
00024         {
00025             get
00026             {
00027                 return false;
00028             }
00029         }
00030
00031         public QuestBook(): base("Quest Book")
00032         {
00033
00034     }
00035
00036         public override void OpenItemInventory(Inventory.Inventory
00037             playerInventory = null)
00038         {
00039             if (myInventory == null)
00040             {
00041                 /* makes the inventory
00042                 myInventory = (GameObject)UnityEngine.Object.Instantiate(
00043                     UnityEngine.Resources.Load("Prefabs/QuestBookInventory"));
00044
00045                 /* opens the inventory and gives it the players inventory
00046                 myInventory.GetComponent<QuestBookInventory>().ToggleInventory(
00047                     playerInventory);
00048                 myInventory.GetComponent<QuestBookInventory>().myItem = this;
00049             }
00050             else
00051             {
00052                 myInventory = null;
00053             }
00054
00055         public override Sprite GetItemSprite()
00056         {
00057             return SpriteDictionary.GetSprite("QuestBook");
00058         }
00059         public override int GetHashCode()
00060         {
00061             return ID;
00062         }
00063     }

```

7.93 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/LoadResources.cs File Reference

Classes

- class BeeGame.LoadResources

Loads all of the resources in the game

Namespaces

- namespace BeeGame

7.94 LoadResources.cs

```

00001 using UnityEngine;
00002 using BeeGame.Core.Dictionaries;
00003
00004 namespace BeeGame
00005 {
00009     public class LoadResources : MonoBehaviour
00010     {
00014         void Awake()
00015         {
00016             Serialization.Serialization.MakeDirectories();
00017             Serialization.Serialization.LoadPlayerPosition(GameObject.Find("Player").GetComponent<Transform>());
00019             Serialization.Serialization.LoadQuests();
00020
00021             SpriteDictionary.LoadSprites();
00022             PrefabDictionary.LoadPrefabs();
00023         }
00024
00025         int delay = 0;
00026
00027         private void Update()
00028         {
00029             if (delay++ >= 1000)
00030                 Serialization.Serialization.SaveQuests();
00031         }
00032     }
00033 }
```

7.95 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/obj/Debug/Temporary GeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs File Reference**7.96 TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs****7.97 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/obj/Debug/Temporary GeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs File Reference****7.98 TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs****7.99 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/obj/Debug/Temporary GeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs File Reference****7.100 TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs**

7.101 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/PlayerLook.cs
File Reference

Classes

- class [BeeGame.Player.PlayerLook](#)
The look for the player

Namespaces

- namespace [BeeGame.Player](#)

7.102 PlayerLook.cs

```
00001 using UnityEngine;
00002 using BeeGame.Core;
00003
00004 namespace BeeGame.Player
00005 {
00009     public class PlayerLook : MonoBehaviour
0010     {
0011         #region Data
0012         public Transform myTransform;
0019         public Transform cameraTransform;
0023         [Range(0, 360)]
0024         public float rotationLock;
0028         public float speed = 5;
0032         float yRot = 0;
0036         float xRot = 0;
0037         #endregion
0038
0039         #region Unity Methods
0040         void Start()
0044         {
0045             Cursor.lockState = CursorLockMode.Locked;
0046             Cursor.visible = false;
0047         }
0048
0052         void Update()
0053         {
0054             /*the look wil not update when a inventory GUI is open
0055             if (!THInput.isAnotherInventoryOpen)
0056             {
0057                 Look();
0058             }
0059         }
0060         #endregion
0061
0062         #region Methods
0063         void Look()
0067         {
0068             //Only X/Y rotation needed as Z rotation would be wierd
0069             yRot += Input.GetAxis("Mouse X") * speed * Time.timeScale;
0070             xRot -= Input.GetAxis("Mouse Y") * speed * Time.timeScale;
0071
0072             //clamps the X rotation so the player camera cannot do flips
0073             xRot = Mathf.Clamp(xRot, -rotationLock, rotationLock);
0074
0075             myTransform.rotation = Quaternion.Euler(0, yRot, 0);
0076             cameraTransform.localRotation = Quaternion.Euler(xRot, 0, 0);
0077         }
0078         #endregion
0079     }
0080 }
```

7.103 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/PlayerMove.cs
File Reference

Classes

- class [BeeGame.Player.PlayerMove](#)
Moves the player

Namespaces

- namespace BeeGame.Player

7.104 PlayerMove.cs

```

00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using System.Text;
00005 using UnityEngine;
00006 using BeeGame.Core;
00007
00008 namespace BeeGame.Player
00009 {
00013     [RequireComponent(typeof(Rigidbody))]
00014     public class PlayerMove : MonoBehaviour
00015     {
00016         #region Data
00017         public float speed = 10f;
00024         public float gravity = 9.81f;
00028         public float maxVelocity = 10f;
00029
00033         private bool canJump = false;
00037         public float jumpHeight = 2f;
00038
00042         private Rigidbody myRigidbody;
00043         #endregion
00044
00045         #region Unity Methods
00046         private void Awake()
00047         {
00051             myRigidbody = GetComponent<Rigidbody>();
00052
00053             //I want to use my own gravity and rotation
00054             myRigidbody.useGravity = false;
00055             myRigidbody.freezeRotation = true;
00056         }
00057
00061         void FixedUpdate()
00062         {
00063             //If the player is grounded it can move
00064             if (canJump)
00065             {
00066                 MovePlayer();
00067             }
00068
00069             //adds the downward force
00070             myRigidbody.AddForce(new Vector3(0, myRigidbody.mass * -gravity, 0));
00071         }
00072
00077         private void OnCollisionStay(Collision collision)
00078         {
00079             canJump = true;
00080         }
00081         #endregion
00082
00083         #region Movement Methods
00084         void MovePlayer()
00085         {
00089             //Calculate the speed we want to achieve
00090             Vector3 targetVelocity = new Vector3(THInput.GetAxis("Horizontal"), 0,
00091             THInput.GetAxis("Vertical"));
00092             targetVelocity = transform.TransformDirection(targetVelocity);
00093             targetVelocity *= speed;
00094
00095             //Apply a force to reach the target speed
00096             Vector3 velocity = myRigidbody.velocity;
00097             Vector3 velocityChange = (targetVelocity - velocity);
00098
00099             //Clamping the velocity so that the player does not infinitely accelerate
00100             velocityChange.x = Mathf.Clamp(velocityChange.x, -maxVelocity, maxVelocity);
00101             velocityChange.z = Mathf.Clamp(velocityChange.z, -maxVelocity, maxVelocity);
00102             velocityChange.y = 0;
00103
00104             //Adds the force to the player so they move in the correct direction
00105             myRigidbody.AddForce(velocityChange, ForceMode.Impulse);
00106
00107             //Jumping
00108             if (canJump && THInput.GetButton("Jump"))
00109             {
00110                 canJump = false;

```

```

00110         myRigidbody.velocity = new Vector3(velocity.x, VerticalJumpSpeed(), velocity.z);
00111     }
00112 }
00113
00118     float VerticalJumpSpeed()
00119     {
00120         /*Gets the correct of fore required for the player to reach the desired apex
00121         /*Can this be done without Square Root as that take alot of work?
00122         return Mathf.Sqrt(2 * jumpHeight * gravity);
00123     }
00124 #endregion
00125 }
00126 }
```

7.105 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/SavePlayerPosition.cs File Reference

Classes

- class [BeeGame.Player.SavePlayerPosition](#)
Saves the player position

Namespaces

- namespace [BeeGame.Player](#)

7.106 SavePlayerPosition.cs

```

00001 using UnityEngine;
00002 using BeeGame.Serialization;
00003
00004 namespace BeeGame.Player
00005 {
00009     public class SavePlayerPosition : MonoBehaviour
00010     {
00014         int counter = 0;
00015
00019         void Update()
00020         {
00021             if(counter == 0)
00022             {
00023                 counter = 1000;
00024                 Serialization.Serialization.SavePlayerPosition(transform);
00025                 //print("saved player");
00026             }
00027             counter--;
00028         }
00029     }
00030 }
```

7.107 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/Selector.cs File Reference

Classes

- class [BeeGame.Player.Selector](#)
Moves the Block selector

Namespaces

- namespace [BeeGame.Player](#)

7.108 Selector.cs

```

00001 using UnityEngine;
00002 using BeeGame.Blocks;
00003 using BeeGame.Terrain.Chunks;
00004 using BeeGame.Inventory.Player_Inventory;
00005 using BeeGame.Items;
00006 using BeeGame.Core;
00007 using static BeeGame.Terrain.LandGeneration.Terrain;
00008 using static BeeGame.Core.THInput;
00009
00010 namespace BeeGame.Player
00011 {
00015     public class Selector : MonoBehaviour
00016     {
00017         #region Data
00018         public GameObject selector;
00022
00026         public PlayerInventory playerInventory;
00027
00031         public LayerMask layers;
00035         private RaycastHit hit;
00036
00040         public int selectedHotbarSlot = 27;
00041         #endregion
00042
00043         #region Unity Methods
00044         void Awake()
00045         {
00046             selector = Instantiate(selector);
00047         }
00048
00055         void FixedUpdate()
00056         {
00057             if (!isAnotherInventoryOpen)
00058                 UpdateSelector();
00059         }
00060
00064         void Update()
00065         {
00066             if (!isAnotherInventoryOpen)
00067             {
00068                 if (GetButtonDown("Break Block"))
00069                     BreakBlock();
00070                 if (GetButtonDown("Place"))
00071                     PlaceBlock();
00072             }
00073         }
00074         #endregion
00075
00076         #region Update
00077         void UpdateSelector()
00078         {
00079             if (Physics.Raycast(transform.position, transform.forward, out hit, 15, layers))
00080             {
00081                 selector.SetActive(true);
00082                 selector.transform.position = GetBlockPos(hit);
00083                 /*selector.SetActive(BlockInPosition(GetBlockPos(hit),
00084
00085                 hit.collider.GetComponent<Chunk>()));
00086
00087                 */
00088                 else
00089                 {
00090                     selector.SetActive(false);
00091                 }
00092                 SelectedSlot();
00093             }
00094
00098             void SelectedSlot()
00099             {
00100                 /* adds 1 to the selected slot and if that is out of range set it to the first hotbar slot
00101                 if (Input.GetAxis("Mouse ScrollWheel") > 0)
00102                 {
00103                     selectedHotbarSlot += 1;
00104                     if (selectedHotbarSlot == 36)
00105                         selectedHotbarSlot = 27;
00106                 }
00107                 /* removes one from the hotbar selector and if the selector would be inside the inventory set
00108                 it to the last slot in the hotbar
00109                 else if (Input.GetAxis("Mouse ScrollWheel") < 0)
00110                 {
00111                     selectedHotbarSlot -= 1;
00112                     if (selectedHotbarSlot == 26)
00113                         selectedHotbarSlot = 35;
00114
00115                 transform.parent.GetComponentInChildren<PlayerInventory>().SelectedSlot(

```

```

0016     selectedHotbarSlot);
0017 }
0018 #endregion
0019 #region Break/Place
0020 void BreakBlock()
0021 {
0022     Chunk chunk = GetChunk(selector.transform.position);
0023
0024     Block block = chunk.world.GetBlock((int)selector.transform.position.x, (int)selector.
0025 transform.position.y, (int)selector.transform.position.z);
0026
0027     if (!block.breakable)
0028         return;
0029
0030     chunk.world.SetBlock((int)selector.transform.position.x, (int)selector.transform.position.
0031 y, (int)selector.transform.position.z, new Air(), true);
0032     /* set to changed so when block is placed down again it will be saved
0033     block.changed = true;
0034     block.BreakBlock(selector.transform.position);
0035
0036 }
0037
0038 void PlaceBlock()
0039 {
0040     Chunk chunk = GetChunk(selector.transform.position);
0041
0042     if (chunk == null)
0043         return;
0044
0045     transform.parent.GetComponentInChildren<PlayerInventory>().GetItemFromHotBar(
0046     selectedHotbarSlot, out var item);
0047
0048     if (item != null)
0049     {
0050         if (item.InteractWithObject())
0051         {
0052             item.InteractWithItem(playerInventory);
0053             return;
0054         }
0055         else if (item.placeable)
0056         {
0057             chunk.world.SetBlock((int)(selector.transform.position.x + hit.normal.x), (int)(
0058             selector.transform.position.y + hit.normal.y), (int)(selector.transform.position.z + hit.normal.z), (
0059             Block)item.CloneObject(), true);
0060         }
0061     }
0062
0063     if (!chunk.GetBlock((int)selector.transform.position.x - chunk.
0064     chunkWorldPos.x, (int)selector.transform.position.y - chunk.
0065     chunkWorldPos.y, (int)selector.transform.position.z - chunk.
0066     chunkWorldPos.z).InteractWithBlock(playerInventory))
0067         return;
0068     }
0069 #endregion
0070 }
0071 }
```

7.109 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Properties/AssemblyInfo.cs File Reference

7.110 AssemblyInfo.cs

```

00001 using System.Resources;
00002 using System.Reflection;
00003 using System.Runtime.CompilerServices;
00004 using System.Runtime.InteropServices;
00005
00006 /* General Information about an assembly is controlled through the following
00007 /* set of attributes. Change these attribute values to modify the information
00008 /* associated with an assembly.
00009 [assembly: AssemblyTitle("BeeGame")]
00010 [assembly: AssemblyDescription("Game made for Computer Science Project")]
00011 [assembly: AssemblyConfiguration("")]
00012 [assembly: AssemblyCompany("")]
00013 [assembly: AssemblyProduct("BeeGame")]
00014 [assembly: AssemblyCopyright("Copyright © 2017")]
00015 [assembly: AssemblyTrademark("")]
00016 [assembly: AssemblyCulture("")]
00017
00018 /* Setting ComVisible to false makes the types in this assembly not visible
```

```

00019 /* to COM components. If you need to access a type in this assembly from
00020 /* COM, set the ComVisible attribute to true on that type.
00021 [assembly: ComVisible(false)]
00022
00023 /* The following GUID is for the ID of the typelib if this project is exposed to COM
00024 [assembly: Guid("9b332f5d-31cc-41f5-9517-5ed40d0e4855")]
00025
00026 /* Version information for an assembly consists of the following four values:
00027 /**
00028 /*     Major Version
00029 /*     Minor Version
00030 /*     Build Number
00031 /*     Revision
00032 /**
00033 /* You can specify all the values or you can default the Build and Revision Numbers
00034 /* by using the '*' as shown below:
00035 /* [assembly: AssemblyVersion("1.0.*")]
00036 [assembly: AssemblyVersion("1.0.0.0")]
00037 [assembly: AssemblyFileVersion("0.0.0.1")]
00038 [assembly: NeutralResourcesLanguage("en")]
00039

```

7.111 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Quest/QuestEvents.cs File Reference

Classes

- class BeeGame.Quest.QuestEvents

Namespaces

- namespace BeeGame.Quest

7.112 QuestEvents.cs

```

00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using System.Text;
00005 using System.Threading.Tasks;
00006 using BeeGame.Core.Enums;
00007
00008 namespace BeeGame.Quest
00009 {
00010     public class QuestEvents
00011     {
00012         public delegate void QuestEventHandler (int quest);
00013         private static event QuestEventHandler itemCraftedEvent = Quests.
    ReturnCraftingTableQuest;
00014         private static event QuestEventHandler itemPickupEvent = Quests.
    ReturnPickupQuest;
00015
00016         private delegate void BeeQuestEventHandler(BeeSpecies species);
00017         private static event BeeQuestEventHandler beeCraftedEvent = Quests.
    ReturnBeeCraftingQuest;
00018
00019         private delegate void PureBeeQuestEventHandler(BeeSpecies species1,
    BeeSpecies species2);
00020         private static event PureBeeQuestEventHandler pureBeeCraftedEvent =
    Quests.ReturnPureBreadBeeCraftingQuest;
00021
00022         public static void CallItemPickupEvent(int id)
00023         {
00024             itemPickupEvent(id);
00025         }
00026
00027         public static void CallItemCraftedEvent(int id)
00028         {
00029             itemCraftedEvent(id);
00030         }
00031
00032         public static void CallBeeCraftedEvent(BeeSpecies species)

```

```
00033     {
00034         beeCraftedEvent(species);
00035     }
00036
00037     public static void CallPureBeeCraftedEvent(
00038         BeeSpecies species1, BeeSpecies species2)
00039     {
00040         pureBeeCraftedEvent(species1, species2);
00041     }
00042 }
```

7.113 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Quest/Quests.cs File Reference

Classes

- class BeeGame.Quest.Quests

Namespaces

- namespace BeeGame.Quest

7.114 Quests.cs

```
00001 using System.Collections.Generic;
00002 using BeeGame.Items;
00003 using BeeGame.Blocks;
00004 using BeeGame.Core.Enums;
00005 using BeeGame.Exceptions;
00006
00007 namespace BeeGame.Quest
00008 {
00009     [System.Serializable]
00010     public static class Quests
00011     {
00015         private static Dictionary<string, object[]> compleatedQuests = new Dictionary<string, object[]>();
00016
00020         private static Dictionary<string, object[]> compleatedUnclaimedQuests = new Dictionary<string, object[]>();
00021
00025         private static Dictionary<string, object[]> currentQuests = new Dictionary<string, object[]>();
00026         {
00027             { $"Pickup: {Wood.ID}", new object[] {new CraftingTable(), $"Crafted: {Grass.ID}"},
00028             "Pickup A Wood Block" },
00029             { $"BeeCrafted: {BeeSpecies.COMMON}", new object[] {new CraftingTable(), "Make a
Common Bee" } }
00030         };
00034
00035         private static Dictionary<string, object[]> lockedQuests = new Dictionary<string, object[]>();
00036         {
00037             { $"Crafted: {Grass.ID}", new object[] {new Dirt(), "nothing", "Use some Dirt in the
Workbench" } }
00038         };
00039
00040         public static Dictionary<string, object[]> ReturnCompleatedQuests()
00041         {
00042             return compleatedUnclaimedQuests;
00043         }
00044
00045         public static Dictionary<string, object[]> ReturnCompleatedClaimedQuests()
00046         {
00047             return compleatedQuests;
00048         }
00049
00050         public static Dictionary<string, object[]> ReturnCurrentQuests()
00051         {
00052             return currentQuests;
00053         }
00054
00055         public static Dictionary<string, object[]> ReturnLockedQuests()
```

```

00055      {
00056          return lockedQuests;
00057      }
00058
00059      public static void LoadQuests(Dictionary<string, object[]> compleated, Dictionary<string,
00060      object[]> compleatedNotCollected, Dictionary<string, object[]> inProgress, Dictionary<string, object[]>
00061      locked)
00062      {
00063          compleatedQuests = compleated;
00064          compleatedUnclaimedQuests = compleatedNotCollected;
00065          currentQuests = inProgress;
00066          lockedQuests = locked;
00067
00068      public static void ClaimQuest(string key)
00069      {
00070          var item = compleatedUnclaimedQuests[key];
00071          compleatedQuests.Add(key, compleatedUnclaimedQuests[key]);
00072          compleatedUnclaimedQuests.Remove(key);
00073
00074          var temp = UnityEngine.Object.Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as UnityEngine.GameObject, UnityEngine.Object.FindObjectOfType<Player.PlayerMove>().transform.position + UnityEngine.Vector3.one,
00075          UnityEngine.Quaternion.identity);
00076          temp.GetComponent<ItemGameObject>().item = (Item)item[0];
00077
00078      public static void AddQuest(string quest, Item result, string nextQuest)
00079      {
00080          if(!lockedQuests.ContainsKey(quest))
00081          {
00082              lockedQuests.Add(quest, new object[] { result, nextQuest });
00083              return;
00084          }
00085          else
00086          {
00087              lockedQuests[quest] = new object[] { result, nextQuest };
00088
00089              throw new QuestAlreadyExistsException($"Warning Quest: {quest}, is already a quest and the old has been overridden!");
00090          }
00091
00092      public static void ReturnPickupQuest(int pickupID)
00093      {
00094          if (currentQuests.ContainsKey($"Pickup: {pickupID}"))
00095          {
00096              UnlockQuests($"Pickup: {pickupID}");
00097          }
00098
00099
00100     public static void ReturnCraftingTableQuest(int craftedItemID)
00101     {
00102         if (currentQuests.ContainsKey($"Crafted: {craftedItemID}"))
00103         {
00104             UnlockQuests($"Crafted: {craftedItemID}");
00105         }
00106
00107
00108     public static void ReturnBeeCraftingQuest(
00109     BeeSpecies primaryBeeSpecies)
00110     {
00111         if (currentQuests.ContainsKey($"BeeCrafted: {primaryBeeSpecies}"))
00112         {
00113             UnlockQuests($"BeeCrafted: {primaryBeeSpecies}");
00114         }
00115
00116     public static void ReturnPureBreadBeeCraftingQuest(
00117     BeeSpecies primaryBeeSpecies, BeeSpecies secondaryBeeSpecies)
00118     {
00119         if (currentQuests.ContainsKey($"PureBredBee: {primaryBeeSpecies} {secondaryBeeSpecies}"))
00120         {
00121             UnlockQuests($"PureBredBee: {primaryBeeSpecies} {secondaryBeeSpecies}");
00122         }
00123
00124     private static void UnlockQuests(string key)
00125     {
00126         compleatedUnclaimedQuests.Add(key, currentQuests[key]);
00127
00128         var objArray = currentQuests[key];
00129
00130         if(objArray.Length > 2)
00131         {
00132             var next = objArray[1];
00133

```

```

00134         if (next is string[] sa)
00135         {
00136             foreach (var q in sa)
00137             {
00138                 currentQuests.Add(q, lockedQuests[q]);
00139                 lockedQuests.Remove(q);
00140             }
00141         }
00142     else if (next is string ss)
00143     {
00144         if (lockedQuests.ContainsKey(ss))
00145         {
00146             currentQuests.Add(ss, lockedQuests[ss]);
00147             lockedQuests.Remove(ss);
00148         }
00149     }
00150 }
00151
00152     currentQuests.Remove(key);
00153
00154     Serialization.Serialization.SaveQuests();
00155 }
00156 }
00157 }
```

7.115 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Resources/Resources.Designer.cs File Reference

Classes

- class BeeGame.Resources.Resources
A strongly-typed resource class, for looking up localized strings, etc.

Namespaces

- namespace BeeGame.Resources

7.116 Resources.Designer.cs

```

00001 //-----
00002 /* <auto-generated>
00003 /*     This code was generated by a tool.
00004 /*     Runtime Version:4.0.30319.42000
00005 /*
00006 /*     Changes to this file may cause incorrect behavior and will be lost if
00007 /*     the code is regenerated.
00008 /* </auto-generated>
00009 //-----
0010
0011 namespace BeeGame.Resources {
0012     using System;
0013     using System.Collections.Generic;
0014     using UnityEngine;
0015
0016     /* This class was auto-generated by the StronglyTypedResourceBuilder
0017     /* class via a tool like ResGen or Visual Studio.
0018     /* To add or remove a member, edit your .ResX file then rerun ResGen
0019     /* with the /str option, or rebuild your VS project.
0020     [global::System.CodeDom.Compiler.GeneratedCodeAttribute("
```

`System.Resources.Tools.StronglyTypedResourceBuilder", "4.0.0.0")]
0021 [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
0022 [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
0023 internal class Resources {
0024
0025 private static global::System.Resources.ResourceManager
0026 resourceMan;
0027
0028 private static global::System.Globalization.CultureInfo resourceCulture;
0029
0030 [global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance", "CA1811:AvoidUncalledPrivateCode")]
0031 internal Resources() {`

```

00034         }
00035
00039     [global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.
00040         EditorBrowsableState.Advanced)]
00040     internal static global::System.Resources.ResourceManager ResourceManager {
00041         get {
00042             if (object.ReferenceEquals(resourceMan, null)) {
00043                 global::System.Resources.ResourceManager temp = new global::System.Resources.
00044                     ResourceManager("BeeGame.Resources.Resources", typeof(Resources).Assembly);
00045                 resourceMan = temp;
00046             }
00047         }
00048     }
00049
00054     [global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.
00055         EditorBrowsableState.Advanced)]
00055     internal static global::System.Globalization.CultureInfo Culture {
00056         get {
00057             return resourceCulture;
00058         }
00059         set {
00060             resourceCulture = value;
00061         }
00062     }
00063
00067     internal static byte[] Prefabs {
00068         get {
00069             object obj = ResourceManager.GetObject("Prefabs", resourceCulture);
00070             return ((byte[])(obj));
00071         }
00072     }
00073
00077     internal static byte[] Sprites {
00078         get {
00079             object obj = ResourceManager.GetObject("Sprites", resourceCulture);
00080             return ((byte[])(obj));
00081         }
00082     }
00083
00088     internal static Dictionary<string, Sprite> GetSprites()
00089     {
00090         string[] splitCharacters = new string[] { "," };
00091         object obj = ResourceManager.GetObject("Sprites", resourceCulture);
00092
00093         /* gets the text from the spries.dat file
00094         string text = System.Text.Encoding.Default.GetString((byte[])obj);
00095         string lineText = "";
00096         string[] splitText;
00097         Texture2D tex;
00098         Dictionary<string, Sprite> sprites = new Dictionary<string, Sprite>();
00099
00100        /* goes through all characters in the file
00101        for (int i = 0; i < text.Length; i++)
00102        {
00103            /* when their is a new line the path for that sprite is found
00104            if (text[i] != '\n')
00105            {
00106                lineText += text[i];
00107            }
00108            else
00109            {
00110                splitText = lineText.Split(splitCharacters, StringSplitOptions.RemoveEmptyEntries);
00111                lineText = "";
00112                tex = UnityEngine.Resources.Load("Sprites/" + splitText[1].Remove(splitText[1].
00113                Length - 1, 1)) as Texture2D;
00114                /* need to crafte a sprite from a texture 2D because
00115                Unity wont allow images to be loaded directly as sprites at runtime...for some reason
00116                sprites.Add(splitText[0], Sprite.Create(tex, new UnityEngine.Rect(0, 0, tex.
00117                width, tex.height), Vector2.zero));
00118            }
00119            splitText = lineText.Split(splitCharacters, StringSplitOptions.RemoveEmptyEntries);
00120            lineText = "";
00121            tex = UnityEngine.Resources.Load("Sprites/" + splitText[1]) as Texture2D;
00122            sprites.Add(splitText[0], Sprite.Create(tex, new UnityEngine.Rect(0, 0, tex.width,
00123                tex.height), Vector2.zero));
00124        }
00125
00130        return sprites;
00131    }
00132
00133    internal static Dictionary<string, GameObject> GetPrefabs()
00134    {
00135        string[] splitCharacters = new string[] { "," };
00136        object obj = ResourceManager.GetObject("Prefabs", resourceCulture);
00137

```

```

00135         string text = System.Text.Encoding.Default.GetString((byte[])obj);
00136         text = text.Remove(0, 3);
00137         string lineText = "";
00138         string[] splitText;
00139         Dictionary<string, GameObject> objects = new Dictionary<string, GameObject>();
00140
00141         /* goes through all characters in the file
00142         for (int i = 0; i < text.Length; i++)
00143         {
00144             /* when their is a new line the path for that sprite is found
00145             if (text[i] != '\n')
00146             {
00147                 lineText += text[i];
00148             }
00149             else
00150             {
00151                 /* when their is a new line the path for that sprite is found
00152                 splitText = lineText.Split(splitCharacters, StringSplitOptions.RemoveEmptyEntries);
00153                 lineText = "";
00154                 objects.Add(splitText[0], UnityEngine.Resources.Load("Prefabs/" + splitText[1].Remove(splitText[1].Length - 1, 1)) as GameObject);
00155             }
00156         }
00157
00158         splitText = lineText.Split(splitCharacters, StringSplitOptions.RemoveEmptyEntries);
00159         lineText = "";
00160         objects.Add(splitText[0], UnityEngine.Resources.Load("Prefabs/" + splitText[1]) as GameObject);
00161         return objects;
00162     }
00163 }
00164 }
00165 }
```

7.117 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Serialization/Serialization.cs

File Reference

Classes

- class [BeeGame.Serialization.Serialization](#)
Serializes and Deserialises things

Namespaces

- namespace [BeeGame.Serialization](#)

7.118 Serialization.cs

```

00001 using System.IO;
00002 using System.Runtime.Serialization;
00003 using System.Runtime.Serialization.Formatters.Binary;
00004 using UnityEngine;
00005 using BeeGame.Core;
00006 using BeeGame.Terrain;
00007 using BeeGame.Terrain.Chunks;
00008 using BeeGame.Inventory;
00009 using BeeGame.Quest;
00010
00011 namespace BeeGame.Serialization
00012 {
00019     public static class Serialization
00020     {
00021         #region Data
00022         public static string worldName = "World";
00029         public static string saveFolderName = "Saves";
00033         private static string savePath;
00034         #endregion
00035
00039         public static void MakeDirectories()
00040         {
00041             savePath = $"{Application.dataPath}/{saveFolderName}/{worldName}";
00042 }
```

```

00043         if (!Directory.Exists(savePath))
00044             Directory.CreateDirectory(savePath);
00045     }
00046
00051     public static void DeleteFile(string fileName)
00052     {
00053         string[] file = Directory.GetFiles(Application.dataPath + "/Saves", "*.dat", SearchOption.
00054             AllDirectories);
00055         string[] splitCharacters = { "/", "\\\", ".dat" };
00056
00057         for (int i = 0; i < file.Length; i++)
00058         {
00059             string[] temp = file[i].Split(splitCharacters, System.StringSplitOptions.
00060             RemoveEmptyEntries);
00061             if (temp[temp.Length - 1] == fileName)
00062             {
00063                 File.Delete(file[i]);
00064
00065                 return;
00066             }
00067         }
00068     }
00069
00070     #region Player
00071     public static void SavePlayerPosition(Transform positon)
00072     {
00073         THVector3[] playerTransform = new THVector3[3];
00074
00075         playerTransform[0] = positon.position;
00076         playerTransform[1] = positon.rotation.eulerAngles;
00077         playerTransform[2] = positon.localScale;
00078
00079         string playerPosSavePath = $"{savePath}/player.dat";
00080
00081         SaveFile(playerTransform, playerPosSavePath);
00082     }
00083
00084     public static void LoadPlayerPosition(Transform playerTransfom)
00085     {
00086         string playerPosSavePath = $"{savePath}/player.dat";
00087
00088         if (!File.Exists(playerPosSavePath))
00089             return;
00090
00091         THVector3[] pos = (THVector3[])LoadFile(playerPosSavePath);
00092
00093         playerTransfom.position = pos[0];
00094         playerTransfom.rotation = (Quaternion)pos[1];
00095         playerTransfom.localScale = pos[2];
00096     }
00097
00098     #endregion
00099
00100     #region Inventory
00101     public static void SerializeInventory(Inventory.Inventory inventory, string inventoryName)
00102     {
00103         string inventorySavePath = $"{savePath}/Inventorys";
00104
00105         if (!Directory.Exists(inventorySavePath))
00106             Directory.CreateDirectory(inventorySavePath);
00107
00108         SaveFile(inventory.GetAllItems(), $"{inventorySavePath}/{inventoryName}.dat");
00109     }
00110
00111     public static void DeSerializeInventory(Inventory.Inventory inventory,
00112     string inventoryName)
00113     {
00114         /* make the path
00115         string inventorySavePath = $"{savePath}/Inventorys/{inventoryName}.dat";
00116
00117         /* checks that the file exists
00118         if (!File.Exists(inventorySavePath))
00119         {
00120             for (int i = 0; i < inventory.items.itemsInInventory.Length; i++)
00121             {
00122                 inventory.items.itemsInInventory[i] = null;
00123             }
00124
00125             SerializeInventory(inventory, inventoryName);
00126
00127             return;
00128         }
00129
00130         inventory.SetAllItems((ItemsInInventory)LoadFile($"{inventorySavePath}"));
00131     }
00132
00133     #endregion

```

```
00153
00154     #region Chunk
00155     public static void SaveChunk(Chunk chunk)
00160     {
00161         /* saves the blocks
00162         SaveChunk save = new SaveChunk(chunk.blocks);
00163
00164         /* if no block was changed return early
00165         if (save.blocks.Count == 0)
00166             return;
00167
00168         /* otherwise save the file
00169         string saveFile = $"{savePath}/{FileName(chunk.chunkWorldPos)}.dat";
00170
00171         SaveFile(save, saveFile);
00172     }
00173
00179     public static bool LoadChunk(Chunk chunk)
00180     {
00181         /* gets the save file
00182         string saveFile = $"{savePath}/{FileName(chunk.chunkWorldPos)}.dat";
00183
00184         /* if the file does not exist return false
00185         if (!File.Exists(saveFile))
00186             return false;
00187
00188         /* set all of the changed blocks in the chunk
00189         SaveChunk save = (SaveChunk)Loadfile(saveFile);
00190
00191         foreach (var block in save.blocks)
00192         {
00193             chunk.blocks[block.Key.x, block.Key.y, block.Key.z] = block.Value;
00194         }
00195
00196         return true;
00197     }
00198
00204     public static string FileName(ChunkWorldPos pos)
00205     {
00206         return $"{pos.x}, {pos.y}, {pos.z}";
00207     }
00208     #endregion
00209
00210     #region Quests
00211     public static void SaveQuests()
00212     {
00213         var array = new System.Collections.Generic.Dictionary<string, object[]>[]
00214             {
00215                 Quests.ReturnCompledatedClaimedQuests(),
00216                 Quests.ReturnCompledatedQuests(), Quests.
00217                 ReturnCurrentQuests(), Quests.ReturnLockedQuests() };
00218
00219         string saveFile = $"{savePath}/quests.dat";
00220
00221         SaveFile(array, saveFile);
00222     }
00223
00224     public static void LoadQuests()
00225     {
00226         string saveFile = $"{savePath}/quests.dat";
00227
00228         if (!File.Exists(saveFile))
00229             return;
00230
00231         var array = (System.Collections.Generic.Dictionary<string, object[]>[])
00232             LoadFile(saveFile);
00233
00234         Quests.LoadQuests(array[0], array[1], array[2], array[3]);
00235     }
00236     #endregion
00237
00238     #region Save/Load Files
00239     private static void SaveFile(object obj, string file)
00240     {
00241         BinaryFormatter bf = new BinaryFormatter();
00242         FileStream fs = new FileStream(file, FileMode.OpenOrCreate);
00243
00244         try
00245         {
00246             bf.Serialize(fs, obj);
00247         }
00248         catch(SerializationException e)
00249         {
00250             Debug.Log($"Serialization Exception: {e}");
00251             throw new SerializationException();
00252         }
00253         finally
```

```
00255     {
00256         fs.Close();
00257     }
00258 }
00259
00260 private static object LoadFile(string file)
00261 {
00262     BinaryFormatter bf = new BinaryFormatter();
00263     FileStream fs = new FileStream(file, FileMode.Open);
00264
00265     try
00266     {
00267         return bf.Deserialize(fs);
00268     }
00269     catch(SerializationException e)
00270     {
00271         Debug.Log($"Deserialization Exception {e}");
00272         throw new SerializationException();
00273     }
00274     finally
00275     {
00276         fs.Close();
00277     }
00278 }
00279 #endregion
00280
00281 }
```

7.119 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/SpawnItem.cs File Reference

Classes

- class BeeGame.SpawnItem

Namespaces

- namespace BeeGame

7.120 SpawnItem.cs

```
00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using System.Text;
00005 using UnityEngine;
00006 using BeeGame.Items;
00007 using BeeGame.Blocks;
00008 using BeeGame.Core.Enums;
00009
00010 namespace BeeGame
00011 {
00012     class SpawnItem : MonoBehaviour
00013     {
00014         void Start()
00015         {
00016             //GameObject go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as
00017             GameObject, transform.position, Quaternion.identity) as GameObject;
00017             //go.GetComponent<ItemGameObject>().item = new Bee(BeeType.DRONE, new NormalBee() { pSpecies =
00018             BeeSpecies.FOREST, sSpecies = BeeSpecies.FOREST });
00019
00019             //go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
00020             transform.position, Quaternion.identity) as GameObject;
00020             //go.GetComponent<ItemGameObject>().item = new Bee(BeeType.PRINCESS, new NormalBee() { pSpecies
00020 = BeeSpecies.FOREST, sSpecies = BeeSpecies.FOREST });
00021
00022             //go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
00022             transform.position, Quaternion.identity) as GameObject;
00023             //go.GetComponent<ItemGameObject>().item = new Bee(BeeType.QUEEN, new QueenBee() { queen =
00023             new NormalBee() { pSpecies = BeeSpecies.FOREST, sSpecies = BeeSpecies.FOREST }, drone =
00023             new NormalBee() { pSpecies = BeeSpecies.FOREST, sSpecies = BeeSpecies.FOREST } });
00024 }
```

```

00025         //go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
00026         transform.position, Quaternion.identity) as GameObject;
00027         //go.GetComponent<ItemGameObject>().item = new Bee(BeeType.DRONE, new NormalBee() { pSpecies =
00028             BeeSpecies.COMMON, sSpecies = BeeSpecies.COMMON });
00029
00030         //go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
00031         transform.position, Quaternion.identity) as GameObject;
00032         //go.GetComponent<ItemGameObject>().item = new Bee(BeeType.PRINCESS, new NormalBee() { pSpecies =
00033             BeeSpecies.COMMON, sSpecies = BeeSpecies.COMMON });
00034
00035         //go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
00036         transform.position, Quaternion.identity) as GameObject;
00037         //go.GetComponent<ItemGameObject>().item = new Bee(BeeType.QUEEN, new QueenBee() { queen = new
00038             NormalBee() { pSpecies = BeeSpecies.COMMON, sSpecies = BeeSpecies.COMMON }, drone = new NormalBee() {
00039                 pSpecies = BeeSpecies.COMMON, sSpecies = BeeSpecies.COMMON } });
00040
00041         //go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
00042         transform.position, Quaternion.identity) as GameObject;
00043         //go.GetComponent<ItemGameObject>().item = new HoneyComb(HoneyCombType.ICEY);
00044
00045         //go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
00046         transform.position, Quaternion.identity) as GameObject;
00047         //go.GetComponent<ItemGameObject>().item = new HoneyComb(HoneyCombType.HONEY);
00048
00049         //go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
00050         transform.position, Quaternion.identity) as GameObject;
00051         //go.GetComponent<ItemGameObject>().item = new Chest();
00052
00053         //go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
00054         transform.position, Quaternion.identity) as GameObject;
00055         //go.GetComponent<ItemGameObject>().item = new Apiary();
00056
00057         //go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
00058         transform.position, Quaternion.identity) as GameObject;
00059         //go.GetComponent<ItemGameObject>().item = new CraftingTable();
00060
00061     }
00062 }

```

7.121 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/Chunk.cs File Reference

Classes

- class [BeeGame.Terrain.Chunks.Chunk](#)

A section of land for the game, used so that land can be generated in parts and not all at once

Namespaces

- namespace [BeeGame.Terrain.Chunks](#)

7.122 Chunk.cs

```

00001 using UnityEngine;
00002 using BeeGame.Blocks;
00003 using BeeGame.Terrain.LandGeneration;
00004 using System.Threading;
00005

```

```

00006 namespace BeeGame.Terrain.Chunks
00007 {
00011     [RequireComponent(typeof(MeshFilter))]
00012     [RequireComponent(typeof(MeshRenderer))]
00013     [RequireComponent(typeof(MeshCollider))]
00014     public class Chunk : MonoBehaviour
00015     {
00016         #region Data
00017         public static int chunkSize = 16;
00025
00029         public Block[,,] blocks = new Block[chunkSize, chunkSize, chunkSize];
00030
00034         public bool update = true;
00038         public bool rendered;
00039
00043         public bool updateCollisionMesh = false;
00047         public bool applyCollisionMesh = false;
00048
00052         public World world;
00056         public ChunkWorldPos chunkWorldPos;
00057
00061         private MeshData mesh = new MeshData();
00062
00066         private MeshFilter filter;
00070         private MeshCollider meshCollider;
00071         #endregion
00072
00073         #region Unity Methods
00074         void Start()
00075         {
00079             filter = GetComponent<MeshFilter>();
00080             meshCollider = GetComponent<MeshCollider>();
00081         }
00082
00086         void Update()
00087         {
00088             lock(mesh)
00089             {
00090                 if (update)
00091                 {
00092                     update = false;
00093                     updateCollisionMesh = true;
00094                     mesh = new MeshData();
00095                     /* Enabling threading here works in editor but not in build?
00096                     /* ok whatever...
00097                     /* Thread thread = new Thread(UpdateChunk);
00098
00099                     /* thread.Start();
00100                     UpdateChunk();
00101                 }
00102
00103                 if (mesh.done && mesh != new MeshData())
00104                 {
00105                     RenderMesh(mesh);
00106                 }
00107
00108                 if (applyCollisionMesh)
00109                     ColliderMesh();
00110             }
00111         }
00112         #endregion
00113
00114         #region Get/Set Blocks
00115         public Block GetBlock(int x, int y, int z, bool checkNebouringChunks = true)
00116         {
00125             /* checks that block is in the chunk
00126             if (InRange(x) && InRange(y) && InRange(z))
00127                 return blocks[x, y, z];
00128
00129             /* if the block is not in the chunk and we should check other chunks do that, otherwise return
00130             /* an air block (empty block)
00131             //if(checkNebouringChunks)
00132             //    return world.GetBlock(chunkWorldPos.x + x, chunkWorldPos.
00133             //        y + y, chunkWorldPos.z + z);
00134
00135             //return new Air();
00136         }
00137
00143         public void SetBlock(int x, int y, int z, Block block, bool checkNebouringChunks =
00144             true)
00145         {
00146             /* sets the block in the position if it is in the chunk, then return early
00147             if (InRange(x) && InRange(y) && InRange(z))
00148             {
00149                 blocks[x, y, z] = block;
00150                 return;
00151             }

```

```

00151             if (checkNebouringChunks)
00152                 /* if the block is not in the chunk find its chunk and set it their
00153                 world.SetBlock(chunkWorldPos.x + x, chunkWorldPos.y + y, chunkWorldPos.
00154                 z + z, block);
00155             }
00156
00162         public static bool InRange(int i)
00163         {
00164             /* if the value is less then 0 or greater than 16 the value is outside the chunk
00165             if (i < 0 || i >= chunkSize)
00166                 return false;
00167             return true;
00168         }
00169     #endregion
00170
00171     #region Mesh
00172     public void SetBlocksUnmodified()
00173     {
00180         foreach (var block in blocks)
00181         {
00182             block.changed = false;
00183         }
00184     }
00185
00189     void UpdateChunk()
00190     {
00191         /* says that this chunk is rendered and initialtes the mesh
00192         rendered = true;
00193
00194         /* goes through every block in the blocks array getting their mesh data
00195         for (int x = 0; x < chunkSize; x++)
00196         {
00197             for (int z = 0; z < chunkSize; z++)
00198             {
00199                 for (int y = 0; y < chunkSize; y++)
00200                 {
00201                     blocks[x, y, z]?.UpdateBlock(x, y, z, this);
00202                     mesh = blocks[x, y, z]?.BlockData(this, x, y, z, mesh) ?? mesh;
00203                 }
00204             }
00205         }
00206         mesh.done = true;
00207     }
00208
00213     void RenderMesh(MeshData meshData)
00214     {
00215         /* Applying the mesh takes the longest but nothing can be dont with the mesh class in a
00216         secondary thread...thanks unity
00217
00218         mesh.done = false;
00219         /* clears the current chunk mesh
00220         filter.mesh.Clear();
00221         /* name for convenience
00222         filter.mesh.name = "Render Mesh";
00223         /* puts the tris and verts from the meshdata into the chunk mesh
00224         filter.mesh.vertices = meshData.verts.ToArray();
00225         filter.mesh.triangles = meshData.tris.ToArray();
00226
00227         /* sets the uvs
00228         filter.mesh.uv = meshData.uv.ToArray();
00229
00230         /* redoes the normals incase they got messed up
00231         filter.mesh.RecalculateNormals();
00232         /* is this necissary as it causes alsot of lag?
00233     }
00237     void ColliderMesh()
00238     {
00239         /* if the chunk has been told to update the collsions but the chunk has ne verts dont do it as
00240         /* their is no point
00241             if (this.mesh.verts.Count == 0)
00242                 return;
00243
00244         /* if the render and collision meshes should be shared set the render mesh to the collision
00245         /* mesh otherwise make a collision mesh
00246             if (this.mesh.shareMeshes)
00247             {
00248                 world.chunkHasMadeCollisionMesh = true;
00249                 applyCollisionMesh = false;
00250                 meshCollider.sharedMesh = filter.mesh;
00251                 return;
00252             }
00253
00254             world.chunkHasMadeCollisionMesh = true;
00255             /* Applying the mesh takes the longest but nothing can be done with the mesh class in a
00256             /* secondary thread...thanks Unity

```

```

00254     /* makes a new mesh setting the name for convenience
00255     Mesh mesh = new Mesh()
00256     {
00257         name = "Collider Mesh",
00258         vertices = this.mesh.colVerts.ToArray(),
00259         triangles = this.mesh.colTris.ToArray()
00260     };
00261
00262     /* recalcs the normals and applies the mesh
00263     mesh.RecalculateNormals();
00264
00265     meshCollider.sharedMesh = mesh;
00266
00267     applyCollisionMesh = false;
00268 }
00269 #endregion
00270 }
00271 }
00272 }
```

7.123 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/LoadChunks.cs File Reference

Classes

- class BeeGame.Terrain.Chunks.LoadChunks
Loads the Chunks around the player

Namespaces

- namespace BeeGame.Terrain.Chunks

7.124 LoadChunks.cs

```

00001 using System;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004 using BeeGame.Terrain.LandGeneration;
00005
00006 namespace BeeGame.Terrain.Chunks
00007 {
00011     public class LoadChunks : MonoBehaviour
00012     {
00013         #region Data
00014         public World world;
00018
00022         private List<ChunkWorldPos> buildList = new List<ChunkWorldPos>();
00023
00027         private static ChunkWorldPos[] chunkPositions = new
00028             ChunkWorldPos[] { new ChunkWorldPos( 0, 0, 0 ), new
00029                 ChunkWorldPos( -1, 0, 0 ), new ChunkWorldPos( 0, 0, -1 ), new
00030                 ChunkWorldPos( 0, 0, 1 ), new ChunkWorldPos( 1, 0, 0 ),
00031                     new ChunkWorldPos( -1, 0, -1 ), new
00032                     ChunkWorldPos( -1, 0, 1 ), new ChunkWorldPos( 1, 0, -1 ), new
00033                     ChunkWorldPos( 1, 0, 1 ), new ChunkWorldPos( -2, 0, 0 ),
00034                         new ChunkWorldPos( 0, 0, -2 ), new
00035                         ChunkWorldPos( 0, 0, 2 ), new ChunkWorldPos( 2, 0, 0 ), new
00036                         ChunkWorldPos( -2, 0, -1 ), new ChunkWorldPos( -2, 0, 1 ),
00037                             new ChunkWorldPos( -1, 0, -2 ), new
00038                             ChunkWorldPos( -1, 0, 2 ), new ChunkWorldPos( 1, 0, -2 ), new
00039                             ChunkWorldPos( 1, 0, 2 ), new ChunkWorldPos( 2, 0, -1 ),
00040                                 new ChunkWorldPos( 2, 0, 1 ), new
00041                                 ChunkWorldPos( -2, 0, -2 ), new ChunkWorldPos( -2, 0, 2 ),
00042                                     new ChunkWorldPos( 2, 0, 2 ), new
00043                                     ChunkWorldPos( -3, 0, 0 ), new
00044                                     ChunkWorldPos( 0, 0, -3 ), new ChunkWorldPos( 0, 0, 3 ), new
00045                                     ChunkWorldPos( 3, 0, 0 ), new ChunkWorldPos( -3, 0, -1 ),
00046                                         new ChunkWorldPos( -3, 0, 1 ), new
00047                                         ChunkWorldPos( -1, 0, -3 ), new ChunkWorldPos( -1, 0, 3 ), new
00048                                         ChunkWorldPos( 1, 0, -3 ), new ChunkWorldPos( 1, 0, 3 ),
00049                                             new ChunkWorldPos( 3, 0, -1 ), new
00050                                             ChunkWorldPos( 3, 0, 1 ), new ChunkWorldPos( -3, 0, -2 ), new
00051                                             ChunkWorldPos( 3, 0, 2 ) }
```

```

    ChunkWorldPos(-3, 0, 2), new ChunkWorldPos(-2, 0, -3),
00035                               new ChunkWorldPos(-2, 0, 3), new
    ChunkWorldPos( 2, 0, -3), new ChunkWorldPos( 2, 0, 3), new
    ChunkWorldPos( 3, 0, -2), new ChunkWorldPos( 3, 0, 2),
00036                               new ChunkWorldPos(-4, 0, 0), new
    ChunkWorldPos( 0, 0, -4), new ChunkWorldPos( 0, 0, 4), new
    ChunkWorldPos( 4, 0, 0), new ChunkWorldPos(-4, 0, -1),
00037                               new ChunkWorldPos(-4, 0, 1), new
    ChunkWorldPos(-1, 0, -4), new ChunkWorldPos(-1, 0, 4), new
    ChunkWorldPos( 1, 0, -4), new ChunkWorldPos( 1, 0, 4),
00038                               new ChunkWorldPos( 4, 0, -1), new
    ChunkWorldPos( 4, 0, 1), new ChunkWorldPos(-3, 0, -3), new
    ChunkWorldPos(-3, 0, 3), new ChunkWorldPos( 3, 0, -3),
00039                               new ChunkWorldPos( 3, 0, 3), new
    ChunkWorldPos(-4, 0, -2), new ChunkWorldPos(-4, 0, 2), new
    ChunkWorldPos(-2, 0, -4), new ChunkWorldPos(-2, 0, 4),
00040                               new ChunkWorldPos( 2, 0, -4), new
    ChunkWorldPos( 2, 0, 4), new ChunkWorldPos( 4, 0, -2), new
    ChunkWorldPos( 4, 0, 2), new ChunkWorldPos(-5, 0, 0),
00041                               new ChunkWorldPos(-4, 0, -3), new
    ChunkWorldPos(-4, 0, 3), new ChunkWorldPos(-3, 0, -4), new
    ChunkWorldPos(-3, 0, 4), new ChunkWorldPos( 0, 0, -5),
00042                               new ChunkWorldPos( 0, 0, 5), new
    ChunkWorldPos( 3, 0, -4), new ChunkWorldPos( 3, 0, 4), new
    ChunkWorldPos( 4, 0, -3), new ChunkWorldPos( 4, 0, 3),
00043                               new ChunkWorldPos( 5, 0, 0), new
    ChunkWorldPos(-5, 0, -1), new ChunkWorldPos(-5, 0, 1), new
    ChunkWorldPos(-1, 0, -5), new ChunkWorldPos(-1, 0, 5),
00044                               new ChunkWorldPos( 1, 0, -5), new
    ChunkWorldPos( 1, 0, 5), new ChunkWorldPos( 5, 0, -1), new
    ChunkWorldPos( 5, 0, 1), new ChunkWorldPos(-5, 0, -2),
00045                               new ChunkWorldPos(-5, 0, 2), new
    ChunkWorldPos(-2, 0, -5), new ChunkWorldPos(-2, 0, 5),
    ChunkWorldPos( 2, 0, -5), new ChunkWorldPos( 2, 0, 5),
00046                               new ChunkWorldPos( 5, 0, -2), new
    ChunkWorldPos( 5, 0, 2), new ChunkWorldPos(-4, 0, -4), new
    ChunkWorldPos(-4, 0, 4), new ChunkWorldPos( 4, 0, -4),
00047                               new ChunkWorldPos( 4, 0, 4), new
    ChunkWorldPos(-5, 0, -3), new ChunkWorldPos(-5, 0, 3), new
    ChunkWorldPos(-3, 0, -5), new ChunkWorldPos(-3, 0, 5),
00048                               new ChunkWorldPos( 3, 0, -5), new
    ChunkWorldPos( 3, 0, 5), new ChunkWorldPos(-6, 0, 0),
    ChunkWorldPos( 5, 0, 3), new ChunkWorldPos(-6, 0, 0),
00049                               new ChunkWorldPos( 0, 0, -6), new
    ChunkWorldPos( 0, 0, 6), new ChunkWorldPos( 6, 0, 0), new
    ChunkWorldPos(-6, 0, -1), new ChunkWorldPos(-6, 0, 1),
00050                               new ChunkWorldPos(-1, 0, -6), new
    ChunkWorldPos(-1, 0, 6), new ChunkWorldPos( 6, 0, -1),
    ChunkWorldPos( 1, 0, 6), new ChunkWorldPos( 6, 0, 1),
00051                               new ChunkWorldPos( 6, 0, 1), new
    ChunkWorldPos(-6, 0, -2), new ChunkWorldPos(-6, 0, 2), new
    ChunkWorldPos(-2, 0, -6), new ChunkWorldPos(-2, 0, 6),
00052                               new ChunkWorldPos( 2, 0, -6), new
    ChunkWorldPos( 2, 0, 6), new ChunkWorldPos( 6, 0, -2),
    ChunkWorldPos( 6, 0, 2), new ChunkWorldPos(-5, 0, -4),
00053                               new ChunkWorldPos(-5, 0, 4), new
    ChunkWorldPos(-4, 0, -5), new ChunkWorldPos(-4, 0, 5), new
    ChunkWorldPos( 4, 0, -5), new ChunkWorldPos( 4, 0, 5),
00054                               new ChunkWorldPos( 5, 0, -4), new
    ChunkWorldPos( 5, 0, 4), new ChunkWorldPos(-6, 0, -3), new
    ChunkWorldPos(-6, 0, 3), new ChunkWorldPos(-3, 0, -6),
00055                               new ChunkWorldPos(-3, 0, 6), new
    ChunkWorldPos( 3, 0, -6), new ChunkWorldPos( 3, 0, 6), new
    ChunkWorldPos( 6, 0, -3), new ChunkWorldPos( 6, 0, 3),
00056                               new ChunkWorldPos(-7, 0, 0), new
    ChunkWorldPos( 0, 0, -7), new ChunkWorldPos( 0, 0, 7), new
    ChunkWorldPos( 7, 0, 0), new ChunkWorldPos(-7, 0, -1),
00057                               new ChunkWorldPos(-7, 0, 1), new
    ChunkWorldPos(-5, 0, -5), new ChunkWorldPos(-5, 0, 5), new
    ChunkWorldPos(-1, 0, -7), new ChunkWorldPos(-1, 0, 7),
00058                               new ChunkWorldPos( 1, 0, -7), new
    ChunkWorldPos( 1, 0, 7), new ChunkWorldPos( 5, 0, -5), new
    ChunkWorldPos( 5, 0, 5), new ChunkWorldPos( 7, 0, -1),
00059                               new ChunkWorldPos( 7, 0, 1), new
    ChunkWorldPos(-6, 0, -4), new ChunkWorldPos(-6, 0, 4), new
    ChunkWorldPos(-4, 0, -6), new ChunkWorldPos(-4, 0, 6),
00060                               new ChunkWorldPos( 4, 0, -6), new
    ChunkWorldPos( 4, 0, 6), new ChunkWorldPos( 6, 0, -4), new
    ChunkWorldPos( 6, 0, 4), new ChunkWorldPos(-7, 0, -2),
00061                               new ChunkWorldPos(-7, 0, 2), new
    ChunkWorldPos(-2, 0, -7), new ChunkWorldPos(-2, 0, 7), new
    ChunkWorldPos( 2, 0, -7), new ChunkWorldPos( 2, 0, 7),
00062                               new ChunkWorldPos( 7, 0, -2), new
    ChunkWorldPos( 7, 0, 2), new ChunkWorldPos(-7, 0, -3), new
    ChunkWorldPos(-7, 0, 3), new ChunkWorldPos(-3, 0, -7),
00063                               new ChunkWorldPos(-3, 0, 7), new
    ChunkWorldPos( 3, 0, -7), new ChunkWorldPos( 3, 0, 7), new

```

```

00064     ChunkWorldPos( 7, 0, -3), new ChunkWorldPos( 7, 0,  3),
00065                     new ChunkWorldPos(-6, 0, -5), new
00066     ChunkWorldPos(-6, 0,  5), new ChunkWorldPos(-5, 0, -6), new
00067     ChunkWorldPos(-5, 0,  6), new ChunkWorldPos( 5, 0, -6),
00068                     new ChunkWorldPos( 5, 0,  6), new
00069     ChunkWorldPos( 6, 0, -5), new ChunkWorldPos( 6, 0,  5) };
00070
00071     private static ChunkWorldPos[] nearbyChunks = new
00072     ChunkWorldPos[] { new ChunkWorldPos(0, 0, 0), new
00073     ChunkWorldPos(1, 0, 0), new ChunkWorldPos(-1, 0, 0), new
00074     ChunkWorldPos(0, 0, 1), new ChunkWorldPos(0, 0, -1),
00075                                         new
00076     ChunkWorldPos(1, 0, 1), new ChunkWorldPos(1, 0, -1), new
00077     ChunkWorldPos(-1, 0, 1), new ChunkWorldPos(-1, 0, -1) };
00078
00079     private static int timer = 0;
00080 #endregion
00081
00082     private void Start()
00083     {
00084         LandGeneration.Terrain.world = world;
00085     }
00086
00087     void Update()
00088     {
00089         if (DeleteChunks())
00090             return;
00091         if (!world.chunkHasMadeCollisionMesh)
00092         {
00093             FindChunksToLoad();
00094             LoadAndRenderChunks();
00095             ApplyCollisionMeshToNearbyChunks();
00096         }
00097         /* stops chunks being made and collision meshes being made at the same time
00098         world.chunkHasMadeCollisionMesh = false;
00099     }
00100
00101     void ApplyCollisionMeshToNearbyChunks()
00102     {
00103         /* gets the player position in chunk coordinates
00104         ChunkWorldPos playerPos = new ChunkWorldPos(Mathf.FloorToInt(
00105             transform.position.x / Chunk.chunkSize) * Chunk.chunkSize, Mathf.FloorToInt(transform.
00106             position.y / Chunk.chunkSize) * Chunk.chunkSize, Mathf.FloorToInt(transform.
00107             position.z / Chunk.chunkSize) * Chunk.chunkSize);
00108
00109         for (int i = 0; i < nearbyChunks.Length; i++)
00110         {
00111             ChunkWorldPos chunkPos = new ChunkWorldPos(nearbyChunks[i].x *
00112                 Chunk.chunkSize + playerPos.x, 0, nearbyChunks[i].z * Chunk.
00113                 chunkSize + playerPos.z);
00114
00115             for (int j = -1; j < 2; j++)
00116             {
00117                 Chunk nearbyChunk = world.GetChunk(chunkPos.x, j *
00118                     Chunk.chunkSize, chunkPos.z);
00119
00120                 if (nearbyChunk != null)
00121                     nearbyChunk.applyCollisionMesh = true;
00122             }
00123         }
00124     }
00125
00126     void LoadAndRenderChunks()
00127     {
00128         /* if there is something in the build list new chunks can be made
00129         if (buildList.Count != 0)
00130         {
00131             /* makes all of the chunks in the build list. Works backwards through the list so that no
00132             chunk is missed because chunks are removed from the list as they are made
00133             for (int i = buildList.Count - 1, j = 0; i >= 0 && j < 8; i--, j++)
00134             {
00135                 BuildChunk(buildList[0]);
00136                 buildList.RemoveAt(0);
00137             }
00138         }
00139     }
00140
00141     void FindChunksToLoad()
00142     {
00143         if (buildList.Count == 0)
00144         {
00145             /* gets the player position in chunk coordinates
00146             ChunkWorldPos playerPos = new ChunkWorldPos(Mathf.FloorToInt(
00147                 transform.position.x / Chunk.chunkSize) * Chunk.chunkSize, Mathf.FloorToInt(transform.
00148                 position.y / Chunk.chunkSize) * Chunk.chunkSize, Mathf.FloorToInt(transform.
00149                 position.z / Chunk.chunkSize) * Chunk.chunkSize);
00150
00151         }
00152     }

```

```

00157             /* check all of the chunk positions and if that position does not have a chunk in it make
00158             it
00159             {
00160                 ChunkWorldPos newChunkPos = new ChunkWorldPos(chunkPositions[
00161                     i].x * Chunk.chunkSize + playerPos.x, 0, chunkPositions[i].z *
00162                     Chunk.chunkSize + playerPos.z);
00163
00164                 Chunk newChunk = world.GetChunk(newChunkPos.x, newChunkPos.
00165                     y, newChunkPos.z);
00166
00167                 if (newChunk != null && (newChunk.rendered || buildList.Contains(newChunkPos)))
00168                     continue;
00169
00170                 for (int y = -1; y < 2; y++)
00171                 {
00172                     for (int x = newChunkPos.x - Chunk.chunkSize; x < newChunkPos.
00173                         x + Chunk.chunkSize; x += Chunk.chunkSize)
00174                     {
00175                         for (int z = newChunkPos.z - Chunk.chunkSize; z < newChunkPos.
00176                             z + Chunk.chunkSize; z += Chunk.chunkSize)
00177                             buildList.Add(new ChunkWorldPos(x, y *
00178                                 Chunk.chunkSize, z));
00179                     }
00180                 }
00181             }
00182
00183             void BuildChunk(ChunkWorldPos pos)
00184             {
00185                 if (world.GetChunk(pos.x, pos.y, pos.z) == null)
00186                     world.CreateChunk(pos.x, pos.y, pos.z);
00187             }
00188
00189             bool DeleteChunks()
00190             {
00191                 /* destroys every 10 call to reduce load on CPU so that chunks are not destroyed and created
00192                 at the same time
00193                 if(timer == 10)
00194                 {
00195                     timer = 0;
00196                     var chunksToDelete = new List<ChunkWorldPos>();
00197
00198                     // *go through all of the built chunks and if the chunk is 256 units away it is assumed to
00199                     // be out of sight so is added to the destroy list
00200                     foreach (var chunk in world.chunks)
00201                     {
00202                         float distance = Vector3.Distance(chunk.Value.transform.position, transform.position);
00203
00204                         if (distance > 256)
00205                             chunksToDelete.Add(chunk.Key);
00206                     }
00207
00208                     foreach (var chunk in chunksToDelete)
00209                     {
00210                         world.DestroyChunk(chunk.x, chunk.y, chunk.z);
00211                     }
00212
00213                     return true;
00214                 }
00215             }
00216
00217             timer++;
00218
00219             return false;
00220         }
00221     }
00222
00223     return false;
00224 }
00225 }
00226 }
```

7.125 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/MeshData.cs File Reference

Classes

- class BeeGame.Terrain.Chunks.MeshData

The data for a Chunks's Mesh

Namespaces

- namespace BeeGame.Terrain.Chunks

7.126 MeshData.cs

```

00001 using System.Collections.Generic;
00002 using UnityEngine;
00003 using BeeGame.Core.Enums;
00004 using BeeGame.Core;
00005
00006 namespace BeeGame.Terrain.Chunks
00007 {
00011     public class MeshData
00012     {
00016         public List<Vector3> verts = new List<Vector3>();
00020         public List<int> tris = new List<int>();
00024         public List<Vector2> uv = new List<Vector2>();
00025
00029         public List<Vector3> colVerts = new List<Vector3>();
00033         public List<int> colTris = new List<int>();
00034
00038         public bool shareMeshes = true;
00039
00040         public bool done = false;
00041
00046         public void AddQuadTriangles(bool addToRenderMesh = true)
00047         {
00048             /*adds the triangles in an anticlockwise order
00049
00050             if (addToRenderMesh)
00051             {
00052                 tris.Add(verts.Count - 4);
00053                 tris.Add(verts.Count - 3);
00054                 tris.Add(verts.Count - 2);
00055                 tris.Add(verts.Count - 4);
00056                 tris.Add(verts.Count - 2);
00057                 tris.Add(verts.Count - 1);
00058             }
00059
00060             colTris.Add(colVerts.Count - 4);
00061             colTris.Add(colVerts.Count - 3);
00062             colTris.Add(colVerts.Count - 2);
00063             colTris.Add(colVerts.Count - 4);
00064             colTris.Add(colVerts.Count - 2);
00065             colTris.Add(colVerts.Count - 1);
00066         }
00067
00074         public void AddVertices(THVector3 pos, bool addToRenderMesh = true,
00075             Direction direction = Direction.DOWN)
00076         {
00077             if (addToRenderMesh)
00078                 verts.Add(pos);
00079
00080             /* if the vertex is on the top face make its position slightly smaller
00081             if(direction == Direction.UP)
00082                 colVerts.Add(pos - new THVector3(0.01f, 0, 0.01f));
00083         }
00091         public void AddTriangle(int tri)
00092         {
00093             tris.Add(tri);
00094
00095             colTris.Add(tri - (verts.Count - colVerts.Count));
00096         }
00097     }
00098 }
```

7.127 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/Terrain/Chunks/SaveChunk.cs File Reference

Classes

- class BeeGame.Terrain.Chunks.SaveChunk
Saves a Chunks modified Blocks for save optimisation

Namespaces

- namespace BeeGame.Terrain.Chunks

7.128 SaveChunk.cs

```

00001 using System;
00002 using System.Collections.Generic;
00003 using BeeGame.Blocks;
00004
00005
00006 namespace BeeGame.Terrain.Chunks
00007 {
00011     [Serializable]
00012     public class SaveChunk
00013     {
00017         public Dictionary<ChunkWorldPos, Block> blocks = new Dictionary<ChunkWorldPos, Block>();
00018
00023         public SaveChunk(Block[, ,] blockArray)
00024         {
00025             for (int x = 0; x < Chunk.chunkSize; x++)
00026             {
00027                 for (int y = 0; y < Chunk.chunkSize; y++)
00028                 {
00029                     for (int z = 0; z < Chunk.chunkSize; z++)
00030                     {
00031                         /* if the block has changed save it
00032                         if (blockArray[x, y, z].changed)
00033                             blocks.Add(new ChunkWorldPos(x, y, z), blockArray[x, y, z]);
00034                     }
00035                 }
00036             }
00037         }
00038     }
00039 }
```

7.129 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/ChunkWorld← Pos.cs File Reference

Classes

- struct BeeGame.Terrain.ChunkWorldPos
Serializable int version of THVector3

Namespaces

- namespace BeeGame.Terrain

7.130 ChunkWorldPos.cs

```

00001 using System;
00002 using BeeGame.Core;
00003
00004 namespace BeeGame.Terrain
00005 {
00009     [Serializable]
0010     public struct ChunkWorldPos
0011     {
0015         public int x, y, z;
0016
0023         public ChunkWorldPos(int x, int y, int z)
0024         {
0025             this.x = x;
0026             this.y = y;
0027             this.z = z;
0028         }
0029     }
0030 }
```

```

00029
00034     public override string ToString()
00035     {
00036         return $"({x}, {y}, {z})";
00037     }
00038
00039     /* TODO probly add the == and != but for now this is fine
00040     [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "
00041     CA2231:OverloadOperatorEqualsOnOverridingValueTypeEquals")]
00042     public override bool Equals(object obj)
00043     {
00044         /* possibly remove and just check if obj is null
00045         if (!(obj is ChunkWorldPos))
00046             return false;
00047
00048         ChunkWorldPos temp = (ChunkWorldPos) obj;
00049
00050         /* possibly change to hashCode checking
00051         if (temp.x == x && temp.y == y && temp.z == z)
00052             return true;
00053
00054         return false;
00055     }
00056
00057     public override int GetHashCode()
00058     {
00059         unchecked
00060         {
00061             int hashCode = 47;
00062
00063             hashCode *= 227 + x.GetHashCode();
00064             hashCode *= 227 + y.GetHashCode();
00065             hashCode *= 227 + z.GetHashCode();
00066
00067             return hashCode;
00068         }
00069     }
00070
00071     public static implicit operator THVector3(ChunkWorldPos pos)
00072     {
00073         return new THVector3(pos.x, pos.y, pos.z);
00074     }
00075
00076     public static explicit operator ChunkWorldPos(THVector3 pos)
00077     {
00078         return new ChunkWorldPos((int)pos.x, (int)pos.y, (int)pos.
00079         z);
00080     }
00081 }
00082 }
```

7.131 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/LandGeneration/Noise/SimplexNoise.cs File Reference

Classes

- class BeeGame.Terrain.LandGeneration.Noise.SimplexNoise

Implementation of the Perlin simplex noise, an improved Perlin noise algorithm. Based loosely on SimplexNoise1234 by Stefan Gustavson <http://staffwww.itn.liu.se/~stegu/aqsis/aqsis-newnoise/>

Namespaces

- namespace BeeGame.Terrain.LandGeneration.Noise

7.132 SimplexNoise.cs

```

00001 /* SimplexNoise for C#
00002 /* Author: Heikki Törmälä
00003
00004 /* This is free and unencumbered software released into the public domain.
00005
```

```

00006 /*Anyone is free to copy, modify, publish, use, compile, sell, or
00007 /*distribute this software, either in source code form or as a compiled
00008 /*binary, for any purpose, commercial or non-commercial, and by any
00009 /*means.
0010
0011 /*In jurisdictions that recognize copyright laws, the author or authors
0012 /*of this software dedicate any and all copyright interest in the
0013 /*software to the public domain. We make this dedication for the benefit
0014 /*of the public at large and to the detriment of our heirs and
0015 /*successors. We intend this dedication to be an overt act of
0016 /*relinquishment in perpetuity of all present and future rights to this
0017 /*software under copyright law.
0018
0019 /*THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
0020 /*EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
0021 /*MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
0022 /*IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY CLAIM, DAMAGES OR
0023 /*OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
0024 /*ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
0025 /*OTHER DEALINGS IN THE SOFTWARE.
0026
0027 /*For more information, please refer to <http://unlicense.org/>
0028
0029
0030 namespace BeeGame.Terrain.LandGeneration.Noise
0031 {
0032     public class SimplexNoise
0033     {
0034         public static float Generate(float x)
0035         {
0036             int i0 = FastFloor(x);
0037             int i1 = i0 + 1;
0038             float x0 = x - i0;
0039             float x1 = x0 - 1.0f;
0040
0041             float n0, n1;
0042
0043             float t0 = 1.0f - x0 * x0;
0044             t0 *= t0;
0045             n0 = t0 * t0 * grad(perm[i0 & 0xff], x0);
0046
0047             float t1 = 1.0f - x1 * x1;
0048             t1 *= t1;
0049             n1 = t1 * t1 * grad(perm[i1 & 0xff], x1);
0050             /* The maximum value of this noise is 8*(3/4)^4 = 2.53125
0051             /* A factor of 0.395 scales to fit exactly within [-1,1]
0052             return 0.395f * (n0 + n1);
0053         }
0054
0055         public static float Generate(float x, float y)
0056         {
0057             const float F2 = 0.366025403f; /* F2 = 0.5*(sqrt(3.0)-1.0)
0058             const float G2 = 0.211324865f; /* G2 = (3.0-Math.sqrt(3.0))/6.0
0059
0060             float n0, n1, n2; /* Noise contributions from the three corners
0061
0062             /* Skew the input space to determine which simplex cell we're in
0063             float s = (x + y) * F2; /* Hairy factor for 2D
0064             float xs = x + s;
0065             float ys = y + s;
0066             int i = FastFloor(xs);
0067             int j = FastFloor(ys);
0068
0069             float t = (float)(i + j) * G2;
0070             float X0 = i - t; /* Unskew the cell origin back to (x,y) space
0071             float Y0 = j - t;
0072             float x0 = x - X0; /* The x,y distances from the cell origin
0073             float y0 = y - Y0;
0074
0075             /* For the 2D case, the simplex shape is an equilateral triangle.
0076             /* Determine which simplex we are in.
0077             int il, jl; /* Offsets for second (middle) corner of simplex in (i,j) coords
0078             if (x0 > y0) { il = 1; jl = 0; } /* lower triangle, XY order: (0,0)->(1,0)->(1,1)
0079             else { il = 0; jl = 1; } /* upper triangle, YX order: (0,0)->(0,1)->(1,1)
0080
0081             /* A step of (1,0) in (i,j) means a step of (1-c,-c) in (x,y), and
0082             /* a step of (0,1) in (i,j) means a step of (-c,1-c) in (x,y), where
0083             /* c = (3-sqrt(3))/6
0084
0085             float x1 = x0 - il + G2; /* Offsets for middle corner in (x,y) unskewed coords
0086             float y1 = y0 - jl + G2;
0087             float x2 = x0 - 1.0f + 2.0f * G2; /* Offsets for last corner in (x,y) unskewed coords
0088             float y2 = y0 - 1.0f + 2.0f * G2;
0089
0090             /* Wrap the integer indices at 256, to avoid indexing perm[] out of bounds
0091             int ii = i % 256;
0092             int jj = j % 256;

```

```

00109
00110     /* Calculate the contribution from the three corners
00111     float t0 = 0.5f - x0 * x0 - y0 * y0;
00112     if (t0 < 0.0f) n0 = 0.0f;
00113     else
00114     {
00115         t0 *= t0;
00116         n0 = t0 * t0 * grad(perm[ii + perm[jj]], x0, y0);
00117     }
00118
00119     float t1 = 0.5f - x1 * x1 - y1 * y1;
00120     if (t1 < 0.0f) n1 = 0.0f;
00121     else
00122     {
00123         t1 *= t1;
00124         n1 = t1 * t1 * grad(perm[ii + il + perm[jj + j1]], x1, y1);
00125     }
00126
00127     float t2 = 0.5f - x2 * x2 - y2 * y2;
00128     if (t2 < 0.0f) n2 = 0.0f;
00129     else
00130     {
00131         t2 *= t2;
00132         n2 = t2 * t2 * grad(perm[ii + 1 + perm[jj + 1]], x2, y2);
00133     }
00134
00135     /* Add contributions from each corner to get the final noise value.
00136     /* The result is scaled to return values in the interval [-1,1].
00137     return 40.0f * (n0 + n1 + n2); /* TODO: The scale factor is preliminary!
00138 }
00139
00140
00141     public static float Generate(float x, float y, float z)
00142     {
00143         /* Simple skewing factors for the 3D case
00144         const float F3 = 0.333333333f;
00145         const float G3 = 0.166666667f;
00146
00147         float n0, n1, n2, n3; /* Noise contributions from the four corners
00148
00149         /* Skew the input space to determine which simplex cell we're in
00150         float s = (x + y + z) * F3; /* Very nice and simple skew factor for 3D
00151         float xs = x + s;
00152         float ys = y + s;
00153         float zs = z + s;
00154         int i = FastFloor(xs);
00155         int j = FastFloor(ys);
00156         int k = FastFloor(zs);
00157
00158         float t = (float)(i + j + k) * G3;
00159         float X0 = i - t; /* Unskew the cell origin back to (x,y,z) space
00160         float Y0 = j - t;
00161         float Z0 = k - t;
00162         float x0 = x - X0; /* The x,y,z distances from the cell origin
00163         float y0 = y - Y0;
00164         float z0 = z - Z0;
00165
00166         /* For the 3D case, the simplex shape is a slightly irregular tetrahedron.
00167         /* Determine which simplex we are in.
00168         int i1, j1, k1; /* Offsets for second corner of simplex in (i,j,k) coords
00169         int i2, j2, k2; /* Offsets for third corner of simplex in (i,j,k) coords
00170
00171         /* This code would benefit from a backport from the GLSL version! */
00172         if (x0 >= y0)
00173         {
00174             if (y0 >= z0)
00175                 { i1 = 1; j1 = 0; k1 = 0; i2 = 1; j2 = 1; k2 = 0; } /* X Y Z order
00176                 else if (x0 >= z0) { i1 = 1; j1 = 0; k1 = 0; i2 = 1; j2 = 0; k2 = 1; } /* X Z Y order
00177                 else { i1 = 0; j1 = 0; k1 = 1; i2 = 1; j2 = 0; k2 = 1; } /* Z X Y order
00178             }
00179             else
00180                 { /* x0<y0
00181                     if (y0 < z0) { i1 = 0; j1 = 0; k1 = 1; i2 = 0; j2 = 1; k2 = 1; } /* Z Y X order
00182                     else if (x0 < z0) { i1 = 0; j1 = 1; k1 = 0; i2 = 0; j2 = 1; k2 = 1; } /* Y Z X order
00183                     else { i1 = 0; j1 = 1; k1 = 0; i2 = 1; j2 = 1; k2 = 0; } /* Y X Z order
00184                 }
00185
00186         /* A step of (1,0,0) in (i,j,k) means a step of (1-c,-c,-c) in (x,y,z),
00187         /* a step of (0,1,0) in (i,j,k) means a step of (-c,1-c,-c) in (x,y,z), and
00188         /* a step of (0,0,1) in (i,j,k) means a step of (-c,-c,1-c) in (x,y,z), where
00189         /* c = 1/6.
00190
00191         float x1 = x0 - i1 + G3; /* Offsets for second corner in (x,y,z) coords
00192         float y1 = y0 - j1 + G3;
00193         float z1 = z0 - k1 + G3;
00194         float x2 = x0 - i2 + 2.0f * G3; /* Offsets for third corner in (x,y,z) coords
00195         float y2 = y0 - j2 + 2.0f * G3;

```

```

00196     float z2 = z0 - k2 + 2.0f * G3;
00197     float x3 = x0 - 1.0f + 3.0f * G3; /* Offsets for last corner in (x,y,z) coords
00198     float y3 = y0 - 1.0f + 3.0f * G3;
00199     float z3 = z0 - 1.0f + 3.0f * G3;
00200
00201     /* Wrap the integer indices at 256, to avoid indexing perm[] out of bounds
00202     int ii = Mod(i, 256);
00203     int jj = Mod(j, 256);
00204     int kk = Mod(k, 256);
00205
00206     /* Calculate the contribution from the four corners
00207     float t0 = 0.6f - x0 * x0 - y0 * y0 - z0 * z0;
00208     if (t0 < 0.0f) n0 = 0.0f;
00209     else
00210     {
00211         t0 *= t0;
00212         n0 = t0 * t0 * grad(perm[ii + perm[jj + perm[kk]]], x0, y0, z0);
00213     }
00214
00215     float t1 = 0.6f - x1 * x1 - y1 * y1 - z1 * z1;
00216     if (t1 < 0.0f) n1 = 0.0f;
00217     else
00218     {
00219         t1 *= t1;
00220         n1 = t1 * t1 * grad(perm[ii + i1 + perm[jj + j1 + perm[kk + k1]]], x1, y1, z1);
00221     }
00222
00223     float t2 = 0.6f - x2 * x2 - y2 * y2 - z2 * z2;
00224     if (t2 < 0.0f) n2 = 0.0f;
00225     else
00226     {
00227         t2 *= t2;
00228         n2 = t2 * t2 * grad(perm[ii + i2 + perm[jj + j2 + perm[kk + k2]]], x2, y2, z2);
00229     }
00230
00231     float t3 = 0.6f - x3 * x3 - y3 * y3 - z3 * z3;
00232     if (t3 < 0.0f) n3 = 0.0f;
00233     else
00234     {
00235         t3 *= t3;
00236         n3 = t3 * t3 * grad(perm[ii + i1 + perm[jj + j1 + perm[kk + 1]]], x3, y3, z3);
00237     }
00238
00239     /* Add contributions from each corner to get the final noise value.
00240     /* The result is scaled to stay just inside [-1,1]
00241     return 32.0f * (n0 + n1 + n2 + n3); /* TODO: The scale factor is preliminary!
00242 }
00243
00244 public static byte[] perm = new byte[512] { 151,160,137,91,90,15,
00245     131,13,201,95,96,53,194,233,7,225,140,36,103,30,69,142,8,99,37,240,21,10,23,
00246     190, 6,148,247,120,234,75,0,26,197,62,94,252,219,203,117,35,11,32,57,177,33,
00247     88,237,149,56,87,174,20,125,136,171,168, 68,175,74,165,71,134,139,48,27,166,
00248     77,146,158,231,83,111,229,122,60,211,133,230,220,105,92,41,55,46,245,40,244,
00249     102,143,54, 65,25,63,161, 1,216,80,73,209,76,132,187,208, 89,18,169,200,196,
00250     135,130,116,188,159,86,164,100,109,198,173,186, 3,64,52,217,226,250,124,123,
00251     5,202,38,147,118,126,255,82,85,212,207,206,59,227,47,16,58,17,182,189,28,42,
00252     223,183,170,213,119,248,152, 2,44,154,163, 70,221,153,101,155,167, 43,172,9,
00253     129,22,39,253, 19,98,108,110,79,113,224,232,178,185, 112,104,218,246,97,228,
00254     251,34,242,193,238,210,144,12,191,179,162,241, 81,51,145,235,249,14,239,107,
00255     49,192,214, 31,181,199,106,157,184, 84,204,176,115,121,50,45,127, 4,150,254,
00256     138,236,205,93,222,114,67,29,24,72,243,141,128,195,78,66,215,61,156,180,
00257     151,160,137,91,90,15,
00258     131,13,201,95,96,53,194,233,7,225,140,36,103,30,69,142,8,99,37,240,21,10,23,
00259     190, 6,148,247,120,234,75,0,26,197,62,94,252,219,203,117,35,11,32,57,177,33,
00260     88,237,149,56,87,174,20,125,136,171,168, 68,175,74,165,71,134,139,48,27,166,
00261     77,146,158,231,83,111,229,122,60,211,133,230,220,105,92,41,55,46,245,40,244,
00262     102,143,54, 65,25,63,161, 1,216,80,73,209,76,132,187,208, 89,18,169,200,196,
00263     135,130,116,188,159,86,164,100,109,198,173,186, 3,64,52,217,226,250,124,123,
00264     5,202,38,147,118,126,255,82,85,212,207,206,59,227,47,16,58,17,182,189,28,42,
00265     223,183,170,213,119,248,152, 2,44,154,163, 70,221,153,101,155,167, 43,172,9,
00266     129,22,39,253, 19,98,108,110,79,113,224,232,178,185, 112,104,218,246,97,228,
00267     251,34,242,193,238,210,144,12,191,179,162,241, 81,51,145,235,249,14,239,107,
00268     49,192,214, 31,181,199,106,157,184, 84,204,176,115,121,50,45,127, 4,150,254,
00269     138,236,205,93,222,114,67,29,24,72,243,141,128,195,78,66,215,61,156,180
00270 };
00271
00272     private static int FastFloor(float x)
00273     {
00274         return (x > 0) ? ((int)x) : (((int)x) - 1);
00275     }
00276
00277     private static int Mod(int x, int m)
00278     {
00279         int a = x % m;
00280         return a < 0 ? a + m : a;
00281     }
00282

```

```

00283     private static float grad(int hash, float x)
00284     {
00285         int h = hash & 15;
00286         float grad = 1.0f + (h & 7);    /* Gradient value 1.0, 2.0, ..., 8.0
00287         if ((h & 8) != 0) grad = -grad;    /* Set a random sign for the gradient
00288         return (grad * x);            /* Multiply the gradient with the distance
00289     }
00290
00291     private static float grad(int hash, float x, float y)
00292     {
00293         int h = hash & 7;        /* Convert low 3 bits of hash code
00294         float u = h < 4 ? x : y; /* into 8 simple gradient directions,
00295         float v = h < 4 ? y : x; /* and compute the dot product with (x,y).
00296         return ((h & 1) != 0 ? -u : u) + ((h & 2) != 0 ? -2.0f * v : 2.0f * v);
00297     }
00298
00299     private static float grad(int hash, float x, float y, float z)
00300     {
00301         int h = hash & 15;      /* Convert low 4 bits of hash code into 12 simple
00302         float u = h < 8 ? x : y; /* gradient directions, and compute dot product.
00303         float v = h < 4 ? y : h == 12 || h == 14 ? x : z; /* Fix repeats at h = 12 to 15
00304         return ((h & 1) != 0 ? -u : u) + ((h & 2) != 0 ? -v : v);
00305     }
00306
00307     private static float grad(int hash, float x, float y, float z, float t)
00308     {
00309         int h = hash & 31;      /* Convert low 5 bits of hash code into 32 simple
00310         float u = h < 24 ? x : y; /* gradient directions, and compute dot product.
00311         float v = h < 16 ? y : z;
00312         float w = h < 8 ? z : t;
00313         return ((h & 1) != 0 ? -u : u) + ((h & 2) != 0 ? -v : v) + ((h & 4) != 0 ? -w : w);
00314     }
00315 }
00316 }
```

7.133 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/LandGeneration/← Terrain.cs File Reference

Classes

- class BeeGame.Terrain.LandGeneration.Terrain
Should use as an interface between the rest of the game and the terrain

Namespaces

- namespace BeeGame.Terrain.LandGeneration

7.134 Terrain.cs

```

00001 using System;
00002 using UnityEngine;
00003 using BeeGame.Terrain.Chunks;
00004 using BeeGame.Blocks;
00005 using BeeGame.Core;
00006
00007 namespace BeeGame.Terrain.LandGeneration
00008 {
00012     public class Terrain
00013     {
00014         public static World world;
00015
00016         #region Setting Position To block Grid
00017         public static ChunkWorldPos GetBlockPos(THVector3 pos)
00023         {
00024             return new ChunkWorldPos()
00025             {
00026                 x = Mathf.RoundToInt(pos.x),
00027                 y = Mathf.RoundToInt(pos.y),
00028                 z = Mathf.RoundToInt(pos.z)
00029             };
00030         }
00031 }
```

```
00038     public static THVector3 GetBlockPos(RaycastHit hit)
00039     {
00040         THVector3 vec3 = new THVector3()
00041         {
00042             x = RoundXZ(hit.point.x, hit.normal.x),
00043             y = RoundY(hit.point.y, hit.normal.y),
00044             z = RoundXZ(hit.point.z, hit.normal.z)
00045         };
00046         return (vec3);
00047     }
00048
00049
00050
00051
00052
00053
00054     public static ChunkWorldPos GetBlockPosFromRayCast(RaycastHit
00055     hit)
00056     {
00057         return new ChunkWorldPos((int)RoundXZ(hit.point.x, hit.normal.x), (int)RoundY(hit.
00058         point.y, hit.normal.y), (int)RoundXZ(hit.point.z, hit.normal.z));
00059     }
00060
00061
00062
00063
00064
00065     static float RoundXZ(float pos, float normal)
00066     {
00067         /* if we are looking at + x/z vlaues
00068         if (pos > 0)
00069         {
00070             if (normal > 0)
00071             {
00072                 pos = (int)pos;
00073                 return pos;
00074             }
00075             else if (normal < 0)
00076             {
00077                 pos = (int)pos;
00078                 return pos - 1;
00079             }
00080             else
00081             {
00082                 if ((pos - (int)pos) > 0.5)
00083                 {
00084                     return (int)pos + 1;
00085                 }
00086                 return (int)pos;
00087             }
00088         }
00089         /* if we are looking at - x/z values
00090         else
00091         {
00092             /* if poitive normal
00093             if (normal > 0)
00094             {
00095                 pos = (int)pos;
00096                 return pos - 1;
00097             }
00098             /* if negative nomrml
00099             if (normal < 0)
00100             {
00101                 pos = (int)pos;
00102                 return pos;
00103             }
00104             /* if their is no normal
00105             if (normal == 0)
00106             {
00107                 /* if pos is greater than 0.5 we are in the next block so go to it
00108                 if ((-pos - (int)-pos) > 0.5)
00109                 {
00110                     return (int)pos - 1;
00111                 }
00112                 return (int)pos;
00113             }
00114         }
00115     }
00116
00117     static float RoundY(float pos, float normal)
00118     {
00119         pos = (float)Math.Round(pos, 1);
00120         if (pos >= 0)
00121         {
00122             if(normal > 0)
00123             {
00124                 if((int)pos % 2 == 0)
00125                     return Mathf.RoundToInt((float)Math.Round(pos, 1));
00126
00127                 return Mathf.RoundToInt((float)Math.Round(pos, 1)) - normal;
00128             }
00129             if((int)pos % 2 == 0)
00130                 return Mathf.RoundToInt((float)Math.Round(pos, 1)) - normal;
00131
00132             return Mathf.RoundToInt((float)Math.Round(pos, 1));
00133         }
00134     }
```

```

00146         }
00147
00148     if(pos <= 0)
00149     {
00150         if (normal > 0)
00151         {
00152             if ((int)pos % 2 == 0)
00153                 /* the Math.Round removes strange rounding errors shown with Mathf.Round eg
sometimes 0.5 would round to 0 not 1
00154                     return Mathf.RoundToInt((float)Math.Round(pos, 1)) - normal;
00155
00156                     return Mathf.RoundToInt((float)Math.Round(pos, 1));// - normal;
00157                 }
00158
00159             if ((int)pos % 2 == 0)
00160                 return Mathf.RoundToInt((float)Math.Round(pos, 1));
00161
00162             return Mathf.RoundToInt((float)Math.Round(pos, 1)) - normal;
00163         }
00164
00165
00166         return Mathf.RoundToInt((float)Math.Round(pos, 1));
00167     }
00168
00169     public static float Round(float pos, float norm, bool adjacent = false)
00170     {
00171         if(pos - (int)pos == 0.5f || pos - (int)pos == -0.5f)
00172         {
00173             if(adjacent)
00174             {
00175                 pos += (norm / 2);
00176             }
00177             else
00178             {
00179                 pos -= (norm / 2);
00180             }
00181         }
00182
00183         return pos;
00184     }
00185 #endregion
00186
00187 #region Get Block
00188 public static ChunkWorldPos GetBlockPos(RaycastHit hit, bool adjacent = false)
00189 {
00190     return GetBlockPos(new THVector3()
00191     {
00192         ///* rounds the hit to the correct position
00193         x = Round(hit.point.x, hit.normal.x, adjacent),
00194         y = Round(hit.point.y, hit.normal.y, adjacent),
00195         z = Round(hit.point.z, hit.normal.z, adjacent)
00196     });
00197 }
00198
00199 public static Block GetBlock(RaycastHit hit, bool adjacent = false)
00200 {
00201     ///* checks that a chunk was hit and if it wasnt return early
00202     Chunk chunk = hit.collider.GetComponent<Chunk>();
00203
00204     if (chunk == null)
00205         return null;
00206
00207     ///* allignes the hit to the block grid and returns the block
00208     ChunkWorldPos pos = GetBlockPos(hit, adjacent);
00209
00210     return chunk.world.GetBlock(pos.x, pos.y, pos.z);
00211 }
00212
00213 public static Block GetBlock(THVector3 pos)
00214 {
00215     Chunk chunk = GetChunk(pos);
00216
00217     if (chunk == null)
00218         return new Air();
00219
00220     chunk.world.GetBlock((int)pos.x, (int)pos.y, (int)pos.z);
00221
00222     return new Block();
00223 }
00224
00225 public static bool BlockInPosition(THVector3 pos,
00226     Chunk chunk)
00227 {
00228     if (chunk == null)
00229         return false;
00230
00231     if (chunk.GetBlock((int)pos.x, (int)pos.y, (int)pos.z) != new

```

```
00253     Air())
00254         return true;
00255     }
00256 }
00257 #endregion
00258
00259     public static Chunk GetChunk(THVector3 vec3)
00260     {
00261         return world.GetChunk((int)vec3.x, (int)vec3.y, (int)vec3.
00262 z);
00263     }
00264
00265 #region Set Block
00266     public static bool SetBlock(RaycastHit hit, Block block, bool adjacent = false)
00267     {
00268         /* checks that a chnk was hit
00269         Chunk chunk = hit.collider.GetComponent<Chunk>();
00270
00271         if (chunk == null)
00272             return false;
00273
00274         /* alligns the hit to the block grid
00275         ChunkWorldPos pos = GetBlockPosFromRayCast(hit);
00276
00277         /* checks that the block trying to be replaced can be replaced eg bedrock cannot be replaced
00278         if (GetBlock(hit, adjacent).breakable)
00279         {
00280             /* sets the position of the block and saves the chunk
00281             chunk.world.SetBlock(pos.x, pos.y, pos.z, block);
00282             Serialization.Serialization.SaveChunk(chunk);
00283         }
00284
00285         return true;
00286     }
00287 #endregion
00288 }
```

7.135 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/LandGeneration/←
TerrainGeneration.cs File Reference

Classes

- class BeeGame.Terrain.LandGeneration.TerrainGeneration
Generates the terrain for the game

Namespaces

- namespace BeeGame.Terrain.LandGeneration

7.136 TerrainGeneration.cs

```
00001 using UnityEngine;
00002 using BeeGame.Terrain.Chunks;
00003 using BeeGame.Terrain.LandGeneration.Noise;
00004 using BeeGame.Serialization;
00005 using System.Collections.Generic;
00006 using System.Threading;
00007
00008 namespace BeeGame.Terrain.LandGeneration
00009 {
00013     public class TerrainGeneration
00014     {
00015         #region Data
00016         private float stoneBaseHeight = -24;
00023         private float stoneBaseNoise = 0.05f;
00027         private float stoneBaseNoiseHeight = 4;
00028
00032         private float stoneMountainHeight = 48;
00036         private float stoneMountainFrequency = 0.008f;
00040         private float stoneMinHeight = -12;
```

```

00041
00045     private float dirtBaseHeight = 1;
00049     private float dirtNoise = 0.04f;
00053     private float dirtNoiseHeight = 3;
00054
00058     private float treeFrequency = 0.2f;
00062     private int treeDensity = 3;
00063
00067     private float caveFrequency = 0.025f;
00071     private int caveSize = 8;
00072 #endregion
00073
00079     public Chunk ChunkGen(Chunk chunk)
00080     {
00081         Chunk outChunk = chunk;
00082         lock (chunk)
00083         {
00084             Thread thread = new Thread(() => ChunkGenThread(chunk, out outChunk)) { Name = $"Generate
00085             Chunk Thread @ {chunk.chunkWorldPos}"};
00086
00087             thread.Start();
00088             return outChunk;
00089         }
00090     }
00096     public void ChunkGenThread(Chunk chunk, out Chunk outChunk)
00097     {
00098         /* for each x and z position in teh chunk
00099         for (int x = chunk.chunkWorldPos.x-3; x < chunk.
00100             chunkWorldPos.x + Chunk.chunkSize + 3; x++)
00101             {
00102                 for (int z = chunk.chunkWorldPos.z-3; z < chunk.
00103                     chunkWorldPos.z + Chunk.chunkSize + 3; z++)
00104                     {
00105                         chunk = GenChunkColum(chunk, x, z);
00106
00107                         chunk.SetBlocksUnmodified();
00108                         outChunk = chunk;
00109                     }
00110
00118         public Chunk GenChunkColum(Chunk chunk, int x, int z)
00119         {
00120             /* the height of the mountain
00121             int stoneHeight = Mathf.FloorToInt(stoneBaseHeight);
00122             stoneHeight += GetNoise(-x, 0, z, stoneMountainFrequency, Mathf.FloorToInt(stoneMountainHeight)
00123             );
00124
00125             /* if the colum is currently to low make it not so low
00126             if (stoneHeight < stoneMinHeight)
00127                 stoneHeight = Mathf.FloorToInt(stoneMinHeight);
00128
00129             /* add the height of normal stone on to the mountain
00130             stoneHeight += GetNoise(x, 0, -z, stoneBaseNoise, Mathf.RoundToInt(stoneBaseNoiseHeight));
00131
00132             /*put dirt on top
00133             int dirtHeight = stoneHeight + Mathf.FloorToInt(dirtBaseHeight);
00134             dirtHeight += GetNoise(x, 100, z, dirtNoise, Mathf.FloorToInt(dirtNoiseHeight));
00135
00136             /* set the colum to the correct blocks
00137             for (int y = chunk.chunkWorldPos.y - 8; y < chunk.
00138                 chunkWorldPos.y + Chunk.chunkSize; y++)
00139                 {
00140                     int caveChance = GetNoise(x + 40, y + 100, z - 50, caveFrequency, 200);
00141
00142                     /* puts a layer of bedrock at the botton the the world
00143                     if (y <= (chunk.chunkWorldPos.y) && chunk.
00144                         chunkWorldPos.y == -16)
00145                         {
00146                             SetBlock(x, y, z, new Blocks.Bedrock(), chunk);
00147                         }
00148                     else if (y <= stoneHeight && caveSize < caveChance)
00149                         {
00150                             SetBlock(x, y, z, new Blocks.Block(), chunk);
00151                         }
00152                     else if (y <= dirtHeight && caveSize < caveChance)
00153                         {
00154                             SetBlock(x, y, z, new Blocks.Grass(), chunk);
00155                             if (y == dirtHeight && GetNoise(x, 0, z, treeFrequency, 100) < treeDensity)
00156                                 CreateTree(x, y + 1, z, chunk);
00157                         }
00158                     else
00159                         {
00160                             SetBlock(x, y, z, new Blocks.Air(), chunk);
00161                         }
00162                 }
00163             }
00164         }
00165     }

```

```

00160             return chunk;
00161         }
00163
00173     public static int GetNoise(int x, int y, int z, float scale, int max)
00174     {
00175         return Mathf.FloorToInt((SimplexNoise.Generate(x * scale, y * scale, z *
00176         scale) + 1f) * (max / 2f));
00177     }
00178
00187     public static void SetBlock(int x, int y, int z, Blocks.Block block,
00188     Chunk chunk, bool replacesBlocks = false)
00189     {
00190         /* corrects the x, y, z pos of the so that the block is placed in the correct position
00191         x -= chunk.chunkWorldPos.x;
00192         y -= chunk.chunkWorldPos.y;
00193         z -= chunk.chunkWorldPos.z;
00194
00195         /* checks that the block is in the chunk and that no block is already their then sets it
00196         if (Chunk.InRange(x) && Chunk.InRange(y) &&
00197             Chunk.InRange(z))
00198             if (replacesBlocks || chunk.blocks[x, y, z] == null)
00199                 chunk.SetBlock(x, y, z, block, false);
00200
00210     void CreateTree(int x, int y, int z, Chunk chunk)
00211     {
00212         /* makes the leaves of teh tree
00213         for (int xi = -2; xi <= 2; xi++)
00214         {
00215             for (int yi = 4; yi <= 8; yi++)
00216             {
00217                 for (int zi = -2; zi <= 2; zi++)
00218                     SetBlock(xi + x, yi + y, zi + z, new Blocks.Leaves(), chunk, true);
00219             }
00220         }
00221
00222         /* makes the trunk of the tree
00223         for (int i = 0; i < 6; i++)
00224         {
00225             SetBlock(x, y + i, z, new Blocks.Wood(), chunk, true);
00226         }
00227     }
00228 }
00229 }
00230 }
00231 }

```

7.137 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/LandGeneration/World.cs File Reference

Classes

- class [BeeGame.Terrain.LandGeneration.World](#)

Allows inter Chunk communication as it stores a list of active chunks

Namespaces

- namespace [BeeGame.Terrain.LandGeneration](#)

7.138 World.cs

```

00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using System.Text;
00005 using UnityEngine;
00006 using BeeGame.Terrain.Chunks;
00007 using BeeGame.Blocks;
00008
00009 namespace BeeGame.Terrain.LandGeneration
00010 {

```

```

00014     public class World : MonoBehaviour
00015     {
00016         #region Data
00017         public Dictionary<ChunkWorldPos, Chunk> chunks = new Dictionary<ChunkWorldPos, Chunk>();
00021
00025         public GameObject chunkPrefab;
00026
00030         public bool chunkHasMadeCollisionMesh = false;
00031     #endregion
00032
00033         #region Creation and Destruction
00034         #region Chunk
00035         public void CreateChunk(int x, int y, int z)
00036         {
00043             /* pos of the chunk
00044             ChunkWorldPos pos = new ChunkWorldPos(x, y, z);
00045
00046             /* makes the chunk at the given position
00047             GameObject newChunk = Instantiate(chunkPrefab, new Vector3(x, y, z), Quaternion.identity);
00048
00049             Chunk chunk = newChunk.GetComponent<Chunk>();
00050
00051             /* setting the chunks pos and a reference to this
00052             chunk.chunkWorldPos = pos;
00053             chunk.world = this;
00054
00055             /* adds the new chunk to the dictionary
00056             chunks.Add(pos, chunk);
00057
00058             /* generates the new chunks blocks
00059             chunk = new TerrainGeneration().ChunkGen(chunk);
00060
00061             //loads any blocks that the chunk has had modified
00062             Serialization.Serialization.LoadChunk(chunk);
00063
00064             /* updates all chunks around this one to reduce drawing of unecessary faces
00065             chunks.TryGetValue(new ChunkWorldPos(x, y - 16, z), out chunk);
00066             if (chunk != null)
00067                 chunk.update = true;
00068
00069             chunks.TryGetValue(new ChunkWorldPos(x, y, z - 16), out chunk);
00070             if (chunk != null)
00071                 chunk.update = true;
00072
00073             chunks.TryGetValue(new ChunkWorldPos(x - 16, y, z), out chunk);
00074             if (chunk != null)
00075                 chunk.update = true;
00076
00077             chunks.TryGetValue(new ChunkWorldPos(x, y + 16, z), out chunk);
00078             if (chunk != null)
00079                 chunk.update = true;
00080
00081             chunks.TryGetValue(new ChunkWorldPos(x, y, z + 16), out chunk);
00082             if (chunk != null)
00083                 chunk.update = true;
00084
00085             chunks.TryGetValue(new ChunkWorldPos(x + 16, y, z), out chunk);
00086             if (chunk != null)
00087                 chunk.update = true;
00088             /* the chunk will then make its meshes
00089         }
00090
00097         public void DestroyChunk(int x, int y, int z)
00098         {
00099             /* if the chunk exists destroy it
00100             if (chunks.TryGetValue(new ChunkWorldPos(x, y, z), out
00101             Chunk chunk))
00102             {
00103                 /* saves the chunk before destroying it incase any block were changed in it
00104                 Serialization.Serialization.SaveChunk(chunk);
00105                 Destroy(chunk.gameObject);
00106                 chunks.Remove(new ChunkWorldPos(x, y, z));
00107             }
00108         #endregion
00109
00110         #region Block
00111         public void SetBlock(int x, int y, int z, Block block, bool saveChunk = false)
00112         {
00120             /* gets the chunk for the block to be placed in
00121             Chunk chunk = GetChunk(x, y, z);
00122
00123             /* if the chunk is not null and the block trying to be replaced is replaceable, replace it
00124             if(chunk != null && chunk.blocks[x - chunk.chunkWorldPos.
00125             x, y - chunk.chunkWorldPos.y, z - chunk.chunkWorldPos.
00126             z].breakable)
00127             {

```

```

00126
00127     chunk.SetBlock(x - chunk.chunkWorldPos.x, y - chunk.
00128     chunkWorldPos.y, z - chunk.chunkWorldPos.z, block);
00129     chunk.update = true;
00130
00131         /*updates the nebouring chunks as when a block is broken it may be in the edje of the
00132     chunk so their meshes also need to be updated
00133         /*only updates chunks that need to be updated as not every chunk will need to be and
00134     sometimes none of them will need to be
00135
00136             /*checks if the block chaged is in the edge if the x value for the chunk
00137             UpdateIfEqual(x - chunk.chunkWorldPos.x, 0, new
00138     ChunkWorldPos(x - 1, y, z));
00139             UpdateIfEqual(x - chunk.chunkWorldPos.x, Chunk.
00140     chunkSize - 1, new ChunkWorldPos(x + 1, y, z));
00141
00142             /*checks if the block chaged is in the edge if the y value for the chunk
00143             UpdateIfEqual(y - chunk.chunkWorldPos.y, 0, new
00144     ChunkWorldPos(x, y - 1, z));
00145             UpdateIfEqual(y - chunk.chunkWorldPos.y, Chunk.
00146     chunkSize - 1, new ChunkWorldPos(x, y + 1, z));
00147
00148             /*checks if the block chaged is in the edge if the z value for the chunk
00149             UpdateIfEqual(z - chunk.chunkWorldPos.z, 0, new
00150     ChunkWorldPos(x, y, z - 1));
00151             UpdateIfEqual(z - chunk.chunkWorldPos.z, Chunk.
00152     chunkSize - 1, new ChunkWorldPos(x, y, z + 1));
00153
00154         if (saveChunk)
00155             Serialization.Serialization.SaveChunk(chunk);
00156     }
00157 }
00158 #endregion
00159 #endregion
00160
00161 #region Get Things
00162 public Chunk GetChunk(int x, int y, int z)
00163 {
00164     float multiple = Chunk.chunkSize;
00165     /* rounds the given x, y, z to a multiple of 16 as chunks are 16x16x16 in size
00166     ChunkWorldPos pos = new ChunkWorldPos()
00167     {
00168         x = Mathf.FloorToInt(x / multiple) * Chunk.chunkSize,
00169         y = Mathf.FloorToInt(y / multiple) * Chunk.chunkSize,
00170         z = Mathf.FloorToInt(z / multiple) * Chunk.chunkSize
00171     };
00172
00173     /* gets the chunk if it exists
00174     chunks.TryGetValue(pos, out Chunk chunk);
00175     /* if the chunk does not exist will return null
00176     return chunk;
00177 }
00178
00179 public Block GetBlock(int x, int y, int z)
00180 {
00181     /* gets the chunk that the block is in
00182     Chunk chunk = GetChunk(x, y, z);
00183
00184     if(chunk != null)
00185     {
00186         /* gets the block in the chunk
00187         return chunk.GetBlock(x - chunk.chunkWorldPos.
00188             x, y - chunk.chunkWorldPos.y, z - chunk.chunkWorldPos.
00189             z) ?? new Air();
00190     }
00191
00192     /* returns an empty block is the chunk was not found
00193     return new Air();
00194 }
00195
00196 #endregion
00197
00198 void UpdateIfEqual(int value1, int value2, ChunkWorldPos pos)
00199 {
00200     if(value1 == value2)
00201     {
00202         Chunk chunk = GetChunk(pos.x, pos.y, pos.z);
00203
00204         if (chunk != null)
00205             chunk.update = true;
00206     }
00207 }
00208
00209 }
```

7.139 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/test.cs File Reference

Classes

- class BeeGame.Test

Namespaces

- namespace BeeGame

7.140 test.cs

```
00001 using UnityEngine;
00002 using BeeGame.Core.Dictionaries;
00003 using BeeGame.Items;
00004 using BeeGame.Blocks;
00005 using BeeGame.Core;
00006
00007 namespace BeeGame
00008 {
00009     public class Test : MonoBehaviour
00010     {
00011         private void Start()
00012         {
00013             CraftingRecipes.AddShapedRecipie(new object[] { "    ", " X ",
00014             "    ", "X", Dirt.ID }, new Grass());
00015             CraftingRecipes.AddShaplessRecipie(new object[] { new
00016             Grass(), 1 }, new Dirt());
00017         }
00018         private void Update()
00019         {
00020             //var temp = Quest.Quests.ReturnCompleatedQuests();
00021             //if(temp.Count > 0)
00022             //{
00023                 // foreach (var item in temp)
00024                 //{
00025                     // Quest.Quests.ClaimQuest(item.Key);
00026                 //}
00027             //}
00028         }
00029     }
00030 }
```

Index

AddItem
 BeeGame::Inventory::ItemsInInventory, 176, 177

AddItemToInventory
 BeeGame::Inventory::Inventory, 143

AddItemToSlots
 BeeGame::Inventory::BlockInventory::Crafting←
 TableInventory, 120

 BeeGame::Inventory::Inventory, 144

AddQuadTriangles
 BeeGame::Terrain::Chunks::MeshData, 189

AddQuest
 BeeGame::Quest::Quests, 226

AddQuests
 BeeGame::Inventory::QuestBookInventory, 219

AddShapedRecipie
 BeeGame::Core::Dictionaries::CraftingRecipies,
 106

AddShaplessRecipie
 BeeGame::Core::Dictionaries::CraftingRecipies,
 107

AddToFloatingItem
 BeeGame::Inventory::InventorySlot, 150

AddToSlot
 BeeGame::Inventory::InventorySlot, 151

AddTriangle
 BeeGame::Terrain::Chunks::MeshData, 190

AddVertices
 BeeGame::Terrain::Chunks::MeshData, 190

Air
 BeeGame::Blocks::Air, 24

Apiary
 BeeGame::Blocks::Apiary, 28

applyCollisionMesh
 BeeGame::Terrain::Chunks::Chunk, 97

ApplyCollisionMeshToNearbyChunks
 BeeGame::Terrain::Chunks::LoadChunks, 182

ApplyPlayerItems
 BeeGame::Inventory::BeeAlyzerInventory, 63

 BeeGame::Inventory::ChestInventory, 87

Awake
 BeeGame::Inventory::Player_Inventory::Player←
 Inventory, 199

 BeeGame::LoadResources, 187

 BeeGame::Player::PlayerMove, 208

 BeeGame::Player::Selector, 238

Bedrock
 BeeGame::Blocks::Bedrock, 46

Bee
 BeeGame::Items::Bee, 50, 51

BeeAlyzer
 BeeGame::Items::BeeAlyzer, 58

beeColour
 BeeGame::Core::Dictionaries::BeeDictionaries, 72

beeCombinationWeights
 BeeGame::Core::Dictionaries::BeeDictionaries, 72

beeCombinations
 BeeGame::Core::Dictionaries::BeeDictionaries, 72

beeCraftedEvent
 BeeGame::Quest::QuestEvents, 224

BeeEffect
 BeeGame::Core::Enums, 14

BeeGame, 11

BeeGame.Blocks, 12

BeeGame.Blocks.Air, 23

BeeGame.Blocks.Apiary, 27

BeeGame.Blocks.Bedrock, 46

BeeGame.Blocks.Block, 74

BeeGame.Blocks.Chest, 81

BeeGame.Blocks.CraftingTable, 112

BeeGame.Blocks.Dirt, 124

BeeGame.Blocks.Grass, 130

BeeGame.Blocks.Leaves, 178

BeeGame.Blocks.Planks, 195

BeeGame.Blocks.Wood, 313

BeeGame.Core, 12

BeeGame.Core.Dictionaries, 13

BeeGame.Core.Dictionaries.BeeCombinationDictionary←
 EqualityComparer, 67

BeeGame.Core.Dictionaries.BeeDictionaries, 69

BeeGame.Core.Dictionaries.CraftingRecipies, 105

BeeGame.Core.Dictionaries.PrefabDictionary, 211

BeeGame.Core.Dictionaries.SpriteDictionary, 260

BeeGame.Core.Enums, 13

BeeGame.Core.Extensions, 127

BeeGame.Core.THInput, 280

BeeGame.Core.THVector2, 291

BeeGame.Core.THVector3, 298

BeeGame.Core.UnityTypeReplacements, 18

BeeGame.Core.UnityTypeReplacements.THQuaternion,
 285

BeeGame.Exceptions, 18

BeeGame.Exceptions.CraftingRecipeAdditionException,
 104

BeeGame.Exceptions.InputException, 140

BeeGame.Exceptions.QuestAlreadyExistsException,
 214

BeeGame.Inventory, 19

BeeGame.Inventory.ApiaryCraftingOutputSlot, 39

BeeGame.Inventory.ApiaryInventory, 40

BeeGame.Inventory.BeeAlyzerInventory, 62

BeeGame.Inventory.BlockInventory, 19

BeeGame.Inventory.BlockInventory.CraftingTable←
 Inventory, 119

BeeGame.Inventory.ChestInventory, 86

BeeGame.Inventory.CraftingOutputSlot, 102

BeeGame.Inventory.Inventory, 142

BeeGame.Inventory.InventorySlot, 149

BeeGame.Inventory.ItemsInInventory, 175

BeeGame.Inventory.Player_Inventory, 19

BeeGame.Inventory.Player_Inventory.PlayerInventory, 198
 BeeGame.Inventory.QuestBookInventory, 218
 BeeGame.Items, 20
 BeeGame.Items.AbstractItem, 22
 BeeGame.Items.ApplyColour, 44
 BeeGame.Items.Bee, 49
 BeeGame.Items.BeeAlyzer, 57
 BeeGame.Items.Honey, 134
 BeeGame.Items.HoneyComb, 136
 BeeGame.Items.Item, 158
 BeeGame.Items.ItemGameObject, 173
 BeeGame.Items.NormalBee, 192
 BeeGame.Items.QueenBee, 212
 BeeGame.Items.QuestBook, 215
 BeeGame.Items.SetBeeGOColours, 250
 BeeGame.Items.Tile, 312
 BeeGame.LoadResources, 186
 BeeGame.Player, 20
 BeeGame.Player.PlayerLook, 204
 BeeGame.Player.PlayerMove, 207
 BeeGame.Player.SavePlayerPosition, 236
 BeeGame.Player.Selector, 237
 BeeGame.Quest, 20
 BeeGame.Quest.QuestEvents, 222
 BeeGame.Quest.Quests, 225
 BeeGame.Resources, 20
 BeeGame.Resources.Resources, 231
 BeeGame.Serialization, 21
 BeeGame.Serialization.Serialization, 242
 BeeGame.SpawnItem, 259
 BeeGame.Terrain, 21
 BeeGame.Terrain.ChunkWorldPos, 99
 BeeGame.Terrain.Chunks, 21
 BeeGame.Terrain.Chunks.Chunk, 91
 BeeGame.Terrain.Chunks.LoadChunks, 181
 BeeGame.Terrain.Chunks.MeshData, 188
 BeeGame.Terrain.Chunks.SaveChunk, 234
 BeeGame.Terrain.LandGeneration, 21
 BeeGame.Terrain.LandGeneration.Noise, 22
 BeeGame.Terrain.LandGeneration.Noise.SimplexNoise, 252
 BeeGame.Terrain.LandGeneration.Terrain, 262
 BeeGame.Terrain.LandGeneration.TerrainGeneration, 270
 BeeGame.Terrain.LandGeneration.World, 316
 BeeGame.Test, 279
 BeeGame::Blocks::Air
 Air, 24
 BlockData, 24
 BreakBlock, 25
 GetHashCode, 25
 ID, 27
 IsSolid, 26
 ToString, 26
 BeeGame::Blocks::Apiary
 Apiary, 28
 BlockData, 29
 BreakBlock, 29
 CombineEffect, 30
 CombineFertility, 30
 CombineLifespan, 31
 CombineProductionSpeed, 31
 CombineSpecies, 32
 GetGameObject, 32
 GetHashCode, 33
 GetItemSprite, 33
 ID, 38
 InteractWithBlock, 33
 MakeBee, 34
 MakeBees, 35
 mutationMultipler, 38
 myGameobject, 38
 Rand, 36
 ReturnChange, 36
 TexturePosition, 37
 ToString, 38
 BeeGame::Blocks::Bedrock
 Bedrock, 46
 BreakBlock, 47
 GetHashCode, 47
 ID, 48
 TexturePosition, 47
 ToString, 48
 BeeGame::Blocks::Block
 Block, 75
 BlockData, 76
 BreakBlock, 77
 breakable, 80
 changed, 80
 GetHashCode, 77
 GetItemSprite, 78
 ID, 80
 InteractWithBlock, 78
 IsSolid, 78
 placeable, 80
 ToString, 79
 UpdateBlock, 79
 BeeGame::Blocks::Chest
 BlockData, 82
 BreakBlock, 83
 Chest, 82
 GetGameObject, 83
 GetHashCode, 84
 GetItemSprite, 84
 ID, 86
 InteractWithBlock, 84
 myGameobject, 86
 TexturePosition, 85
 ToString, 85
 BeeGame::Blocks::CraftingTable
 BlockData, 113
 BreakBlock, 114
 CraftingTable, 113
 GetGameObject, 114
 GetHashCode, 114

GetItemSprite, 115
ID, 118
InteractWithBlock, 115
IsSolid, 116
myGameObject, 118
ReturnShapedRecipieItem, 116, 117
ReturnShapelessRecipieItem, 117
TexturePosition, 117
BeeGame::Blocks::Dirt
Dirt, 125
GetHashCode, 125
GetItemSprite, 126
ID, 127
TexturePosition, 126
ToString, 127
BeeGame::Blocks::Grass
GetHashCode, 132
GetItemSprite, 132
Grass, 131
ID, 134
TexturePosition, 132
ToString, 133
UpdateBlock, 133
BeeGame::Blocks::Leaves
GetHashCode, 179
GetItemSprite, 179
ID, 181
IsSolid, 179
Leaves, 178
TexturePosition, 180
ToString, 180
BeeGame::Blocks::Planks
GetHashCode, 196
GetItemSprite, 197
ID, 198
Planks, 196
TexturePosition, 197
ToString, 198
BeeGame::Blocks::Wood
GetHashCode, 314
GetItemSprite, 314
ID, 315
TexturePosition, 314
ToString, 315
Wood, 314
BeeGame::Core::Dictionaries::BeeCombination←
DictionaryEqualityComparer
Equals, 68
GetHashCode, 68
BeeGame::Core::Dictionaries::BeeDictionaries
beeColour, 72
beeCombinationWeights, 72
beeCombinations, 72
beeProduce, 73
CombColour, 69
GetBeeColour, 70
GetBeeProduce, 70
GetCombColour, 70
GetCombinations, 71
GetWeights, 71
honeyCoumbColour, 73
BeeGame::Core::Dictionaries::CraftingRecipies
AddShapedRecipie, 106
AddShapelessRecipie, 107
GetShapedRecipieItem, 108
GetShapelessRecipieResult, 109, 110
GetShapelessRecipieString, 110
shapedCraftingRecipies, 111
shapelessRecipies, 111
BeeGame::Core::Dictionaries::PrefabDictionary
GetPrefab, 211
LoadPrefabs, 212
prefabDictionary, 212
BeeGame::Core::Dictionaries::SpriteDictionary
GetSprite, 261
itemSpriteDictionary, 262
LoadSprites, 261
BeeGame::Core::Enums
BeeEffect, 14
BeeHumidityPreference, 14
BeeLifeSpan, 15
BeeProductionSpeed, 15
BeeSpecies, 15
BeeTempPreference, 17
BeeType, 17
Direction, 17
HoneyCombType, 18
BeeGame::Core::Extensions
CloneObject< T >, 128
ColourSprite, 129
SpawnItem, 130
BeeGame::Core::THInput
blockInventoryJustClosed, 284
chestOpen, 284
GetAxis, 281
GetButton, 282
GetButtonDown, 283
GetButtonUp, 283
inputButtons, 284
isAnotherInventoryOpen, 285
BeeGame::Core::THVector2
Equals, 293
GetHashCode, 293
operator THVector2, 293
operator Vector2, 294
operator!=, 294
operator*, 294, 295
operator+, 295
operator-, 296
operator/, 296, 297
operator==, 297
THVector2, 292
ToString, 297
x, 298
y, 298
BeeGame::Core::THVector3

Distance, 301
 Equals, 302
 GetHashCode, 302
 operator Quaternion, 302
 operator THVector3, 303
 operator Vector3, 303
 operator!=, 304
 operator*, 304, 305
 operator+, 306, 307
 operator-, 307, 308
 operator/, 309, 310
 operator==, 310
 THVector3, 300, 301
 ToString, 311
 x, 311
 y, 311
 z, 311

BeeGame::Core::UnityTypeReplacements::THQuaternion
 identity, 290
 operator Quaternion, 287
 operator THQuaternion, 287
 operator*, 287
 operator+, 288
 operator-, 288, 289
 operator/, 289
 THQuaternion, 286
 w, 290
 x, 290
 y, 290
 z, 290

BeeGame::Exceptions::CraftingRecipeAddition
 Exception
 CraftingRecipeAdditionException, 104

BeeGame::Exceptions::InputException
 InputException, 141

BeeGame::Exceptions::QuestAlreadyExistsException
 QuestAlreadyExistsException, 214, 215

BeeGame::Inventory::ApiaryCraftingOutputSlot
 OnPointerClick, 39
 Update, 40

BeeGame::Inventory::ApiaryInventory
 beesCombineing, 43
 CheckforBees, 41
 combinationTime, 43
 FixedUpdate, 42
 SetChestInventory, 42
 timerSlider, 44
 Update, 43

BeeGame::Inventory::BeeAlyzerInventory
 ApplyPlayerItems, 63
 CheckForBeeAndHoney, 63
 DropItemsFromInventory, 64
 infoText, 66
 myItem, 67
 playerInventory, 67
 ReturnData, 64
 SaveInv, 65
 SetPlayerItems, 65

BeeGame::Inventory::BlockInventory::CraftingTable
 Inventory
 AddItemToSlots, 120
 CheckShapedRecipie, 120
 CheckShapelessRecipie, 120
 CraftedItemRemoved, 121
 craftingResultRemoved, 124
 DropItemsFromInventory, 121
 ItemRemovedFromResult, 121
 OnDestroy, 122
 SaveInv, 122
 SetChestInventory, 122
 Start, 123
 ToggleInventory, 123
 Update, 123

BeeGame::Inventory::ChestInventory
 ApplyPlayerItems, 87
 inventory, 90
 inventoryPosition, 90
 inventorySize, 90
 playerinventory, 90
 SetChestInventory, 87
 SetPlayerItems, 88
 ToggleInventory, 88
 Update, 89
 UpdateChestInventory, 89

BeeGame::Inventory::CraftingOutputSlot
 OnPointerClick, 103
 Update, 103

BeeGame::Inventory::Inventory
 AddItemToInventory, 143
 AddItemToSlots, 144
 DrawItemAtCursor, 144
 floatingItem, 147
 GetAllItems, 144
 inventoryName, 148
 InventorySet, 145
 items, 148
 myblock, 148
 PutItemsInSlots, 145
 SaveInv, 145
 SetAllItems, 146
 SetInventorySize, 146
 SetItemInventory, 147
 slots, 148
 spriteAtCursor, 148
 thisInventoryOpen, 149
 ToggleInventory, 147
 UpdateBase, 147

BeeGame::Inventory::InventorySlot
 AddToFloatingItem, 150
 AddToSlot, 151
 CheckFloatingItem, 152
 CheckItem, 152
 item, 156
 itemText, 157

itemsCanBeInserted, 156
myInventory, 157
OnDisable, 152
OnPointerClick, 152
OnPointerEnter, 154
OnPointerExit, 154
selectedSlot, 157
slotIndex, 157
SplitStack, 155
SwapItems, 155
Update, 155
UpdateIcon, 156
BeeGame::Inventory::ItemsInInventory
AddItem, 176, 177
ItemsInInventory, 176
itemsInInventory, 177
BeeGame::Inventory::Player_Inventory::PlayerInventory
Awake, 199
GetItemFromHotBar, 200
OpenPlayerInventory, 200
PickupItem, 201
playerInventory, 203
RemoveItemFromInventory, 201
SelectedSlot, 202
SetPlayerInventory, 202
Update, 202
BeeGame::Inventory::QuestBookInventory
AddQuests, 219
myItem, 222
QuestAchieved, 220
questButton, 222
scrollObject, 222
Start, 220
ToggleInventory, 221
Update, 221
BeeGame::Items::AbstractItem
GetHashCode, 22
GetItemID, 23
GetItemName, 23
BeeGame::Items::ApplyColour
colour, 45
objects, 45
Start, 45
BeeGame::Items::Bee
Bee, 50, 51
beeType, 56
canSeeBeeData, 55
ConvertToQueen, 51, 52
GetGameObject, 52
GetHashCode, 52
GetItemID, 53
GetItemSprite, 53
ID, 55
itemSprite, 55
MakeBeeWithStats, 54
maxStack, 56
maxStackCount, 56
normalBee, 56
previousBeeType, 56
queenBee, 57
BeeGame::Items::BeeAlyzer
BeeAlyzer, 58
GetHashCode, 59
GetItemID, 59
GetItemSprite, 59
ID, 61
InteractWithItem, 59
InteractWithObject, 60
maxStackCount, 61
myInventory, 61
OpenItemInventory, 60
placeable, 61
BeeGame::Items::Honey
GetHashCode, 135
GetItemID, 135
ID, 136
ToString, 135
BeeGame::Items::HoneyComb
CombColour, 140
GetGameObject, 138
GetHashCode, 138
GetItemID, 138
GetItemSprite, 139
HoneyComb, 137
ID, 139
itemSprite, 139
type, 140
BeeGame::Items::Item
Clone, 160
Equals, 161
FaceDataDown, 161
FaceDataEast, 162
FaceDataNorth, 163
FaceDataSouth, 164
FaceDataUp, 165
FaceDataWest, 165
FaceUVs, 166
GetGameObject, 167
GetHashCode, 167
GetItemID, 167
GetItemName, 168
GetItemSprite, 168
ID, 172
InteractWithItem, 168
InteractWithObject, 169
Item, 160
ItemMesh, 169
itemName, 172
itemStackCount, 172
maxStackCount, 172
operator!=, 170
operator==, 170
placeable, 172
TexturePosition, 171
tileSize, 172
ToString, 171

usesGameObject, 173

BeeGame::Items::ItemGameObject

- go, 175
- item, 175
- MakeMesh, 174
- Start, 174
- Update, 174

BeeGame::Items::NormalBee

- GetHashCode, 193
- pEffect, 193
- pFertility, 193
- pLifespan, 193
- pProdSpeed, 194
- pSpecies, 194
- sEffect, 194
- sFertility, 194
- sLifespan, 194
- sProdSpeed, 195
- sSpecies, 195

BeeGame::Items::QueenBee

- drone, 214
- GetHashCode, 213
- queen, 214
- QueenBee, 213

BeeGame::Items::QuestBook

- GetHashCode, 216
- GetItemSprite, 217
- ID, 218
- maxStackCount, 218
- OpenItemInventory, 217
- placeable, 218
- QuestBook, 216

BeeGame::Items::SetBeeGOColours

- beeType, 251
- colour, 251
- crown, 251
- objects, 252
- Start, 251
- tiara, 252

BeeGame::Items::Tile

- x, 312
- y, 312

BeeGame::LoadResources

- Awake, 187
- delay, 188
- Update, 187

BeeGame::Player::PlayerLook

- cameraTransform, 205
- Look, 204
- myTransform, 206
- rotationLock, 206
- speed, 206
- Start, 205
- Update, 205
- xRot, 206
- yRot, 206

BeeGame::Player::PlayerMove

- Awake, 208

canJump, 210

FixedUpdate, 208

gravity, 210

jumpHeight, 210

maxVelocity, 210

MovePlayer, 208

myRigidBody, 210

OnCollisionStay, 209

speed, 211

VerticalJumpSpeed, 209

BeeGame::Player::SavePlayerPosition

- counter, 237
- Update, 236

BeeGame::Player::Selector

- Awake, 238
- BreakBlock, 238
- FixedUpdate, 239
- hit, 241
- layers, 241
- PlaceBlock, 239
- playerInventory, 241
- selectedHotbarSlot, 241
- SelectedSlot, 240
- selector, 242
- Update, 240
- UpdateSelector, 240

BeeGame::Quest::QuestEvents

- beeCraftedEvent, 224
- BeeQuestEventHandler, 223
- CallBeeCraftedEvent, 223
- CallItemCraftedEvent, 223
- CallItemPickupEvent, 223
- CallPureBeeCraftedEvent, 224
- itemCraftedEvent, 224
- itemPickupEvent, 225
- pureBeeCraftedEvent, 225
- PureBeeQuestEventHandler, 224
- QuestEventHandler, 224

BeeGame::Quest::Quests

- AddQuest, 226
- ClaimQuest, 226
- compleatedQuests, 229
- compleatedUnclaimedQuests, 230
- currentQuests, 230
- LoadQuests, 226
- lockedQuests, 230
- ReturnBeeCraftingQuest, 227
- ReturnCompleatedClaimedQuests, 227
- ReturnCompleatedQuests, 227
- ReturnCraftingTableQuest, 227
- ReturnCurrentQuests, 228
- ReturnLockedQuests, 228
- ReturnPickupQuest, 228
- ReturnPureBreadBeeCraftingQuest, 228
- UnlockQuests, 229

BeeGame::Resources::Resources

- Culture, 234
- GetPrefabs, 232

GetSprites, 232
Prefabs, 234
resourceCulture, 233
resourceMan, 233
ResourceManager, 234
Resources, 231
Sprites, 234
BeeGame::Serialization::Serialization
DeSerializeInventory, 244
DeleteFile, 243
FileName, 244
LoadChunk, 245
LoadFile, 245
LoadPlayerPosition, 246
LoadQuests, 246
MakeDirectorys, 247
SaveChunk, 247
SaveFile, 248
saveFolderName, 249
savePath, 250
SavePlayerPosition, 248
SaveQuests, 249
SerializeInventory, 249
worldName, 250
BeeGame::SpawnItem
OnDrawGizmos, 259
Start, 259
BeeGame::Terrain::ChunkWorldPos
ChunkWorldPos, 99
Equals, 100
GetHashCode, 100
operator ChunkWorldPos, 100
operator THVector3, 101
ToString, 101
x, 102
y, 102
z, 102
BeeGame::Terrain::Chunks::Chunk
applyCollisionMesh, 97
blocks, 97
chunkSize, 97
chunkWorldPos, 97
ColliderMesh, 92
filter, 97
GetBlock, 93
InRange, 93
mesh, 97
meshCollider, 98
RenderMesh, 94
rendered, 98
SetBlock, 94
SetBlocksUnmodified, 95
Start, 95
Update, 96
update, 98
UpdateChunk, 96
updateCollisionMesh, 98
world, 98
BeeGame::Terrain::Chunks::LoadChunks
ApplyCollisionMeshToNearbyChunks, 182
BuildChunk, 182
buildList, 185
chunkPositions, 185
DeleteChunks, 183
FindChunksToLoad, 183
LoadAndRenderChunks, 184
nearbyChunks, 185
Start, 184
timer, 186
Update, 185
world, 186
BeeGame::Terrain::Chunks::MeshData
AddQuadTriangles, 189
AddTriangle, 190
AddVertices, 190
colTris, 191
colVerts, 191
done, 191
shareMeshes, 191
tris, 191
uv, 192
verts, 192
BeeGame::Terrain::Chunks::SaveChunk
blocks, 235
SaveChunk, 235
BeeGame::Terrain::LandGeneration::Noise::Simplex
Noise
FastFloor, 253
Generate, 253–255
grad, 256, 257
Mod, 257
perm, 258
BeeGame::Terrain::LandGeneration::Terrain
BlockInPosition, 263
GetBlock, 263, 264
GetBlockPos, 264, 265
GetBlockPosFromRayCast, 266
GetChunk, 266
Round, 266
RoundXZ, 267
RoundY, 268
SetBlock, 269
world, 270
BeeGame::Terrain::LandGeneration::TerrainGeneration
caveFrequency, 277
caveSize, 277
ChunkGen, 272
ChunkGenThread, 273
CreateTree, 273
dirtBaseHeight, 277
dirtNoise, 277
dirtNoiseHeight, 277
GenChunkColum, 274
GetNoise, 275
SetBlock, 276
stoneBaseHeight, 277

stoneBaseNoise, 278
 stoneBaseNoiseHeight, 278
 stoneMinHeight, 278
 stoneMountainFrequency, 278
 stoneMountainHeight, 278
 treeDensity, 279
 treeFrequency, 279
 BeeGame::Terrain::LandGeneration::World
 chunkHasMadeCollisionMesh, 321
 chunkPrefab, 321
 chunks, 321
 CreateChunk, 316
 DestroyChunk, 317
 GetBlock, 318
 GetChunk, 318
 SetBlock, 319
 UpdateIfEqual, 320
 BeeGame::Test
 Start, 280
 Update, 280
 BeeHumidityPreference
 BeeGame::Core::Enums, 14
 BeeLifeSpan
 BeeGame::Core::Enums, 15
 beeProduce
 BeeGame::Core::Dictionaries::BeeDictionaries, 73
 BeeProductionSpeed
 BeeGame::Core::Enums, 15
 BeeQuestEventHandler
 BeeGame::Quest::QuestEvents, 223
 BeeSpecies
 BeeGame::Core::Enums, 15
 BeeTempPreference
 BeeGame::Core::Enums, 17
 BeeType
 BeeGame::Core::Enums, 17
 beeType
 BeeGame::Items::Bee, 56
 BeeGame::Items::SetBeeGOColours, 251
 beesCombineing
 BeeGame::Inventory::ApiaryInventory, 43
 Block
 BeeGame::Blocks::Block, 75
 BlockData
 BeeGame::Blocks::Air, 24
 BeeGame::Blocks::Apiary, 29
 BeeGame::Blocks::Block, 76
 BeeGame::Blocks::Chest, 82
 BeeGame::Blocks::CraftingTable, 113
 BlockInPosition
 BeeGame::Terrain::LandGeneration::Terrain, 263
 blockInventoryJustClosed
 BeeGame::Core::THInput, 284
 blocks
 BeeGame::Terrain::Chunks::Chunk, 97
 BeeGame::Terrain::Chunks::SaveChunk, 235
 BreakBlock
 BeeGame::Blocks::Air, 25
 BeeGame::Blocks::Apiary, 29
 BeeGame::Blocks::Bedrock, 47
 BeeGame::Blocks::Block, 77
 BeeGame::Blocks::Chest, 83
 BeeGame::Blocks::CraftingTable, 114
 BeeGame::Player::Selector, 238
 breakable
 BeeGame::Blocks::Block, 80
 BuildChunk
 BeeGame::Terrain::Chunks::LoadChunks, 182
 buildList
 BeeGame::Terrain::Chunks::LoadChunks, 185
 C:/Users/Toothless/Documents/GitHub/BeeGame/←
 Code/BeeGame/BeeGame/Blocks/Air.cs,
 321, 322
 C:/Users/Toothless/Documents/GitHub/BeeGame/←
 Code/BeeGame/BeeGame/Blocks/Apiary.cs,
 322
 C:/Users/Toothless/Documents/GitHub/BeeGame/←
 Code/BeeGame/BeeGame/Blocks/Bedrock.cs,
 326
 C:/Users/Toothless/Documents/GitHub/BeeGame/←
 Code/BeeGame/BeeGame/Blocks/Block.cs,
 327
 C:/Users/Toothless/Documents/GitHub/BeeGame/←
 Code/BeeGame/BeeGame/Blocks/Chest.cs,
 328, 329
 C:/Users/Toothless/Documents/GitHub/BeeGame/←
 Code/BeeGame/BeeGame/Blocks/Crafting←
 Table.cs, 330
 C:/Users/Toothless/Documents/GitHub/BeeGame/←
 Code/BeeGame/BeeGame/Blocks/Dirt.cs,
 331, 332
 C:/Users/Toothless/Documents/GitHub/BeeGame/←
 Code/BeeGame/BeeGame/Blocks/Grass.cs,
 332, 333
 C:/Users/Toothless/Documents/GitHub/BeeGame/←
 Code/BeeGame/BeeGame/Blocks/Leaves.cs,
 333, 334
 C:/Users/Toothless/Documents/GitHub/BeeGame/←
 Code/BeeGame/BeeGame/Blocks/Planks.cs,
 334, 335
 C:/Users/Toothless/Documents/GitHub/BeeGame/←
 Code/BeeGame/BeeGame/Blocks/Wood.cs,
 335
 C:/Users/Toothless/Documents/GitHub/BeeGame/←
 Code/BeeGame/BeeGame/Core/Dictionarys/←
 BeeDictionaries.cs, 336
 C:/Users/Toothless/Documents/GitHub/BeeGame/←
 Code/BeeGame/BeeGame/Core/Dictionarys/←
 CraftingRecipies.cs, 338
 C:/Users/Toothless/Documents/GitHub/BeeGame/←
 Code/BeeGame/BeeGame/Core/Dictionarys/←
 EqualityComperors.cs, 340
 C:/Users/Toothless/Documents/GitHub/BeeGame/←
 Code/BeeGame/BeeGame/Core/Dictionarys/←
 PrefabDictionary.cs, 341

C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Core/Dictionarys/←
SpriteDictionary.cs, 341, 342
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Core/Enums/←
Enums.cs, 342, 343
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Core/Extensions.cs, 344
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Core/UnityType Replacements/THInput.cs, 346
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Core/UnityType Replacements/THQuaternion.cs, 348
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Core/UnityType Replacements/THVector2.cs, 350
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Core/UnityType Replacements/THVector3.cs, 352
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Exceptions/CraftingRecipeAdditionException.cs, 354, 355
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Exceptions/InputException.cs, 355, 356
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Exceptions/QuestAlreadyExistsException.cs, 356
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Inventory/ApiaryCraftingOutputSlot.cs, 357
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Inventory/BlockInventory/ApiaryInventory.cs, 357, 358
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Inventory/BlockInventory/ChestInventory.cs, 359
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Inventory/BlockInventory/CraftingTableInventory.cs, 360, 361
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Inventory/CraftingOutputSlot.cs, 362, 363
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Inventory/Inventory.cs, 363
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Inventory/InventorySlot.cs, 365
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Inventory/ItemInventory/BeeAlyzerInventory.cs, 368, 369
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Inventory/ItemInventory/QuestBookInventory.cs, 370, 371
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Inventory/ItemsInInventory.cs, 372
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Inventory/PlayerInventory/PlayerInventory.cs, 373
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Items/AbstractItem.cs, 375
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Items/ApplyColour.cs, 376
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Items/Bee.cs, 376, 377
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Items/BeeAlyzer.cs, 379, 380
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Items/GameObjectStuff/ItemGameObject.cs, 381
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Items/GameObjectStuff/SetBeeGOColours.cs, 382
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Items/Honey.cs, 383
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Items/HoneyComb.cs, 383, 384
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Items/Item.cs, 384, 385
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Items/QuestBook.cs, 388, 389
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/LoadResources.cs, 389, 390
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Player/PlayerLook.cs, 391
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Player/PlayerMove.cs, 391, 392
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Player/SavePlayerPosition.cs, 393
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Player/Selector.cs, 393, 394
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Properties/AssemblyInfo.cs, 395
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Quest/QuestEvents.cs, 396
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Quest/Quests.cs,

397
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Resources/←
Resources.Designer.cs, 399
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Serialization/←
Serialization.cs, 401
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/SpawnItem.cs,
404
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Terrain/Chunk←
WorldPos.cs, 413
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Terrain/Chunks/←
Chunk.cs, 405
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Terrain/Chunks/←
LoadChunks.cs, 408
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Terrain/Chunks/←
MeshData.cs, 411, 412
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Terrain/Chunks/←
SaveChunk.cs, 412, 413
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Terrain/Land←
Generation/Noise/SimplexNoise.cs, 414
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Terrain/Land←
Generation/Terrain.cs, 418
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Terrain/Land←
Generation/TerrainGeneration.cs, 421
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/Terrain/Land←
Generation/World.cs, 423
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/obj/Debug/←
TemporaryGeneratedFile_036C0B5B-1481-
4323-8D20-8F5ADCB23D92.cs, 390
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/obj/Debug/←
TemporaryGeneratedFile_5937a670-0e60-
4077-877b-f7221da3dda1.cs, 390
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/obj/Debug/←
TemporaryGeneratedFile_E7A71F73-0F8D-
4B9B-B56E-8E70B10BC5D3.cs, 390
C:/Users/Toothless/Documents/GitHub/BeeGame/←
Code/BeeGame/BeeGame/test.cs, 426
CallBeeCraftedEvent
BeeGame::Quest::QuestEvents, 223
CallItemCraftedEvent
BeeGame::Quest::QuestEvents, 223
CallItemPickupEvent
BeeGame::Quest::QuestEvents, 223
CallPureBeeCraftedEvent

BeeGame::QuestEvents, 224
cameraTransform
BeeGame::Player::PlayerLook, 205
canJump
BeeGame::Player::PlayerMove, 210
canSeeBeeData
BeeGame::Items::Bee, 55
caveFrequency
BeeGame::Terrain::LandGeneration::Terrain←
Generation, 277
caveSize
BeeGame::Terrain::LandGeneration::Terrain←
Generation, 277
changed
BeeGame::Blocks::Block, 80
CheckFloatingItem
BeeGame::Inventory::InventorySlot, 152
CheckForBeeAndHoney
BeeGame::Inventory::BeeAlyzerInventory, 63
CheckItem
BeeGame::Inventory::InventorySlot, 152
CheckShapedRecipie
BeeGame::Inventory::BlockInventory::Crafting←
TableInventory, 120
CheckShapelessRecipie
BeeGame::Inventory::BlockInventory::Crafting←
TableInventory, 120
CheckforBees
BeeGame::Inventory::ApiaryInventory, 41
Chest
BeeGame::Blocks::Chest, 82
chestOpen
BeeGame::Core::THInput, 284
ChunkGen
BeeGame::Terrain::LandGeneration::Terrain←
Generation, 272
ChunkGenThread
BeeGame::Terrain::LandGeneration::Terrain←
Generation, 273
chunkHasMadeCollisionMesh
BeeGame::Terrain::LandGeneration::World, 321
chunkPositions
BeeGame::Terrain::Chunks::LoadChunks, 185
chunkPrefab
BeeGame::Terrain::LandGeneration::World, 321
chunkSize
BeeGame::Terrain::Chunks::Chunk, 97
ChunkWorldPos
BeeGame::Terrain::ChunkWorldPos, 99
chunkWorldPos
BeeGame::Terrain::Chunks::Chunk, 97
chunks
BeeGame::Terrain::LandGeneration::World, 321
ClaimQuest
BeeGame::Quest::Quests, 226
Clone
BeeGame::Items::Item, 160
CloneObject< T >

BeeGame::Core::Extensions, 128
colTris
 BeeGame::Terrain::Chunks::MeshData, 191
colVerts
 BeeGame::Terrain::Chunks::MeshData, 191
ColliderMesh
 BeeGame::Terrain::Chunks::Chunk, 92
colour
 BeeGame::Items::ApplyColour, 45
 BeeGame::Items::SetBeeGOColours, 251
ColourSprite
 BeeGame::Core::Extensions, 129
CombColour
 BeeGame::Core::Dictionaries::BeeDictionaries, 69
 BeeGame::Items::HoneyComb, 140
combinationTime
 BeeGame::Inventory::ApiaryInventory, 43
CombineEffect
 BeeGame::Blocks::Apiary, 30
CombineFertility
 BeeGame::Blocks::Apiary, 30
CombineLifespan
 BeeGame::Blocks::Apiary, 31
CombineProductionSpeed
 BeeGame::Blocks::Apiary, 31
CombineSpecies
 BeeGame::Blocks::Apiary, 32
compleatedQuests
 BeeGame::Quest::Quests, 229
compleatedUnclaimedQuests
 BeeGame::Quest::Quests, 230
ConvertToQueen
 BeeGame::Items::Bee, 51, 52
counter
 BeeGame::Player::SavePlayerPosition, 237
CraftedItemRemoved
 BeeGame::Inventory::BlockInventory::Crafting←
 TableInventory, 121
CraftingRecipeAdditionException
 BeeGame::Exceptions::CraftingRecipeAddition←
 Exception, 104
craftingResultRemoved
 BeeGame::Inventory::BlockInventory::Crafting←
 TableInventory, 124
CraftingTable
 BeeGame::Blocks::CraftingTable, 113
CreateChunk
 BeeGame::Terrain::LandGeneration::World, 316
CreateTree
 BeeGame::Terrain::LandGeneration::Terrain←
 Generation, 273
crown
 BeeGame::Items::SetBeeGOColours, 251
Culture
 BeeGame::Resources::Resources, 234
currentQuests
 BeeGame::Quest::Quests, 230
DeSerializeInventory
 BeeGame::Serialization::Serialization, 244
delay
 BeeGame::LoadResources, 188
DeleteChunks
 BeeGame::Terrain::Chunks::LoadChunks, 183
DeleteFile
 BeeGame::Serialization::Serialization, 243
DestroyChunk
 BeeGame::Terrain::LandGeneration::World, 317
Direction
 BeeGame::Core::Enums, 17
Dirt
 BeeGame::Blocks::Dirt, 125
dirtBaseHeight
 BeeGame::Terrain::LandGeneration::Terrain←
 Generation, 277
dirtNoise
 BeeGame::Terrain::LandGeneration::Terrain←
 Generation, 277
dirtNoiseHeight
 BeeGame::Terrain::LandGeneration::Terrain←
 Generation, 277
Distance
 BeeGame::Core::THVector3, 301
done
 BeeGame::Terrain::Chunks::MeshData, 191
DrawItemAtCursor
 BeeGame::Inventory::Inventory, 144
drone
 BeeGame::Items::QueenBee, 214
DropItemsFromInventory
 BeeGame::Inventory::BeeAlyzerInventory, 64
 BeeGame::Inventory::BlockInventory::Crafting←
 TableInventory, 121
Equals
 BeeGame::Core::Dictionaries::BeeCombination←
 DictionaryEqualityComparer, 68
 BeeGame::Core::THVector2, 293
 BeeGame::Core::THVector3, 302
 BeeGame::Items::Item, 161
 BeeGame::Terrain::ChunkWorldPos, 100
FaceDataDown
 BeeGame::Items::Item, 161
FaceDataEast
 BeeGame::Items::Item, 162
FaceDataNorth
 BeeGame::Items::Item, 163
FaceDataSouth
 BeeGame::Items::Item, 164
FaceDataUp
 BeeGame::Items::Item, 165
FaceDataWest
 BeeGame::Items::Item, 165
FaceUVs
 BeeGame::Items::Item, 166
FastFloor

BeeGame::Terrain::LandGeneration::Noise::SimplexNoise, 253

FileName
BeeGame::Serialization::Serialization, 244

filter
BeeGame::Terrain::Chunks::Chunk, 97

FindChunksToLoad
BeeGame::Terrain::Chunks::LoadChunks, 183

FixedUpdate
BeeGame::Inventory::ApiaryInventory, 42
BeeGame::Player::PlayerMove, 208
BeeGame::Player::Selector, 239

floatingItem
BeeGame::Inventory::Inventory, 147

GenChunkColum
BeeGame::Terrain::LandGeneration::TerrainGeneration, 274

Generate
BeeGame::Terrain::LandGeneration::Noise::SimplexNoise, 253–255

GetAllItems
BeeGame::Inventory::Inventory, 144

GetAxis
BeeGame::Core::THInput, 281

GetBeeColour
BeeGame::Core::Dictionaries::BeeDictionaries, 70

GetBeeProduce
BeeGame::Core::Dictionaries::BeeDictionaries, 70

GetBlock
BeeGame::Terrain::Chunks::Chunk, 93
BeeGame::Terrain::LandGeneration::Terrain, 263, 264
BeeGame::Terrain::LandGeneration::World, 318

GetBlockPos
BeeGame::Terrain::LandGeneration::Terrain, 264, 265

GetBlockPosFromRayCast
BeeGame::Terrain::LandGeneration::Terrain, 266

GetButton
BeeGame::Core::THInput, 282

GetButtonDown
BeeGame::Core::THInput, 283

GetButtonUp
BeeGame::Core::THInput, 283

GetChunk
BeeGame::Terrain::LandGeneration::Terrain, 266
BeeGame::Terrain::LandGeneration::World, 318

GetCombColour
BeeGame::Core::Dictionaries::BeeDictionaries, 70

GetCombinations
BeeGame::Core::Dictionaries::BeeDictionaries, 71

GetGameObject
BeeGame::Blocks::Apiary, 32
BeeGame::Blocks::Chest, 83
BeeGame::Blocks::CraftingTable, 114
BeeGame::Items::Bee, 52
BeeGame::Items::HoneyComb, 138
BeeGame::Items::Item, 167

GetHashCode
BeeGame::Blocks::Air, 25
BeeGame::Blocks::Apiary, 33
BeeGame::Blocks::Bedrock, 47
BeeGame::Blocks::Block, 77
BeeGame::Blocks::Chest, 84
BeeGame::Blocks::CraftingTable, 114
BeeGame::Blocks::Dirt, 125
BeeGame::Blocks::Grass, 132
BeeGame::Blocks::Leaves, 179
BeeGame::Blocks::Planks, 196
BeeGame::Blocks::Wood, 314
BeeGame::Core::Dictionaries::BeeCombinationDictionaryEqualityComparer, 68

BeeGame::Core::THVector2, 293

BeeGame::Core::THVector3, 302

BeeGame::Items::AbstractItem, 22

BeeGame::Items::Bee, 52

BeeGame::Items::BeeAlyzer, 59

BeeGame::Items::Honey, 135

BeeGame::Items::HoneyComb, 138

BeeGame::Items::Item, 167

BeeGame::Items::NormalBee, 193

BeeGame::Items::QueenBee, 213

BeeGame::Items::QuestBook, 216

BeeGame::Terrain::ChunkWorldPos, 100

GetItemFromHotBar
BeeGame::Inventory::Player_Inventory::PlayerInventory, 200

GetItemID
BeeGame::Items::AbstractItem, 23

BeeGame::Items::Bee, 53

BeeGame::Items::BeeAlyzer, 59

BeeGame::Items::Honey, 135

BeeGame::Items::HoneyComb, 138

BeeGame::Items::Item, 167

GetItemName
BeeGame::Items::AbstractItem, 23

BeeGame::Items::Item, 168

GetItemSprite
BeeGame::Blocks::Apiary, 33
BeeGame::Blocks::Block, 78
BeeGame::Blocks::Chest, 84
BeeGame::Blocks::CraftingTable, 115
BeeGame::Blocks::Dirt, 126
BeeGame::Blocks::Grass, 132
BeeGame::Blocks::Leaves, 179
BeeGame::Blocks::Planks, 197
BeeGame::Blocks::Wood, 314
BeeGame::Items::Bee, 53
BeeGame::Items::BeeAlyzer, 59
BeeGame::Items::HoneyComb, 139

BeeGame::Items::Item, 168

BeeGame::Items::QuestBook, 217

GetNoise
BeeGame::Terrain::LandGeneration::TerrainGeneration, 275

GetPrefab

BeeGame::Core::Dictionaries::PrefabDictionary, 211
GetPrefabs
 BeeGame::Resources::Resources, 232
GetShapedRecipeItem
 BeeGame::Core::Dictionaries::CraftingRecipies, 108
GetShaplessRecipeResult
 BeeGame::Core::Dictionaries::CraftingRecipies, 109, 110
GetShaplessRecipeString
 BeeGame::Core::Dictionaries::CraftingRecipies, 110
GetSprite
 BeeGame::Core::Dictionaries::SpriteDictionary, 261
GetSprites
 BeeGame::Resources::Resources, 232
GetWeights
 BeeGame::Core::Dictionaries::BeeDictionaries, 71
go
 BeeGame::Items::ItemGameObject, 175
grad
 BeeGame::Terrain::LandGeneration::Noise::SimplexNoise, 256, 257
Grass
 BeeGame::Blocks::Grass, 131
gravity
 BeeGame::Player::PlayerMove, 210

hit
 BeeGame::Player::Selector, 241
HoneyComb
 BeeGame::Items::HoneyComb, 137
HoneyCombType
 BeeGame::Core::Enums, 18
honeyCoubmbColour
 BeeGame::Core::Dictionaries::BeeDictionaries, 73

ID
 BeeGame::Blocks::Air, 27
 BeeGame::Blocks::Apiary, 38
 BeeGame::Blocks::Bedrock, 48
 BeeGame::Blocks::Block, 80
 BeeGame::Blocks::Chest, 86
 BeeGame::Blocks::CraftingTable, 118
 BeeGame::Blocks::Dirt, 127
 BeeGame::Blocks::Grass, 134
 BeeGame::Blocks::Leaves, 181
 BeeGame::Blocks::Planks, 198
 BeeGame::Blocks::Wood, 315
 BeeGame::Items::Bee, 55
 BeeGame::Items::BeeAlyzer, 61
 BeeGame::Items::Honey, 136
 BeeGame::Items::HoneyComb, 139
 BeeGame::Items::Item, 172
 BeeGame::Items::QuestBook, 218
identity
 BeeGame::Core::UnityTypeReplacements::THQuaternion, 290
InRange
 BeeGame::Terrain::Chunks::Chunk, 93
infoText
 BeeGame::Inventory::BeeAlyzerInventory, 66
inputButtons
 BeeGame::Core::THInput, 284
InputException
 BeeGame::Exceptions::InputException, 141
InteractWithBlock
 BeeGame::Blocks::Apiary, 33
 BeeGame::Blocks::Block, 78
 BeeGame::Blocks::Chest, 84
 BeeGame::Blocks::CraftingTable, 115
InteractWithItem
 BeeGame::Items::BeeAlyzer, 59
 BeeGame::Items::Item, 168
InteractWithObject
 BeeGame::Items::BeeAlyzer, 60
 BeeGame::Items::Item, 169
inventory
 BeeGame::Inventory::ChestInventory, 90
inventoryName
 BeeGame::Inventory::Inventory, 148
inventoryPosition
 BeeGame::Inventory::ChestInventory, 90
InventorySet
 BeeGame::Inventory::Inventory, 145
inventorySize
 BeeGame::Inventory::ChestInventory, 90
isAnotherInventoryOpen
 BeeGame::Core::THInput, 285
IsSolid
 BeeGame::Blocks::Air, 26
 BeeGame::Blocks::Block, 78
 BeeGame::Blocks::CraftingTable, 116
 BeeGame::Blocks::Leaves, 179
Item
 BeeGame::Items::Item, 160
item
 BeeGame::Inventory::InventorySlot, 156
 BeeGame::Items::ItemGameObject, 175
itemCraftedEvent
 BeeGame::Quest::QuestEvents, 224
ItemMesh
 BeeGame::Items::Item, 169
itemName
 BeeGame::Items::Item, 172
itemPickupEvent
 BeeGame::Quest::QuestEvents, 225
ItemRemovedFromResult
 BeeGame::Inventory::BlockInventory::CraftingTableInventory, 121
itemSprite
 BeeGame::Items::Bee, 55
 BeeGame::Items::HoneyComb, 139
itemSpriteDictionary

BeeGame::Core::Dictionaries::SpriteDictionary, 262
 itemStackCount
 BeeGame::Items::Item, 172
 itemText
 BeeGame::Inventory::InventorySlot, 157
 items
 BeeGame::Inventory::Inventory, 148
 itemsCanBeInserted
 BeeGame::Inventory::InventorySlot, 156
 ItemsInInventory
 BeeGame::Inventory::ItemsInInventory, 176
 itemsInInventory
 BeeGame::Inventory::ItemsInInventory, 177

 jumpHeight
 BeeGame::Player::PlayerMove, 210

 layers
 BeeGame::Player::Selector, 241
 Leaves
 BeeGame::Blocks::Leaves, 178
 LoadAndRenderChunks
 BeeGame::Terrain::Chunks::LoadChunks, 184
 LoadChunk
 BeeGame::Serialization::Serialization, 245
 LoadFile
 BeeGame::Serialization::Serialization, 245
 LoadPlayerPosition
 BeeGame::Serialization::Serialization, 246
 LoadPrefabs
 BeeGame::Core::Dictionaries::PrefabDictionary, 212
 LoadQuests
 BeeGame::Quest::Quests, 226
 BeeGame::Serialization::Serialization, 246
 LoadSprites
 BeeGame::Core::Dictionaries::SpriteDictionary, 261
 lockedQuests
 BeeGame::Quest::Quests, 230
 Look
 BeeGame::Player::PlayerLook, 204

 MakeBee
 BeeGame::Blocks::Apiary, 34
 MakeBeeWithStats
 BeeGame::Items::Bee, 54
 MakeBees
 BeeGame::Blocks::Apiary, 35
 MakeDirectories
 BeeGame::Serialization::Serialization, 247
 MakeMesh
 BeeGame::Items::ItemGameObject, 174
 maxStack
 BeeGame::Items::Bee, 56
 maxStackCount
 BeeGame::Items::Bee, 56
 BeeGame::Items::BeeAlyzer, 61

 BeeGame::Items::Item, 172
 BeeGame::Items::QuestBook, 218
 maxVelocity
 BeeGame::Player::PlayerMove, 210
 mesh
 BeeGame::Terrain::Chunks::Chunk, 97
 meshCollider
 BeeGame::Terrain::Chunks::Chunk, 98
 Mod
 BeeGame::Terrain::LandGeneration::Noise::SimplexNoise, 257
 MovePlayer
 BeeGame::Player::PlayerMove, 208
 mutationMultipler
 BeeGame::Blocks::Apiary, 38
 myGameObject
 BeeGame::Blocks::Apiary, 38
 BeeGame::Blocks::Chest, 86
 BeeGame::Blocks::CraftingTable, 118
 myInventory
 BeeGame::Inventory::InventorySlot, 157
 BeeGame::Items::BeeAlyzer, 61
 myItem
 BeeGame::Inventory::BeeAlyzerInventory, 67
 BeeGame::Inventory::QuestBookInventory, 222
 myRigidBody
 BeeGame::Player::PlayerMove, 210
 myTransform
 BeeGame::Player::PlayerLook, 206
 myblock
 BeeGame::Inventory::Inventory, 148

 nearbyChunks
 BeeGame::Terrain::Chunks::LoadChunks, 185
 normalBee
 BeeGame::Items::Bee, 56

 objects
 BeeGame::Items::ApplyColour, 45
 BeeGame::Items::SetBeeGOColours, 252
 OnCollisionStay
 BeeGame::Player::PlayerMove, 209
 OnDestroy
 BeeGame::Inventory::BlockInventory::CraftingTableInventory, 122
 OnDisable
 BeeGame::Inventory::InventorySlot, 152
 OnDrawGizmos
 BeeGame::SpawnItem, 259
 OnPointerClick
 BeeGame::Inventory::ApiaryCraftingOutputSlot, 39
 BeeGame::Inventory::CraftingOutputSlot, 103
 BeeGame::Inventory::InventorySlot, 152
 OnPointerEnter
 BeeGame::Inventory::InventorySlot, 154
 OnPointerExit
 BeeGame::Inventory::InventorySlot, 154
 OpenItemInventory
 BeeGame::Items::BeeAlyzer, 60

BeeGame::Items::QuestBook, 217
OpenPlayerInventory
 BeeGame::Inventory::Player_Inventory::Player←
 Inventory, 200
operator ChunkWorldPos
 BeeGame::Terrain::ChunkWorldPos, 100
operator Quaternion
 BeeGame::Core::THVector3, 302
 BeeGame::Core::UnityTypeReplacements::TH←
 Quaternion, 287
operator THQuaternion
 BeeGame::Core::UnityTypeReplacements::TH←
 Quaternion, 287
operator THVector2
 BeeGame::Core::THVector2, 293
operator THVector3
 BeeGame::Core::THVector3, 303
 BeeGame::Terrain::ChunkWorldPos, 101
operator Vector2
 BeeGame::Core::THVector2, 294
operator Vector3
 BeeGame::Core::THVector3, 303
operator!=
 BeeGame::Core::THVector2, 294
 BeeGame::Core::THVector3, 304
 BeeGame::Items::Item, 170
operator*
 BeeGame::Core::THVector2, 294, 295
 BeeGame::Core::THVector3, 304, 305
 BeeGame::Core::UnityTypeReplacements::TH←
 Quaternion, 287
operator+
 BeeGame::Core::THVector2, 295
 BeeGame::Core::THVector3, 306, 307
 BeeGame::Core::UnityTypeReplacements::TH←
 Quaternion, 288
operator-
 BeeGame::Core::THVector2, 296
 BeeGame::Core::THVector3, 307, 308
 BeeGame::Core::UnityTypeReplacements::TH←
 Quaternion, 288, 289
operator/
 BeeGame::Core::THVector2, 296, 297
 BeeGame::Core::THVector3, 309, 310
 BeeGame::Core::UnityTypeReplacements::TH←
 Quaternion, 289
operator==
 BeeGame::Core::THVector2, 297
 BeeGame::Core::THVector3, 310
 BeeGame::Items::Item, 170
pEffect
 BeeGame::Items::NormalBee, 193
pFertility
 BeeGame::Items::NormalBee, 193
pLifespan
 BeeGame::Items::NormalBee, 193
pProdSpeed
 BeeGame::Items::NormalBee, 194
pSpecies
 BeeGame::Items::NormalBee, 194
perm
 BeeGame::Terrain::LandGeneration::Noise::←
 SimplexNoise, 258
PickupItem
 BeeGame::Inventory::Player_Inventory::Player←
 Inventory, 201
PlaceBlock
 BeeGame::Player::Selector, 239
placeable
 BeeGame::Blocks::Block, 80
 BeeGame::Items::BeeAlyzer, 61
 BeeGame::Items::Item, 172
 BeeGame::Items::QuestBook, 218
Planks
 BeeGame::Blocks::Planks, 196
playerInventory
 BeeGame::Inventory::BeeAlyzerInventory, 67
 BeeGame::Inventory::Player_Inventory::Player←
 Inventory, 203
 BeeGame::Player::Selector, 241
playerinventory
 BeeGame::Inventory::ChestInventory, 90
prefabDictionary
 BeeGame::Core::Dictionaries::PrefabDictionary,
 212
Prefabs
 BeeGame::Resources::Resources, 234
previousBeeType
 BeeGame::Items::Bee, 56
pureBeeCraftedEvent
 BeeGame::Quest::QuestEvents, 225
PureBeeEventHandler
 BeeGame::Quest::QuestEvents, 224
PutItemsInSlots
 BeeGame::Inventory::Inventory, 145
queen
 BeeGame::Items::QueenBee, 214
QueenBee
 BeeGame::Items::QueenBee, 213
queenBee
 BeeGame::Items::Bee, 57
QuestAchieved
 BeeGame::Inventory::QuestBookInventory, 220
QuestAlreadyExistsException
 BeeGame::Exceptions::QuestAlreadyExists←
 Exception, 214, 215
QuestBook
 BeeGame::Items::QuestBook, 216
questButton
 BeeGame::Inventory::QuestBookInventory, 222
QuestEventHandler
 BeeGame::Quest::QuestEvents, 224
Rand
 BeeGame::Blocks::Apiary, 36
RemoveItemFromInventory

BeeGame::Inventory::Player_Inventory::Player←
 Inventory, 201
 RenderMesh
 BeeGame::Terrain::Chunks::Chunk, 94
 rendered
 BeeGame::Terrain::Chunks::Chunk, 98
 resourceCulture
 BeeGame::Resources::Resources, 233
 resourceMan
 BeeGame::Resources::Resources, 233
 ResourceManager
 BeeGame::Resources::Resources, 234
 Resources
 BeeGame::Resources::Resources, 231
 ReturnBeeCraftingQuest
 BeeGame::Quest::Quests, 227
 ReturnChange
 BeeGame::Blocks::Apiary, 36
 ReturnCompleatedClaimedQuests
 BeeGame::Quest::Quests, 227
 ReturnCompleatedQuests
 BeeGame::Quest::Quests, 227
 ReturnCraftingTableQuest
 BeeGame::Quest::Quests, 227
 ReturnCurrentQuests
 BeeGame::Quest::Quests, 228
 ReturnData
 BeeGame::Inventory::BeeAlyzerInventory, 64
 ReturnLockedQuests
 BeeGame::Quest::Quests, 228
 ReturnPickupQuest
 BeeGame::Quest::Quests, 228
 ReturnPureBreadBeeCraftingQuest
 BeeGame::Quest::Quests, 228
 ReturnShapedRecipieItem
 BeeGame::Blocks::CraftingTable, 116, 117
 ReturnShapelessRecipieItem
 BeeGame::Blocks::CraftingTable, 117
 rotationLock
 BeeGame::Player::PlayerLook, 206
 Round
 BeeGame::Terrain::LandGeneration::Terrain, 266
 RoundXZ
 BeeGame::Terrain::LandGeneration::Terrain, 267
 RoundY
 BeeGame::Terrain::LandGeneration::Terrain, 268
 sEffect
 BeeGame::Items::NormalBee, 194
 sFertility
 BeeGame::Items::NormalBee, 194
 sLifespan
 BeeGame::Items::NormalBee, 194
 sProdSpeed
 BeeGame::Items::NormalBee, 195
 sSpecies
 BeeGame::Items::NormalBee, 195
 SaveChunk
 BeeGame::Serialization::Serialization, 247
 BeeGame::Terrain::Chunks::SaveChunk, 235
 SaveFile
 BeeGame::Serialization::Serialization, 248
 saveFolderName
 BeeGame::Serialization::Serialization, 249
 SaveInv
 BeeGame::Inventory::BeeAlyzerInventory, 65
 BeeGame::Inventory::BlockInventory::Crafting←
 TableInventory, 122
 BeeGame::Inventory::Inventory, 145
 savePath
 BeeGame::Serialization::Serialization, 250
 SavePlayerPosition
 BeeGame::Serialization::Serialization, 248
 SaveQuests
 BeeGame::Serialization::Serialization, 249
 scrollObject
 BeeGame::Inventory::QuestBookInventory, 222
 selectedHotbarSlot
 BeeGame::Player::Selector, 241
 SelectedSlot
 BeeGame::Inventory::Player_Inventory::Player←
 Inventory, 202
 BeeGame::Player::Selector, 240
 selectedSlot
 BeeGame::Inventory::InventorySlot, 157
 selector
 BeeGame::Player::Selector, 242
 SerializeInventory
 BeeGame::Serialization::Serialization, 249
 SetAllItems
 BeeGame::Inventory::Inventory, 146
 SetBlock
 BeeGame::Terrain::Chunks::Chunk, 94
 BeeGame::Terrain::LandGeneration::Terrain, 269
 BeeGame::Terrain::LandGeneration::Terrain←
 Generation, 276
 BeeGame::Terrain::LandGeneration::World, 319
 SetBlocksUnmodified
 BeeGame::Terrain::Chunks::Chunk, 95
 SetChestInventory
 BeeGame::Inventory::ApiaryInventory, 42
 BeeGame::Inventory::BlockInventory::Crafting←
 TableInventory, 122
 BeeGame::Inventory::ChestInventory, 87
 SetInventorySize
 BeeGame::Inventory::Inventory, 146
 SetItemInventory
 BeeGame::Inventory::Inventory, 147
 SetPlayerInventory
 BeeGame::Inventory::Player_Inventory::Player←
 Inventory, 202
 SetPlayerItems
 BeeGame::Inventory::BeeAlyzerInventory, 65
 BeeGame::Inventory::ChestInventory, 88
 shapedCraftingRecipes
 BeeGame::Core::Dictionaries::CraftingRecipes,
 111

shaplessRecipes
 BeeGame::Core::Dictionaries::CraftingRecipies, 111

shareMeshes
 BeeGame::Terrain::Chunks::MeshData, 191

slotIndex
 BeeGame::Inventory::InventorySlot, 157

slots
 BeeGame::Inventory::Inventory, 148

SpawnItem
 BeeGame::Core::Extensions, 130

speed
 BeeGame::Player::PlayerLook, 206
 BeeGame::Player::PlayerMove, 211

SplitStack
 BeeGame::Inventory::InventorySlot, 155

spriteAtCursor
 BeeGame::Inventory::Inventory, 148

Sprites
 BeeGame::Resources::Resources, 234

Start
 BeeGame::Inventory::BlockInventory::Crafting←
 TableInventory, 123
 BeeGame::Inventory::QuestBookInventory, 220
 BeeGame::Items::ApplyColour, 45
 BeeGame::Items::ItemGameObject, 174
 BeeGame::Items::SetBeeGOColours, 251
 BeeGame::Player::PlayerLook, 205
 BeeGame::SpawnItem, 259
 BeeGame::Terrain::Chunks::Chunk, 95
 BeeGame::Terrain::Chunks::LoadChunks, 184
 BeeGame::Test, 280

stoneBaseHeight
 BeeGame::Terrain::LandGeneration::Terrain←
 Generation, 277

stoneBaseNoise
 BeeGame::Terrain::LandGeneration::Terrain←
 Generation, 278

stoneBaseNoiseHeight
 BeeGame::Terrain::LandGeneration::Terrain←
 Generation, 278

stoneMinHeight
 BeeGame::Terrain::LandGeneration::Terrain←
 Generation, 278

stoneMountainFrequency
 BeeGame::Terrain::LandGeneration::Terrain←
 Generation, 278

stoneMountainHeight
 BeeGame::Terrain::LandGeneration::Terrain←
 Generation, 278

SwapItems
 BeeGame::Inventory::InventorySlot, 155

THQuaternion
 BeeGame::Core::UnityTypeReplacements::TH←
 Quaternion, 286

THVector2
 BeeGame::Core::THVector2, 292

THVector3
 BeeGame::Core::THVector3, 300, 301

TexturePosition
 BeeGame::Blocks::Apiary, 37
 BeeGame::Blocks::Bedrock, 47
 BeeGame::Blocks::Chest, 85
 BeeGame::Blocks::CraftingTable, 117
 BeeGame::Blocks::Dirt, 126
 BeeGame::Blocks::Grass, 132
 BeeGame::Blocks::Leaves, 180
 BeeGame::Blocks::Planks, 197
 BeeGame::Blocks::Wood, 314
 BeeGame::Items::Item, 171

thisInventoryOpen
 BeeGame::Inventory::Inventory, 149

tiara
 BeeGame::Items::SetBeeGOColours, 252

tileSize
 BeeGame::Items::Item, 172

timer
 BeeGame::Terrain::Chunks::LoadChunks, 186

timerSlideer
 BeeGame::Inventory::ApiaryInventory, 44

ToString
 BeeGame::Blocks::Air, 26
 BeeGame::Blocks::Apiary, 38
 BeeGame::Blocks::Bedrock, 48
 BeeGame::Blocks::Block, 79
 BeeGame::Blocks::Chest, 85
 BeeGame::Blocks::Dirt, 127
 BeeGame::Blocks::Grass, 133
 BeeGame::Blocks::Leaves, 180
 BeeGame::Blocks::Planks, 198
 BeeGame::Blocks::Wood, 315
 BeeGame::Core::THVector2, 297
 BeeGame::Core::THVector3, 311
 BeeGame::Items::Honey, 135
 BeeGame::Items::Item, 171
 BeeGame::Terrain::ChunkWorldPos, 101

ToggleInventory
 BeeGame::Inventory::BeeAlyzerInventory, 65
 BeeGame::Inventory::BlockInventory::Crafting←
 TableInventory, 123
 BeeGame::Inventory::ChestInventory, 88
 BeeGame::Inventory::Inventory, 147
 BeeGame::Inventory::QuestBookInventory, 221

treeDensity
 BeeGame::Terrain::LandGeneration::Terrain←
 Generation, 279

treeFrequency
 BeeGame::Terrain::LandGeneration::Terrain←
 Generation, 279

tris
 BeeGame::Terrain::Chunks::MeshData, 191

type
 BeeGame::Items::HoneyComb, 140

UnlockQuests
 BeeGame::Quest::Quests, 229

Update

BeeGame::Inventory::ApiaryCraftingOutputSlot, 40 x
 BeeGame::Inventory::ApiaryInventory, 43
 BeeGame::Inventory::BeeAlyzerInventory, 66
 BeeGame::Inventory::BlockInventory::Crafting←
 TableInventory, 123
 BeeGame::Inventory::ChestInventory, 89
 BeeGame::Inventory::CraftingOutputSlot, 103
 BeeGame::Inventory::InventorySlot, 155
 BeeGame::Inventory::Player_Inventory::Player←
 Inventory, 202
 BeeGame::Inventory::QuestBookInventory, 221
 BeeGame::Items::ItemGameObject, 174
 BeeGame::LoadResources, 187
 BeeGame::Player::PlayerLook, 205
 BeeGame::Player::SavePlayerPosition, 236
 BeeGame::Player::Selector, 240
 BeeGame::Terrain::Chunks::Chunk, 96
 BeeGame::Terrain::Chunks::LoadChunks, 185
 BeeGame::Test, 280

update
 BeeGame::Terrain::Chunks::Chunk, 98

UpdateBase
 BeeGame::Inventory::Inventory, 147

UpdateBlock
 BeeGame::Blocks::Block, 79
 BeeGame::Blocks::Grass, 133

UpdateChestInventory
 BeeGame::Inventory::ChestInventory, 89

UpdateChunk
 BeeGame::Terrain::Chunks::Chunk, 96

updateCollisionMesh
 BeeGame::Terrain::Chunks::Chunk, 98

UpdateIcon
 BeeGame::Inventory::InventorySlot, 156

UpdateIfEqual
 BeeGame::Terrain::LandGeneration::World, 320

UpdateSelector
 BeeGame::Player::Selector, 240

usesGameObject
 BeeGame::Items::Item, 173

uv
 BeeGame::Terrain::Chunks::MeshData, 192

VerticalJumpSpeed
 BeeGame::Player::PlayerMove, 209

verts
 BeeGame::Terrain::Chunks::MeshData, 192

w
 BeeGame::Core::UnityTypeReplacements::TH←
 Quaternion, 290

Wood
 BeeGame::Blocks::Wood, 314

world
 BeeGame::Terrain::Chunks::Chunk, 98
 BeeGame::Terrain::Chunks::LoadChunks, 186
 BeeGame::Terrain::LandGeneration::Terrain, 270

worldName
 BeeGame::Serialization::Serialization, 250