

Bee Game

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	7
4.1	File List	7
5	Namespace Documentation	10
5.1	BeeGame Namespace Reference	10
5.2	BeeGame.Blocks Namespace Reference	11
5.3	BeeGame.Core Namespace Reference	11
5.4	BeeGame.Core.Dictionarys Namespace Reference	11
5.5	BeeGame.Core.Enums Namespace Reference	12
5.5.1	Enumeration Type Documentation	12
5.6	BeeGame.Core.UnityTypeReplacements Namespace Reference	17
5.7	BeeGame.Exceptipns Namespace Reference	17
5.8	BeeGame.Inventory Namespace Reference	17
5.9	BeeGame.Inventory.BlockInventory Namespace Reference	18
5.10	BeeGame.Inventory.Player_Inventory Namespace Reference	18
5.11	BeeGame.Items Namespace Reference	18
5.12	BeeGame.Player Namespace Reference	18
5.13	BeeGame.Quest Namespace Reference	19
5.14	BeeGame.Resources Namespace Reference	19
5.15	BeeGame.Serialization Namespace Reference	19
5.16	BeeGame.Terrain Namespace Reference	19
5.17	BeeGame.Terrain.Chunks Namespace Reference	19
5.18	BeeGame.Terrain.LandGeneration Namespace Reference	20
5.19	BeeGame.Terrain.LandGeneration.Noise Namespace Reference	20

6	Class Documentation	20
6.1	BeeGame.Items.AbstractItem Class Reference	20
6.1.1	Detailed Description	21
6.1.2	Member Function Documentation	21
6.2	BeeGame.Blocks.Air Class Reference	22
6.2.1	Detailed Description	22
6.2.2	Constructor & Destructor Documentation	22
6.2.3	Member Function Documentation	23
6.2.4	Member Data Documentation	25
6.3	BeeGame.Blocks.Apiary Class Reference	25
6.3.1	Detailed Description	27
6.3.2	Constructor & Destructor Documentation	27
6.3.3	Member Function Documentation	27
6.3.4	Member Data Documentation	37
6.4	BeeGame.Inventory.ApiaryInventory Class Reference	37
6.4.1	Detailed Description	38
6.4.2	Member Function Documentation	38
6.4.3	Member Data Documentation	40
6.5	BeeGame.Items.ApplyColour Class Reference	41
6.5.1	Detailed Description	42
6.5.2	Member Function Documentation	42
6.5.3	Member Data Documentation	42
6.6	BeeGame.Blocks.Bedrock Class Reference	43
6.6.1	Detailed Description	43
6.6.2	Constructor & Destructor Documentation	43
6.6.3	Member Function Documentation	44
6.6.4	Member Data Documentation	45
6.7	BeeGame.Items.Bee Class Reference	46
6.7.1	Detailed Description	47
6.7.2	Constructor & Destructor Documentation	47

6.7.3	Member Function Documentation	48
6.7.4	Member Data Documentation	51
6.7.5	Property Documentation	52
6.8	BeeGame.Core.Dictionarys.BeeCombinationDictionaryEqualityComparer Class Reference	53
6.8.1	Detailed Description	54
6.8.2	Member Function Documentation	54
6.9	BeeGame.Core.Dictionarys.BeeDictionarys Class Reference	55
6.9.1	Detailed Description	55
6.9.2	Member Function Documentation	55
6.9.3	Member Data Documentation	58
6.10	BeeGame.Blocks.Block Class Reference	60
6.10.1	Detailed Description	61
6.10.2	Constructor & Destructor Documentation	61
6.10.3	Member Function Documentation	62
6.10.4	Member Data Documentation	66
6.11	BeeGame.Blocks.Chest Class Reference	67
6.11.1	Detailed Description	67
6.11.2	Constructor & Destructor Documentation	68
6.11.3	Member Function Documentation	68
6.11.4	Member Data Documentation	71
6.12	BeeGame.Inventory.ChestInventory Class Reference	72
6.12.1	Detailed Description	73
6.12.2	Member Function Documentation	73
6.12.3	Member Data Documentation	75
6.13	BeeGame.Terrain.Chunks.Chunk Class Reference	76
6.13.1	Detailed Description	77
6.13.2	Member Function Documentation	78
6.13.3	Member Data Documentation	82
6.14	BeeGame.Terrain.ChunkWorldPos Struct Reference	84
6.14.1	Detailed Description	85

6.14.2	Constructor & Destructor Documentation	85
6.14.3	Member Function Documentation	85
6.14.4	Member Data Documentation	87
6.15	BeeGame.Exceptipns.CraftingRecipieAdditionException Class Reference	88
6.15.1	Detailed Description	88
6.15.2	Constructor & Destructor Documentation	88
6.16	BeeGame.Core.Dictionarys.CraftingRecipies Class Reference	89
6.16.1	Detailed Description	90
6.16.2	Member Function Documentation	90
6.16.3	Member Data Documentation	94
6.17	BeeGame.Blocks.CraftingTable Class Reference	95
6.17.1	Detailed Description	96
6.17.2	Constructor & Destructor Documentation	96
6.17.3	Member Function Documentation	96
6.17.4	Member Data Documentation	101
6.18	BeeGame.Inventory.BlockInventory.CraftingTableInventory Class Reference	101
6.18.1	Detailed Description	102
6.18.2	Member Function Documentation	102
6.18.3	Member Data Documentation	107
6.19	BeeGame.Blocks.Dirt Class Reference	107
6.19.1	Detailed Description	108
6.19.2	Constructor & Destructor Documentation	108
6.19.3	Member Function Documentation	108
6.19.4	Member Data Documentation	110
6.20	BeeGame.Core.Events Class Reference	110
6.20.1	Detailed Description	110
6.20.2	Member Function Documentation	110
6.20.3	Member Data Documentation	111
6.21	BeeGame.Core.Extensions Class Reference	112
6.21.1	Detailed Description	112

6.21.2	Member Function Documentation	112
6.22	BeeGame.Blocks.Grass Class Reference	115
6.22.1	Detailed Description	115
6.22.2	Constructor & Destructor Documentation	115
6.22.3	Member Function Documentation	116
6.22.4	Member Data Documentation	118
6.23	BeeGame.Items.HoneyComb Class Reference	118
6.23.1	Detailed Description	119
6.23.2	Constructor & Destructor Documentation	119
6.23.3	Member Function Documentation	120
6.23.4	Member Data Documentation	122
6.23.5	Property Documentation	122
6.24	BeeGame.Exceptions.InputException Class Reference	123
6.24.1	Detailed Description	123
6.24.2	Constructor & Destructor Documentation	123
6.25	BeeGame.Inventory.Inventory Class Reference	124
6.25.1	Detailed Description	125
6.25.2	Member Function Documentation	125
6.25.3	Member Data Documentation	129
6.26	BeeGame.Inventory.InventorySlot Class Reference	131
6.26.1	Detailed Description	132
6.26.2	Member Function Documentation	132
6.26.3	Member Data Documentation	138
6.27	BeeGame.Items.Item Class Reference	140
6.27.1	Detailed Description	142
6.27.2	Constructor & Destructor Documentation	142
6.27.3	Member Function Documentation	142
6.27.4	Member Data Documentation	153
6.27.5	Property Documentation	154
6.28	BeeGame.Items.ItemGameObject Class Reference	155

6.28.1 Detailed Description	156
6.28.2 Member Function Documentation	156
6.28.3 Member Data Documentation	157
6.29 BeeGame.Inventory.ItemsInInventory Class Reference	157
6.29.1 Detailed Description	158
6.29.2 Constructor & Destructor Documentation	158
6.29.3 Member Function Documentation	158
6.29.4 Member Data Documentation	159
6.30 BeeGame.Blocks.Leaves Class Reference	160
6.30.1 Detailed Description	160
6.30.2 Constructor & Destructor Documentation	160
6.30.3 Member Function Documentation	161
6.30.4 Member Data Documentation	163
6.31 BeeGame.Terrain.Chunks.LoadChunks Class Reference	163
6.31.1 Detailed Description	164
6.31.2 Member Function Documentation	164
6.31.3 Member Data Documentation	167
6.32 BeeGame.LoadResources Class Reference	168
6.32.1 Detailed Description	169
6.32.2 Member Function Documentation	169
6.33 BeeGame.Terrain.Chunks.MeshData Class Reference	169
6.33.1 Detailed Description	170
6.33.2 Member Function Documentation	170
6.33.3 Member Data Documentation	172
6.34 BeeGame.Items.NormalBee Class Reference	173
6.34.1 Detailed Description	174
6.34.2 Member Function Documentation	174
6.34.3 Member Data Documentation	174
6.35 BeeGame.Inventory.Player_Inventory.PlayerInventory Class Reference	176
6.35.1 Detailed Description	177

6.35.2	Member Function Documentation	177
6.35.3	Member Data Documentation	181
6.36	BeeGame.Player.PlayerLook Class Reference	181
6.36.1	Detailed Description	182
6.36.2	Member Function Documentation	182
6.36.3	Member Data Documentation	183
6.37	BeeGame.Player.PlayerMove Class Reference	184
6.37.1	Detailed Description	185
6.37.2	Member Function Documentation	185
6.37.3	Member Data Documentation	187
6.38	BeeGame.Core.Dictionary.PrefabDictionary Class Reference	188
6.38.1	Detailed Description	189
6.38.2	Member Function Documentation	189
6.38.3	Member Data Documentation	190
6.39	BeeGame.Items.QueenBee Class Reference	190
6.39.1	Detailed Description	190
6.39.2	Constructor & Destructor Documentation	190
6.39.3	Member Function Documentation	191
6.39.4	Property Documentation	191
6.40	BeeGame.Quest.QuestBook Class Reference	192
6.40.1	Detailed Description	192
6.40.2	Constructor & Destructor Documentation	192
6.40.3	Member Function Documentation	192
6.40.4	Member Data Documentation	194
6.41	BeeGame.Resources.Resources Class Reference	194
6.41.1	Detailed Description	194
6.41.2	Constructor & Destructor Documentation	195
6.41.3	Member Function Documentation	195
6.41.4	Member Data Documentation	196
6.41.5	Property Documentation	196

6.42	BeeGame.Terrain.Chunks.SaveChunk Class Reference	197
6.42.1	Detailed Description	198
6.42.2	Constructor & Destructor Documentation	198
6.42.3	Member Data Documentation	198
6.43	BeeGame.Player.SavePlayerPosition Class Reference	199
6.43.1	Detailed Description	199
6.43.2	Member Function Documentation	199
6.43.3	Member Data Documentation	200
6.44	BeeGame.Player.Selector Class Reference	200
6.44.1	Detailed Description	201
6.44.2	Member Function Documentation	201
6.44.3	Member Data Documentation	204
6.45	BeeGame.Serialization.Serialization Class Reference	205
6.45.1	Detailed Description	206
6.45.2	Member Function Documentation	206
6.45.3	Member Data Documentation	211
6.46	BeeGame.Terrain.LandGeneration.Noise.SimplexNoise Class Reference	212
6.46.1	Detailed Description	213
6.46.2	Member Function Documentation	213
6.46.3	Member Data Documentation	218
6.47	BeeGame.SpawnItem Class Reference	219
6.47.1	Detailed Description	219
6.47.2	Member Function Documentation	219
6.48	BeeGame.Core.Dictionarys.SpriteDictionary Class Reference	220
6.48.1	Detailed Description	221
6.48.2	Member Function Documentation	221
6.48.3	Member Data Documentation	222
6.49	BeeGame.Terrain.LandGeneration.Terrain Class Reference	222
6.49.1	Detailed Description	223
6.49.2	Member Function Documentation	223

6.49.3	Member Data Documentation	230
6.50	BeeGame.Terrain.LandGeneration.TerrainGeneration Class Reference	230
6.50.1	Detailed Description	231
6.50.2	Member Function Documentation	232
6.50.3	Member Data Documentation	237
6.51	BeeGame.Test Class Reference	239
6.51.1	Detailed Description	240
6.51.2	Member Function Documentation	240
6.52	BeeGame.Core.THInput Class Reference	240
6.52.1	Detailed Description	241
6.52.2	Member Function Documentation	241
6.52.3	Member Data Documentation	244
6.53	BeeGame.Core.UnityTypeReplacements.THQuaternion Struct Reference	245
6.53.1	Detailed Description	245
6.53.2	Member Data Documentation	246
6.54	BeeGame.Core.THVector2 Struct Reference	246
6.54.1	Detailed Description	247
6.54.2	Constructor & Destructor Documentation	247
6.54.3	Member Function Documentation	248
6.54.4	Member Data Documentation	253
6.55	BeeGame.Core.THVector3 Struct Reference	254
6.55.1	Detailed Description	255
6.55.2	Constructor & Destructor Documentation	255
6.55.3	Member Function Documentation	257
6.55.4	Member Data Documentation	267
6.56	BeeGame.Items.Tile Struct Reference	267
6.56.1	Detailed Description	268
6.56.2	Member Data Documentation	268
6.57	BeeGame.Blocks.Wood Class Reference	268
6.57.1	Detailed Description	269
6.57.2	Constructor & Destructor Documentation	269
6.57.3	Member Function Documentation	269
6.57.4	Member Data Documentation	271
6.58	BeeGame.Terrain.LandGeneration.World Class Reference	271
6.58.1	Detailed Description	272
6.58.2	Member Function Documentation	272
6.58.3	Member Data Documentation	277

7	File Documentation	277
7.1	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Air.cs File Reference	277
7.2	Air.cs	278
7.3	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Apiary.cs File Reference	278
7.4	Apiary.cs	278
7.5	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Bedrock.cs File Reference	282
7.6	Bedrock.cs	282
7.7	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Block.cs File Reference	283
7.8	Block.cs	283
7.9	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Chest.cs File Reference	284
7.10	Chest.cs	285
7.11	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Crafting↔Table.cs File Reference	286
7.12	CraftingTable.cs	286
7.13	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Dirt.cs File Reference	287
7.14	Dirt.cs	288
7.15	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Grass.cs File Reference	288
7.16	Grass.cs	288
7.17	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Leaves.cs File Reference	289
7.18	Leaves.cs	290
7.19	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Wood.cs File Reference	290
7.20	Wood.cs	290
7.21	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionary/↔BeeDictionary.cs File Reference	291
7.22	BeeDictionary.cs	291
7.23	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionary/↔CraftingRecipies.cs File Reference	293

7.24	CraftingRecipies.cs	293
7.25	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/↔ EqualityComperors.cs File Reference	295
7.26	EqualityComperors.cs	295
7.27	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/↔ PrefabDictionary.cs File Reference	296
7.28	PrefabDictionary.cs	296
7.29	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/↔ SpriteDictionary.cs File Reference	296
7.30	SpriteDictionary.cs	297
7.31	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Enums/↔ Enums.cs File Reference	297
7.32	Enums.cs	298
7.33	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Events.cs File Reference	299
7.34	Events.cs	299
7.35	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Extensions.cs File Reference	300
7.36	Extensions.cs	300
7.37	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityType↔ Replacements/THInput.cs File Reference	302
7.38	THInput.cs	302
7.39	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityType↔ Replacements/THQuaternion.cs File Reference	304
7.40	THQuaternion.cs	304
7.41	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityType↔ Replacements/THVector2.cs File Reference	304
7.42	THVector2.cs	304
7.43	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityType↔ Replacements/THVector3.cs File Reference	306
7.44	THVector3.cs	307
7.45	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Exceptipns/↔ CraftingRecipieAdditionException.cs File Reference	309
7.46	CraftingRecipieAdditionException.cs	309
7.47	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Exceptipns/Input↔ Exception.cs File Reference	309

7.48	InputException.cs	310
7.49	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Block↔ Inventory/ApiaryInventory.cs File Reference	310
7.50	ApiaryInventory.cs	310
7.51	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Block↔ Inventory/ChestInventory.cs File Reference	312
7.52	ChestInventory.cs	312
7.53	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Block↔ Inventory/CraftingTableInventory.cs File Reference	313
7.54	CraftingTableInventory.cs	313
7.55	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Inventory.cs File Reference	315
7.56	Inventory.cs	315
7.57	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Inventory↔ Slot.cs File Reference	317
7.58	InventorySlot.cs	317
7.59	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/ItemsIn↔ Inventory.cs File Reference	320
7.60	ItemsInInventory.cs	321
7.61	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Player Inventory/PlayerInventory.cs File Reference	321
7.62	PlayerInventory.cs	321
7.63	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/Abstract↔ Item.cs File Reference	323
7.64	AbstractItem.cs	323
7.65	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/ApplyColour.cs File Reference	324
7.66	ApplyColour.cs	324
7.67	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/Bee.cs File Reference	324
7.68	Bee.cs	325
7.69	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/Honey↔ Comb.cs File Reference	327
7.70	HoneyComb.cs	328
7.71	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/Item.cs File Reference	328

7.72	Item.cs	329
7.73	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/ItemGame↔ Object.cs File Reference	332
7.74	ItemGameObject.cs	333
7.75	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/LoadResources.cs File Reference	333
7.76	LoadResources.cs	334
7.77	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/obj/Debug/Temporary↔ GeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs File Reference	334
7.78	TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs	334
7.79	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/obj/Debug/Temporary↔ GeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs File Reference	334
7.80	TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs	334
7.81	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/obj/Debug/Temporary↔ GeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs File Reference	334
7.82	TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs	334
7.83	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/PlayerLook.cs File Reference	334
7.84	PlayerLook.cs	335
7.85	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/Player↔ Move.cs File Reference	335
7.86	PlayerMove.cs	336
7.87	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/SavePlayer↔ Position.cs File Reference	337
7.88	SavePlayerPosition.cs	337
7.89	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/Selector.cs File Reference	337
7.90	Selector.cs	338
7.91	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Properties/Assembly↔ Info.cs File Reference	339
7.92	AssemblyInfo.cs	339
7.93	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Quest/QuestBook.cs File Reference	340
7.94	QuestBook.cs	340
7.95	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Resources/Resources.Designer.↔ cs File Reference	340

7.96 Resources.Designer.cs	341
7.97 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Serialization/↔ Serialization.cs File Reference	342
7.98 Serialization.cs	343
7.99 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/SpawnlTem.cs File Reference	345
7.100SpawnlTem.cs	345
7.101 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/↔ Chunk.cs File Reference	346
7.102Chunk.cs	346
7.103C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/↔ LoadChunks.cs File Reference	349
7.104LoadChunks.cs	349
7.105C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/↔ MeshData.cs File Reference	352
7.106MeshData.cs	353
7.107C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/↔ SaveChunk.cs File Reference	353
7.108SaveChunk.cs	354
7.109C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunk↔ WorldPos.cs File Reference	354
7.110ChunkWorldPos.cs	354
7.111 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Land↔ Generation/Noise/SimplexNoise.cs File Reference	355
7.112SimplexNoise.cs	355
7.113C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Land↔ Generation/Terrain.cs File Reference	359
7.114Terrain.cs	359
7.115C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Land↔ Generation/TerrainGeneration.cs File Reference	362
7.116TerrainGeneration.cs	362
7.117C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Land↔ Generation/World.cs File Reference	364
7.118World.cs	364
7.119C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/test.cs File Reference	367
7.120test.cs	367

Index	369
-----------------------	-----

1 Namespace Index

1.1 Packages

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

BeeGame	10
BeeGame.Blocks	11
BeeGame.Core	11
BeeGame.Core.Dictionarys	11
BeeGame.Core.Enums	12
BeeGame.Core.UnityTypeReplacements	17
BeeGame.Exceptipns	17
BeeGame.Inventory	17
BeeGame.Inventory.BlockInventory	18
BeeGame.Inventory.Player_Inventory	18
BeeGame.Items	18
BeeGame.Player	18
BeeGame.Quest	19
BeeGame.Resources	19
BeeGame.Serialization	19
BeeGame.Terrain	19
BeeGame.Terrain.Chunks	19
BeeGame.Terrain.LandGeneration	20
BeeGame.Terrain.LandGeneration.Noise	20

2 Hierarchical Index

2.1 Class Hierarchy

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

BeeGame.Items.AbstractItem	20
BeeGame.Items.Item	140

BeeGame.Blocks.Block	60
BeeGame.Blocks.Air	22
BeeGame.Blocks.Apiary	25
BeeGame.Blocks.Bedrock	43
BeeGame.Blocks.Chest	67
BeeGame.Blocks.CraftingTable	95
BeeGame.Blocks.Dirt	107
BeeGame.Blocks.Grass	115
BeeGame.Blocks.Leaves	160
BeeGame.Blocks.Wood	268
BeeGame.Items.Bee	46
BeeGame.Items.HoneyComb	118
BeeGame.Quest.QuestBook	192
BeeGame.Core.Dictionarys.BeeDictionarys	55
BeeGame.Terrain.ChunkWorldPos	84
BeeGame.Core.Dictionarys.CraftingRecipies	89
BeeGame.Core.Events	110
Exception	
BeeGame.Exceptipns.CraftingRecipieAdditionException	88
BeeGame.Exceptipns.InputException	123
BeeGame.Core.Extensions	112
ICloneable	
BeeGame.Items.Item	140
IEqualityComparer	
BeeGame.Core.Dictionarys.BeeCombinationDictionaryEqualityComparer	53
IPointerClickHandler	
BeeGame.Inventory.InventorySlot	131
IPointerEnterHandler	
BeeGame.Inventory.InventorySlot	131
IPointerExitHandler	
BeeGame.Inventory.InventorySlot	131
BeeGame.Inventory.ItemsInInventory	157
BeeGame.Terrain.Chunks.MeshData	169
MonoBehaviour	
BeeGame.Inventory.Inventory	124

BeeGame.Inventory.ChestInventory	72
BeeGame.Inventory.ApiaryInventory	37
BeeGame.Inventory.BlockInventory.CraftingTableInventory	101
BeeGame.Inventory.Player_Inventory.PlayerInventory	176
BeeGame.Inventory.InventorySlot	131
BeeGame.Items.ApplyColour	41
BeeGame.Items.ItemGameObject	155
BeeGame.LoadResources	168
BeeGame.Player.PlayerLook	181
BeeGame.Player.PlayerMove	184
BeeGame.Player.SavePlayerPosition	199
BeeGame.Player.Selector	200
BeeGame.SpawnItem	219
BeeGame.Terrain.Chunks.Chunk	76
BeeGame.Terrain.Chunks.LoadChunks	163
BeeGame.Terrain.LandGeneration.World	271
BeeGame.Test	239
BeeGame.Items.NormalBee	173
BeeGame.Core.Dictionarys.PrefabDictionary	188
BeeGame.Items.QueenBee	190
BeeGame.Resources.Resources	194
BeeGame.Terrain.Chunks.SaveChunk	197
BeeGame.Serialization.Serialization	205
BeeGame.Terrain.LandGeneration.Noise.SimplexNoise	212
BeeGame.Core.Dictionarys.SpriteDictionary	220
BeeGame.Terrain.LandGeneration.Terrain	222
BeeGame.Terrain.LandGeneration.TerrainGeneration	230
BeeGame.Core.THInput	240
BeeGame.Core.UnityTypeReplacements.THQuaternion	245
BeeGame.Core.THVector2	246
BeeGame.Core.THVector3	254
BeeGame.Items.Tile	267

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?	20
BeeGame.Blocks.Air	
Air Block is an empty block that does not render and has no collider	22
BeeGame.Blocks.Apiary	
Apiary Block	25
BeeGame.Inventory.ApiaryInventory	
Inventory for Apiarys Apiary	37
BeeGame.Items.ApplyColour	
Applies a given colour to a gameobject	41
BeeGame.Blocks.Bedrock	
Bedrock Block	43
BeeGame.Items.Bee	
The bee item	46
BeeGame.Core.Dictionarys.BeeCombinationDictionaryEqualityComparer	53
BeeGame.Core.Dictionarys.BeeDictionarys	55
BeeGame.Blocks.Block	
Base class for blocks	60
BeeGame.Blocks.Chest	
Chest Block	67
BeeGame.Inventory.ChestInventory	
Incentory for the chests	72
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	76
BeeGame.Terrain.ChunkWorldPos	
Serializable int version of THVector3	84
BeeGame.Exceptipns.CraftingRecipieAdditionException	88
BeeGame.Core.Dictionarys.CraftingRecipies	89
BeeGame.Blocks.CraftingTable	
The Workbanch Block class	95
BeeGame.Inventory.BlockInventory.CraftingTableInventory	
Invnetory for the CraftingTable Block	101
BeeGame.Blocks.Dirt	
Dirt Block	107
BeeGame.Core.Events	110

BeeGame.Core.Extensions	112
BeeGame.Blocks.Grass	
Grass Block	115
BeeGame.Items.HoneyComb	
Honey comb item produced by bees	118
BeeGame.Exceptipns.InputException	123
BeeGame.Inventory.Inventory	
Base class for all inventorys in the game	124
BeeGame.Inventory.InventorySlot	131
BeeGame.Items.Item	
Base class for all Items and Blocks in the game	140
BeeGame.Items.ItemGameObject	
Interface between item and inity gameobjects	155
BeeGame.Inventory.ItemsInInventory	
Class that holds all of the items in the inventory. Can be serialized so inventory may be saved	157
BeeGame.Blocks.Leaves	160
BeeGame.Terrain.Chunks.LoadChunks	
Loads the Chunks around the player	163
BeeGame.LoadResources	
Loads all of the resources in the game	168
BeeGame.Terrain.Chunks.MeshData	
The data for a Chunks 's Mesh	169
BeeGame.Items.NormalBee	173
BeeGame.Inventory.Player_Inventory.PlayerInventory	
Controls the player inventory	176
BeeGame.Player.PlayerLook	
The look for the player	181
BeeGame.Player.PlayerMove	
Moves the player	184
BeeGame.Core.Dictionarys.PrefabDictionary	
The prefabs avaiable to the game	188
BeeGame.Items.QueenBee	190
BeeGame.Quest.QuestBook	192
BeeGame.Resources.Resources	
A strongly-typed resource class, for looking up localized strings, etc.	194
BeeGame.Terrain.Chunks.SaveChunk	
Saves a Chunks modified Blocks for save optimisation	197
BeeGame.Player.SavePlayerPosition	
Saves the player postion	199

BeeGame.Player.Selector	
Moves the Block selector	200
BeeGame.Serialization.Serialization	
Serializes and Deserialises things	205
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/	212
BeeGame.SpawnItem	219
BeeGame.Core.Dictionarys.SpriteDictionary	
All of the sprites available to the game	220
BeeGame.Terrain.LandGeneration.Terrain	
Should use as an interface between the rest of the game and the terrain	222
BeeGame.Terrain.LandGeneration.TerrainGeneration	
Generates the terrain for the game	230
BeeGame.Test	239
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	240
BeeGame.Core.UnityTypeReplacements.THQuaternion	245
BeeGame.Core.THVector2	
Serializable version of Vector2	246
BeeGame.Core.THVector3	
Serializable version of Vector3	254
BeeGame.Items.Tile	
Position of the items texture	267
BeeGame.Blocks.Wood	268
BeeGame.Terrain.LandGeneration.World	
Allows inter Chunk communication as it stores a list of active chunks	271

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	333
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/SpawnItem.cs	345
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/test.cs	367
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Air.cs	277

C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/ Apiary.cs	278
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/ Bedrock.cs	282
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/ Block.cs	283
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/ Chest.cs	284
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/ Crafting↔ Table.cs	286
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/ Dirt.cs	287
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/ Grass.cs	288
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/ Leaves.cs	289
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/ Wood.cs	290
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/ Events.cs	299
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/ Extensions.cs	300
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/ Bee↔ Dictionary.cs	291
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/ Crafting↔ Recipies.cs	293
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/ Equality↔ Comperors.cs	295
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/ Prefab↔ Dictionary.cs	296
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/ Sprite↔ Dictionary.cs	296
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Enums/ Enums.↔ cs	297
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityType↔ Replacements/ THInput.cs	302
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityType↔ Replacements/ THQuaternion.cs	304
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityType↔ Replacements/ THVector2.cs	304
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityType↔ Replacements/ THVector3.cs	306
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Exceptipns/ Crafting↔ RecipieAdditionException.cs	309
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Exceptipns/ Input↔ Exception.cs	309
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/ Inventory.↔ cs	315

C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Inventory↔ Slot.cs	317
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/ItemsIn↔ Inventory.cs	320
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Block↔ Inventory/ApiaryInventory.cs	310
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Block↔ Inventory/ChestInventory.cs	312
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Block↔ Inventory/CraftingTableInventory.cs	313
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Player In- ventory/PlayerInventory.cs	321
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/AbstractItem.↔ cs	323
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/ApplyColour.cs	324
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/Bee.cs	324
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/HoneyComb.cs	327
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/Item.cs	328
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/ItemGame↔ Object.cs	332
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/obj/Debug/Temporary↔ GeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs	334
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/obj/Debug/Temporary↔ GeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs	334
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/obj/Debug/Temporary↔ GeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs	334
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/PlayerLook.cs	334
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/PlayerMove.cs	335
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/SavePlayer↔ Position.cs	337
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/Selector.cs	337
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Properties/Assembly↔ Info.cs	339
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Quest/QuestBook.cs	340
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Resources/Resources.↔ Designer.cs	340
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Serialization/Serialization.↔ cs	342

C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/ Chunk↔ WorldPos.cs	354
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/ Chunk.↔ cs	346
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/ Load↔ Chunks.cs	349
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/ Mesh↔ Data.cs	352
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/ Save↔ Chunk.cs	353
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/LandGeneration/ Terrain.↔ cs	359
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/LandGeneration/ Terrain↔ Generation.cs	362
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/LandGeneration/ World.↔ cs	364
C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/LandGeneration/ ↔ Noise/ SimplexNoise.cs	355

5 Namespace Documentation

5.1 BeeGame Namespace Reference

Namespaces

- namespace [Blocks](#)
- namespace [Core](#)
- namespace [Exceptipns](#)
- 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 [Wood](#)

5.3 BeeGame.Core Namespace Reference

Namespaces

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

Classes

- class [Events](#)
- 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.Dictionarys Namespace Reference

Classes

- class [BeeCombinationDictionaryEqualityComparer](#)
- class [BeeDictionarys](#)
- class [CraftingRecipies](#)
- 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.POSION](#) }
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	
POSITION	

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

```
00056      {  
00057          NONE, POSITION  
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,
          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	

Enumerator

UNWEARY	
INDUSTRIOUS	
ICY	
GLACIAL	
NOBLE	
IMPERIAL	
MAJESTIC	
MIRY	
BOGGY	
HERIOC	
PHANTASMAL	
SPECTRAL	
HERMETIC	
SECLUDED	
SINISTER	
FIENDISH	
DEMONIC	
FRUGAL	
AUSTER	
VINDICTIVE	
EXOTIC	
ENDEMIC	
VENGEFUL	
AVENGING	
SETADFAST	
HEROIC	

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

```
00016     {  
00017         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, ENDEMIC, VENGEFUL, AVENGING,  
SETADFAST, HEROIC  
00018     };
```

5.5.1.6 BeeTempPreference

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

The different bee temp preferences

Enumerator

FROZEN	
COLD	
TEMPERATE	
HOT	
Generated by Doxygen	

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.Exceptipns Namespace Reference

Classes

- class [CraftingRecipieAdditionException](#)
- class [InputException](#)

5.8 BeeGame.Inventory Namespace Reference

Namespaces

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

Classes

- class [ApiaryInventory](#)
Inventory for Apiarys Apiary
- class [ChestInventory](#)
Incentory for the chests
- class [Inventory](#)
Base class for all inventorys 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

5.9 BeeGame.Inventory.BlockInventory Namespace Reference

Classes

- class [CraftingTableInventory](#)
Invnetory 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 [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 inity gameobjects
- class [NormalBee](#)
- class [QueenBee](#)
- 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 postion
- class [Selector](#)
Moves the Block selector

5.13 BeeGame.Quest Namespace Reference

Classes

- class [QuestBook](#)

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

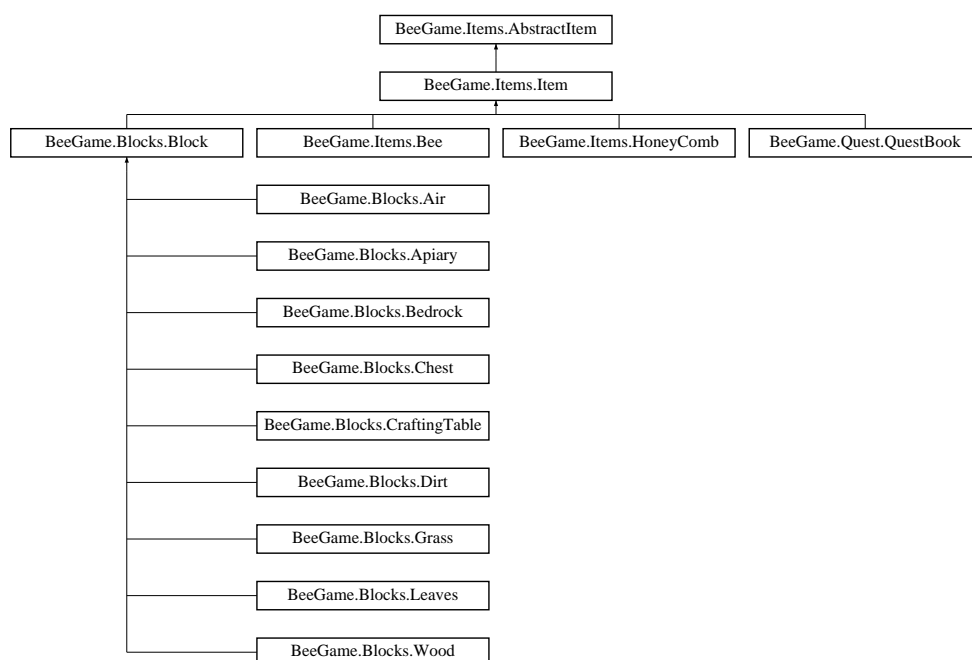
- class [SimplexNoise](#)
Implementation of the Perlin simplex noise, an improved Perlin noise algorithm. Based loosely on SimplexNoise1234 by Stefan Gustavson <http://staffwww.itn.liu.se/~stegu/aqsis/aqsis-newnoise/>

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.Blocks.Apiary](#), [BeeGame.Items.HoneyComb](#), [BeeGame.Blocks.Grass](#), [BeeGame.Blocks.Bedrock](#), [BeeGame.Blocks.Air](#), [BeeGame.Blocks.Dirt](#), [BeeGame.Blocks.Leaves](#), [BeeGame.Blocks.Wood](#), and [BeeGame.Quest.QuestBook](#).

6.1.2.2 GetItemID()

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

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

6.1.2.3 GetItemName()

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

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

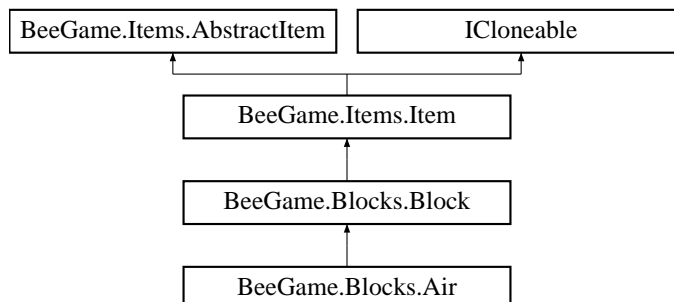
The documentation for this class was generated from the following file:

- 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 chesk 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](#).

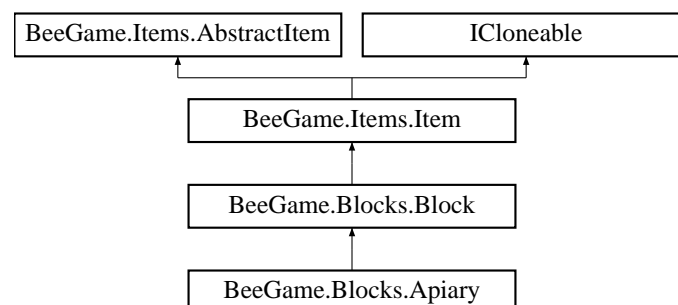
The documentation for this class was generated from the following file:

- 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 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);
00077                 myGameObject.GetComponent<ChestInventory>().inventoryPosition =
00078                 new THVector3(x, y, z) + chunk.chunkWorldPos;
00078                 myGameObject.GetComponent<ChestInventory>().SetChestInventory();
00079             }
00080             return base.BlockData(chunk, x, y, z, meshData, true);
00081         }

```

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);
00091             UnityEngine.Object.Destroy(myGameObject);
00092             /* removes the collision mesh from the chunk
00093             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>	Fist BeeEffect
<i>b2</i>	Second BeeEffect

Returns

A new BeeEffect

Definition at line 308 of file [Apiary.cs](#).

```
00309         {
00310             return (BeeEffect)ReturnChange((int)b1, (int)b2, (int)
00311                 BeeEffect.POSION);
00311         }
```

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>	Fist Bees fertility
<i>b2</i>	Second Bees fertility

Returns

A new fertility, uint

Definition at line 297 of file [Apiary.cs](#).

```
00298         {
00299             return (uint)ReturnChange((int)b1, (int)b2, 5, 1);
00300         }
```

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>	Fist BeeLifeSpan
<i>b2</i>	Second BeeLifeSpan

Returns

A new BeeLifeSpan

Definition at line 286 of file [Apiary.cs](#).

```

00287         {
00288             return (BeeLifeSpan)ReturnChange((int)b1, (int)b2, (int)
BeeLifeSpan.SEATURTLE);
00289         }

```

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>	Fist BeeProductionSpeed
<i>b2</i>	Second BeeProductionSpeed

Returns

A new BeeProductionSpeed

Definition at line 319 of file [Apiary.cs](#).

```

00320         {
00321             return (BeeProductionSpeed)ReturnChange((int)b1, (int)b2, (int)
BeeProductionSpeed.FAST);
00322         }

```

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 239 of file [Apiary.cs](#).

```

00240     {
00241         BeeSpecies[] possibleSpecies = BeeDictionarys.
GetCombinations(s1, s2);
00242         float[] weights = possibleSpecies.Length > 2 ? BeeDictionarys.
GetWeights(possibleSpecies) : new float[] { 0.5f, 0.5f };
00243
00244         var randomNum = Rand(weights);
00245         var weightsSum = 0f;
00246
00247         /* when the number generated is less than the current sum of the weights return that bee
00248         for (int i = 0; i < weights.Length; i++)
00249         {
00250             if(randomNum <= weightsSum)
00251             {
00252                 return possibleSpecies[i];
00253             }
00254
00255             weightsSum += weights[i];
00256         }
00257
00258         /* if for some reason the weights cannot work return the first bee in the combination list
00259         return possibleSpecies[0];
00260     }

```

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 102 of file [Apiary.cs](#).

```
00103         {
00104             return ID;
00105         }
```

6.3.3.10 InteractWithBlock()

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

Toggles the ApiaryInventory for the block

Parameters

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

Returns

Definition at line 122 of file [Apiary.cs](#).

```
00123         {
00124             myGameobject.GetComponent<ApiaryInventory>().myblock = this;
00125             myGameobject.GetComponent<ApiaryInventory>().ToggleInventory(inv);
00126             return true;
00127         }
```

6.3.3.11 MakeBee()

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

Nakes a new Bee

Parameters

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

Returns

A new Bee

Definition at line 208 of file [Apiary.cs](#).

```

00209         {
00210             /** gives all of the primary and secondary stats to the bee
00211             NormalBee nb = new NormalBee()
00212             {
00213                 pSpecies = CombineSpecies(queen.queen.
00214                 sSpecies, queen.drone.sSpecies),
00215                 sSpecies = CombineSpecies(queen.queen.
00216                 sSpecies, queen.drone.sSpecies),
00217                 pEffect = CombineEffect(queen.queen.sEffect, queen.
00218                 drone.sEffect),
00219                 sEffect = CombineEffect(queen.queen.sEffect, queen.
00220                 drone.sEffect),
00221                 pFertility = CombineFertility(queen.queen.
00222                 sFertility, queen.drone.sFertility),
00223                 sFertility = CombineFertility(queen.queen.
00224                 sFertility, queen.drone.sFertility),
00225                 pLifespan = CombineLifespan(queen.queen.
00226                 sLifespan, queen.drone.sLifespan),
00227                 sLifespan = CombineLifespan(queen.queen.
00228                 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             /** returns the new bee
00235             return new Bee(beeType, nb);
00236         }

```

6.3.3.12 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 necisseraly 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 invetory directly

Definition at line 138 of file [Apiary.cs](#).

```

00139         {
00140             Item[] producedItems = new Item[9];
00141
00142             /** will always return a new princess and drone
00143             queenBee); producedItems[0] = MakeBee(BeeType.PRINCESS, queen.
00144             queenBee); producedItems[1] = MakeBee(BeeType.DRONE, queen.
00145
00146             var repeats = UnityEngine.Random.Range(0, queen.queenBee.
00147             queen.pFertility);
00148
00149             /** produces as many other children as the bee staats will allow
00150             for (int i = 0; i < repeats; i++)
00151             {
00152                 producedItems[i + 2] = MakeBee(queen.queenBee.
00153             queen.pFertility > 6 ? (BeeType)UnityEngine.Random.Range(1, 3) :
00154             BeeType.DRONE, queen.queenBee);
00155
00156             if (producedItems[i + 2] is Bee b && b.beeType !=
00157             BeeType.PRINCESS)
00158                 producedItems[i + 2].itemStackCount =
00159             UnityEngine.Random.Range(1, (int)queen.queenBee.queen.
00160             pFertility + 1);
00161             }
00162
00163             /** gets the produced items
00164             var beeProduce = BeeDictionary.s.GetBeeProduce(queen.
00165             queenBee.queen.pSpecies);
00166
00167             /** chnages the stack count of the produced items to the correct number
00168             for (int i = 0; i < beeProduce.Length; i++)
00169             {
00170                 beeProduce[i].itemStackCount += UnityEngine.Random.Range(1, (int)
00171             queen.queenBee.queen.sProdSpeed + 1);
00172             }
00173
00174             /** adds the itmes that the bee species produces into the procued item array
00175             for (int i = (int)queen.queenBee.queen.pFertility + 2, prod = 0; prod <
00176             beeProduce.Length; i++, prod++)
00177             {
00178                 producedItems[i] = beeProduce[prod];
00179             }
00180
00181             /** puts the items into the inventory
00182             for (int i = 0; i < 9; i++)
00183             {
00184                 if (inventory[i + 2] != null)
00185                 {
00186                     /** if the slot has the same item in it and it wont be more than the max stack out but
00187                     the new item into it
00188                     if (producedItems[i] == inventory[i + 2] && inventory[i + 2].
00189                     itemStackCount + 1 <= inventory[i + 2].maxStackCount)
00190                         inventory[i + 2].itemStackCount++;
00191                     else
00192                         /** otherwise find a new slot to put the item into
00193                         for (int j = i; j < (9 - i); j++)
00194                         {
00195                             if (inventory[j + 2] == null)
00196                             {
00197                                 inventory[j + 2] = producedItems[i];
00198                                 break;
00199                             }
00200                             else if (producedItems[i] == inventory[j + 2] && inventory[j + 2].
00201                     itemStackCount + 1 <= inventory[j + 2].maxStackCount)
00202                             {
00203                                 inventory[j + 2].itemStackCount++;
00204                                 break;
00205                             }
00206                         }
00207                     }
00208                 }
00209                 /** if the slot is empty put the item into it
00210                 else
00211                     inventory[i + 2] = producedItems[i];
00212             }
00213         }
00214     }

```


6.3.3.13 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 267 of file [Apiary.cs](#).

```
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     }
```

6.3.3.14 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 [mutationMultiplier](#), the value of wich is dictated by the apairy

Definition at line 335 of file [Apiary.cs](#).

```

00336         {
00337             /** b1 and b2 are checked for which one is bigger than the other here as the
00338             /** queen my have a lower stat the an the drone and the drone is always passed in second
00339             var change = UnityEngine.Random.Range(b1 < b2 ? b1 : b2, (b2 > b1 ? b2 : b1) + 2);
00340
00341             /** this will make it possible for the bees to mutate during combination of the stats are the
same
00342             /** it will also cause more random mutation more mimicing nature
00343             change += UnityEngine.Random.Range(-mutationMultiplier,
mutationMultiplier);
00344
00345             /** as all but on ef the stats are enums they have a min/max value so need to check that this
is not exceded
00346             if (change > maxChange)
00347                 change = maxChange;
00348             else if (minChange > change)
00349                 change = minChange;
00350
00351             return change;
00352
00353         }

```

6.3.3.15 TexturePosition()

```

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

```

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

Parameters

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

Returns

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

Returns a trnsaparent 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         }

```

6.3.3.16 ToString()

```

override string BeeGame.Blocks.Apiary.ToString ( )

```

The item name and ID as a string

Returns

A nicely formatted string

Definition at line 111 of file [Apiary.cs](#).

```

00112         {
00113             return $"{itemName} \nID: {GetItemID()}";
00114         }

```

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 mutationMultiplier

```
int BeeGame.Blocks.Apiary.mutationMultiplier
```

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](#).

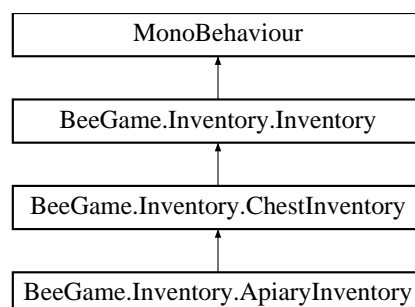
The documentation for this class was generated from the following file:

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

6.4 BeeGame.Inventory.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](#)
Sider 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

6.4.1 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](#).

6.4.2 Member Function Documentation

6.4.2.1 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 checked 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
.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             beesCombineing = true;
00107
00108             SaveInv();
00109
00110             /* set the slider max to the combination time
00111             timerSlideer.maxValue = combinationTime;
00112         }
00113     }
```

6.4.2.2 FixedUpdate()

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

Updates the combination time because of this was frame rate dependant and 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     }
```

6.4.2.3 SetChestInventory()

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

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

Parameters

<code>invName</code>	
----------------------	--

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

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

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

6.4.2.4 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
00042             if(items.itemsInInventory.Length > 0 && !
00043             beesCombineing)
00043             CheckforBees();
00044
00045             /* if the currently combineing bees has finished combineing
00046             if (combinationTime < 0 && beesCombineing)
00047             {
00048                 /* make the items that the bees should make and destroy the spent queen
00049                 ((Apiary)myblock).MakeBees(items.
00049                 itemsInInventory[0] as Items.Bee, ref items.
00050                 itemsInInventory);
00050                 beesCombineing = false;
00051                 items.itemsInInventory[0] = null;
00052
00053                 /* save the channages to the inventory
00054                 SaveInv();
00055             }
00056         }
```

6.4.3 Member Data Documentation

6.4.3.1 beesCombineing

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

Are bees currently combineing

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

6.4.3.2 combinationTime

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

How long does the current combineing bee have left

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

6.4.3.3 timerSlideer

```
Slider BeeGame.Inventory.ApiaryInventory.timerSlideer
```

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

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

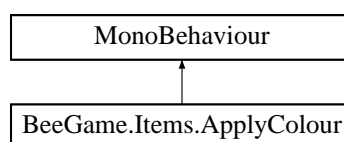
The documentation for this class was generated from the following file:

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

6.5 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 to

Private Member Functions

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

6.5.1 Detailed Description

Applies a given colour to a gameobject

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

6.5.2 Member Function Documentation

6.5.2.1 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         }  
00039     }
```

6.5.3 Member Data Documentation

6.5.3.1 colour

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

Colour to apply

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

6.5.3.2 objects

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

Objects to apply the colour to

Array set in the editor

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

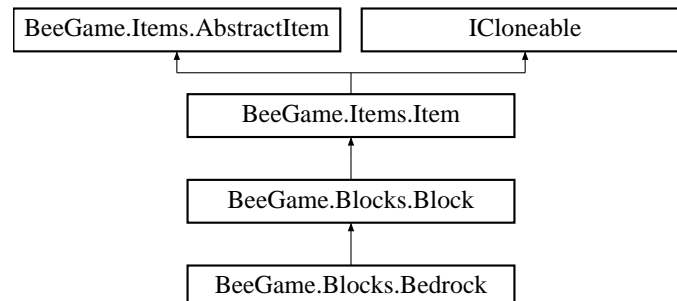
The documentation for this class was generated from the following file:

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

6.6 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 te 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

6.6.1 Detailed Description

Bedrock Block

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

6.6.2 Constructor & Destructor Documentation

6.6.2.1 Bedrock()

BeeGame.Blocks.Bedrock.Bedrock ()

Constructor

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

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

6.6.3 Member Function Documentation

6.6.3.1 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         }
```

6.6.3.2 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         }
```

6.6.3.3 TexturePosition()

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

Position of 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         }
```

6.6.3.4 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         }
```

6.6.4 Member Data Documentation

6.6.4.1 ID

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

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

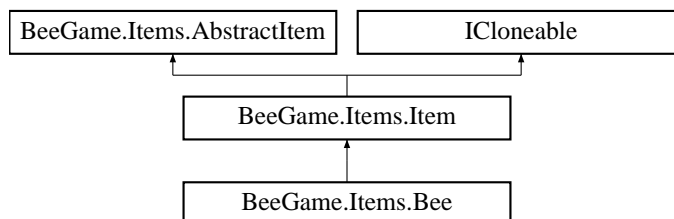
The documentation for this class was generated from the following file:

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

6.7 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 [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](#), [BeeEffect](#) effect=[BeeEffect.NONE](#), [BeeProductionSpeed](#) prodSpeed=[BeeProductionSpeed.NORMAL](#))
Make a bee with given stats
- override [int](#) [GetHashCode](#) ()
Returns the hashcode for this [Item](#)

Static Public Member Functions

- static void [ConvertToQueen](#) ([Bee](#) princess, [NormalBee](#) drone)
Will convert this bee to a [BeeType.QUEEN](#) using 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]
Overridden 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

6.7.1 Detailed Description

The bee item

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

6.7.2 Constructor & Destructor Documentation

6.7.2.1 Bee() [1/3]

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

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

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

6.7.2.2 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 69 of file [Bee.cs](#).

```

00069                                     : base(new CultureInfo("en-US", false).TextInfo.
    ToTitleCase($"{normalBee.pSpecies} {beeType}".ToLower()))
00070     {
00071         if (beeType == BeeType.PRINCESS || beeType ==
    BeeType.QUEEN)
00072             maxStack = 1;
00073             this.beeType = beeType;
00074             this.normalBee = normalBee;
00075     }

```

6.7.2.3 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 82 of file [Bee.cs](#).

```

00082                                     : base(new CultureInfo("en-US", false).TextInfo.
    ToTitleCase($"{queenBee.queen.pSpecies} {beeType}".ToLower()))
00083     {
00084         if (beeType == BeeType.PRINCESS || beeType ==
    BeeType.QUEEN)
00085             maxStack = 1;
00086             this.beeType = beeType;
00087             this.queenBee = queenBee;
00088     }

```

6.7.3 Member Function Documentation

6.7.3.1 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>	
--------------	--

Definition at line 142 of file [Bee.cs](#).

```
00143         {
00144             ConvertToQueen(ref princess, drone);
00145         }
```

6.7.3.2 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 152 of file [Bee.cs](#).

```
00153         {
00154             princess.beeType = BeeType.QUEEN;
00155             princess.queenBee = new QueenBee(princess.normalBee, drone);
00156             princess.normalBee = null;
00157
00158             princess.itemName = new CultureInfo("en-US", false).TextInfo.ToTitleCase($"
{princess.queenBee.queen.pSpecies} {princess.beeType}".ToLower());
00159         }
```

6.7.3.3 GetHashCode()

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

Retuens the hashcode for this [Item](#)

Returns

9

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

Definition at line 202 of file [Bee.cs](#).

```
00203         {
00204             return ID;
00205         }
```

6.7.3.4 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 131 of file [Bee.cs](#).

```
00132         {
00133             return $"{GetHashCode()}\\{(int)beeType}{queenBee?.GetHashCode() ?? normalBee?.GetHashCode()}";
00134         }
```

6.7.3.5 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 96 of file [Bee.cs](#).

```
00097         {
00098             /* if the bee has not change in any way dont rebuild the sprite as that takes time
00099             if(previousBeeType == beeType && itemSprite != null)
00100             {
00101                 return itemSprite;
00102             }
00103
00104             previousBeeType = beeType;
00105
00106             /* set the correct sprite and colour
00107             if (beeType == BeeType.QUEEN)
00108             {
00109                 /* avoids the crown, black body, yellow body, and both colours of the wings
00110                 Color[] colorsToAvoid = { new Color(0, 0, 0), new Color(232f, 200f, 42f, 255f) / 255f, new
Color(232f, 213f, 106f, 255f) / 255f, new Color(156f, 146f, 130f, 255f) / 255f, new Color(225f, 223f, 219f, 2
255f) / 255f };
00111                 return itemSprite = SpriteDictionary.
GetSprite("Queen").ColourSprite(BeeDictionary.
GetBeeColour((BeeSpecies) (queenBee?.queen.
pSpecies)), coloursToAvoid: colorsToAvoid);
00112             }
00113             else if (beeType == BeeType.PRINCESS)
00114             {
00115                 /* avoids the tiara, black body, yellow body, and both colours of the wings
00116                 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
55f) / 255f, new Color(232f, 200, 42, 255f) / 255f };
00117                 return itemSprite = SpriteDictionary.
GetSprite("Princess").ColourSprite(BeeDictionary.
GetBeeColour((BeeSpecies) (normalBee?.pSpecies)), coloursToAvoid:
colorsToAvoid);
00118             }
00119             else
00120             {
00121                 /* avoids the block body, yellow body, and both wing colours
00122                 Color[] colorsToAvoid = { new Color(0, 0, 0), new Color(156f, 146f, 130f, 255f) / 255f, new
Color(225f, 223f, 219f, 255f) / 255f, new Color(232f, 200, 42, 255f) / 255f };
00123                 return itemSprite = SpriteDictionary.
GetSprite("Drone").ColourSprite(BeeDictionary.
GetBeeColour((BeeSpecies) normalBee?.pSpecies), coloursToAvoid:
colorsToAvoid);
00124             }
00125         }
```


6.7.3.6 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 171 of file [Bee.cs](#).

```

00172         {
00173             NormalBee normBee = new NormalBee()
00174             {
00175                 pSpecies = species,
00176                 pLifespan = lifespan,
00177                 pFertility = fertility,
00178                 pProdSpeed = prodSpeed,
00179                 pEffect = effect,
00180                 sEffect = effect,
00181                 sFertility = fertility,
00182                 sLifespan = lifespan,
00183                 sProdSpeed = prodSpeed,
00184                 sSpecies = species
00185             };
00186
00187             switch (beeType)
00188             {
00189                 case BeeType.QUEEN:
00190                     return new Bee(beeType, new QueenBee(normBee, normBee));
00191                 default:
00192                     return new Bee(beeType, normBee);
00193             }
00194         }

```

6.7.4 Member Data Documentation

6.7.4.1 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](#).

6.7.4.2 ID

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

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

6.7.4.3 itemSprite

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

This bees Sprite

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

6.7.4.4 maxStack

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

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

6.7.5 Property Documentation

6.7.5.1 beeType

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

This bees BeeType

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

6.7.5.2 maxStackCount

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

Overriden so can be set

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

6.7.5.3 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](#).

6.7.5.4 previousBeeType

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

What was this bees `BeeType`?

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

6.7.5.5 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](#)

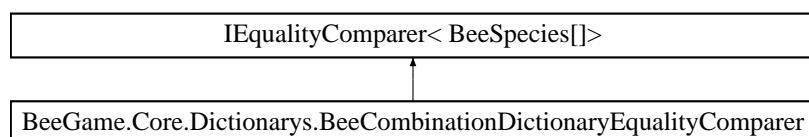
Definition at line 48 of file [Bee.cs](#).

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

- `C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/Bee.cs`

6.8 BeeGame.Core.Dictionarys.BeeCombinationDictionaryEqualityComparer Class Reference

Inheritance diagram for `BeeGame.Core.Dictionarys.BeeCombinationDictionaryEqualityComparer`:



Public Member Functions

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

6.8.1 Detailed Description

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

6.8.2 Member Function Documentation

6.8.2.1 Equals()

```
bool BeeGame.Core.Dictionarys.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             /* if 1 means both y elements are the same so no combination has been found
00020             if(y.Intersect(x).Count() <= 1)
00021                 return false;
00022             return true;
00023         }
00024     }
00025     return false;
00026 }
00027 }
```

6.8.2.2 GetHashCode()

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

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

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

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

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

6.9 BeeGame.Core.Dictionarys.BeeDictionarys 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 coumb

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 , optionally 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 > [honeyCoumbColour](#)

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

6.9.1 Detailed Description

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

6.9.2 Member Function Documentation

6.9.2.1 CombColour()

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

Makes a new colour given Red, r , Green, g , Blue, b , optionally 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 118 of file [BeeDictionary.cs](#).

```
00119      {
00120          return new Color(r / 255f, g / 255f, b / 255f);
00121      }
```

6.9.2.2 GetBeeColour()

```
static Color BeeGame.Core.Dictionarys.BeeDictionarys.GetBeeColour (
    BeeSpecies species ) [static]
```

Definition at line 91 of file [BeeDictionary.cs](#).

```
00092      {
00093          beeColour.TryGetValue(species, out Color colour);
00094      }
00095      return colour != null ? colour : new Color();
00096  }
```

6.9.2.3 GetBeeProduce()

```
static Items.Item [] BeeGame.Core.Dictionarys.BeeDictionarys.GetBeeProduce (
    BeeSpecies species ) [static]
```

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

```
00076      {
00077          beeProduce.TryGetValue(species, out Items.Item[] produce);
00078      }
00079      /* of the produce cant be found then return a honey comb as it is probly a bug
00080      return produce ?? new Items.Item[1] { new Items.HoneyComb(
HoneyCombType.HONEY) };
00081  }
```

6.9.2.4 GetCombColour()

```
static Color BeeGame.Core.Dictionarys.BeeDictionarys.GetCombColour (
    HoneyCombType type ) [static]
```

Returns colour if the given honey couble

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 [honeyCoubColour](#) dictionary

Definition at line 128 of file [BeeDictionarys.cs](#).

```

00129         {
00130             honeyCoubColour.TryGetValue(type, out var temp);
00131
00132             if (temp == null)
00133                 return new Color(1, 0, 0);
00134             return temp;
00135         }
00136     }

```

6.9.2.5 GetCombinations()

```

static BeeSpecies [] BeeGame.Core.Dictionarys.BeeDictionarys.GetCombinations (
    BeeSpecies s1,
    BeeSpecies s2 ) [static]

```

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

```

00041         {
00042             var beeSpecies = new BeeSpecies[2] { s1, s2 };
00043             var returnBeeList = new List<BeeSpecies>();
00044
00045             var keys = beeCombinations.Keys.ToArray();
00046             var comparor = new BeeCombinationDictionaryEqualityComparer
00047         ();
00048
00049             for (int i = 0; i < keys.Length; i++)
00050             {
00051                 if (comparor.Equals(keys[i], beeSpecies))
00052                 {
00053                     var temp = beeCombinations[keys[i]];
00054                     for (int j = 0; j < temp.Length; j++)
00055                     {
00056                         returnBeeList.Add(temp[j]);
00057                     }
00058                 }
00059             }
00060
00061             returnBeeList.Add(s1);
00062             returnBeeList.Add(s2);
00063
00064             return returnBeeList.ToArray();
00065         }

```

6.9.2.6 GetWeights()

```
static float [] BeeGame.Core.Dictionarys.BeeDictionarys.GetWeights (
    BeeSpecies [] species ) [static]
```

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

```
00019     {
00020         var returnArray = new float[species.Length];
00021
00022         for (int i = 0; i < species.Length; i++)
00023         {
00024             if(beeCombinationWeights.ContainsKey(species[i]))
00025                 returnArray[i] = beeCombinationWeights[species[i]];
00026             else
00027                 returnArray[i] = 0.5f;
00028         }
00029
00030         return returnArray;
00031     }
```

6.9.3 Member Data Documentation

6.9.3.1 beeColour

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

Initial value:

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

Definition at line 85 of file [BeeDictionarys.cs](#).

6.9.3.2 beeCombinations

```
Dictionary<BeeSpecies[], BeeSpecies[]> BeeGame.Core.Dictionarys.BeeDictionarys.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 35 of file [BeeDictionarys.cs](#).

6.9.3.3 beeCombinationWeights

Dictionary<BeeSpecies, float> BeeGame.Core.Dictionarys.BeeDictionarys.beeCombinationWeights
[static], [private]

Initial value:

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

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

6.9.3.4 beeProduce

Dictionary<BeeSpecies, Items.Item[]> BeeGame.Core.Dictionarys.BeeDictionarys.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 69 of file [BeeDictionarys.cs](#).

6.9.3.5 honeyCoubColour

Dictionary<HoneyCombType, Color> BeeGame.Core.Dictionarys.BeeDictionarys.honeyCoubColour
[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 teh HoneyCombTypes

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

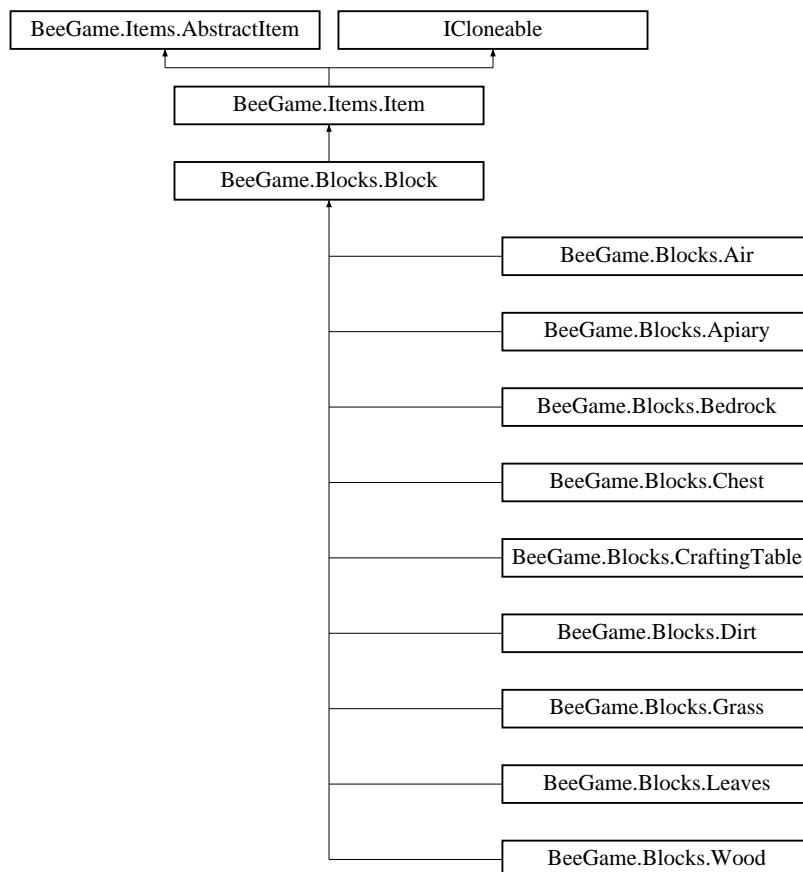
The documentation for this class was generated from the following file:

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

6.10 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](#) ()
Hascode 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

6.10.1 Detailed Description

Base class for blocks

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

6.10.2 Constructor & Destructor Documentation

6.10.2.1 [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     }
```

6.10.2.2 [Block\(\)](#) [2/2]

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

Sets placeabel 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     }
```

6.10.3 Member Function Documentation**6.10.3.1 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 collsion 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 givn mesh.
If only collsion 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
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     }

```

6.10.3.2 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("
Prefabs/ItemGameObject") as GameObject, pos, Quaternion.identity) as GameObject;
00065         go.GetComponent<ItemGameObject>().item = this;
00066     }

```

6.10.3.3 GetHashCode()

```

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

```

Hascode 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.Leaves](#), and [BeeGame.Blocks.Wood](#).

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

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

6.10.3.4 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.Grass](#), [BeeGame.Blocks.Dirt](#), [BeeGame.Blocks.Wood](#), and [BeeGame.Blocks.Leaves](#).

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

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

6.10.3.5 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      }
```

6.10.3.6 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.Air](#), and [BeeGame.Blocks.Leaves](#).

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

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

6.10.3.7 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         }
```

6.10.3.8 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 { }
```

6.10.4 Member Data Documentation

6.10.4.1 breakable

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

Can this [Block](#) be broken

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

6.10.4.2 changed

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

Has this block been placed by the player

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

6.10.4.3 ID

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

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

6.10.4.4 placeable

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

Sets so that blocks can be placed

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

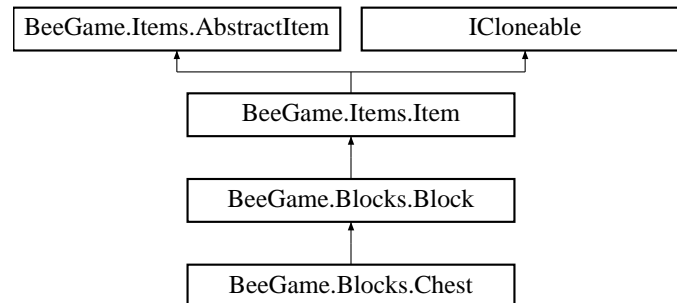
The documentation for this class was generated from the following file:

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

6.11 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 game 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 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

6.11.1 Detailed Description

Chest Block

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

6.11.2 Constructor & Destructor Documentation

6.11.2.1 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     }
```

6.11.3 Member Function Documentation

6.11.3.1 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
<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(
PrefabDictionary.GetPrefab("Chest"), new THVector3(x, y, z) + chunk.
chunkWorldPos, Quaternion.identity, chunk.transform);
00079                 myGameObject.GetComponent<ChestInventory>().inventoryPosition =
new THVector3(x, y, z) + chunk.chunkWorldPos;
00080                 myGameObject.GetComponent<ChestInventory>().SetChestInventory();
00081             }
00082             return base.BlockData(chunk, x, y, z, meshData, true);
00083         }

```

6.11.3.2 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<
ChestInventory>().inventoryName);
00093             UnityEngine.Object.Destroy(myGameObject);
00094             /* removes the collision mesh from the chunk
00095             base.BreakBlock(pos);
00096         }

```

6.11.3.3 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         }

```

6.11.3.4 GetHashCode()

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

Gets the ID of the [Block](#)

Returns

8

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

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

```
00118      {  
00119          return ID;  
00120      }
```

6.11.3.5 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 interactiong with
------------	---

Returns

true

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

```
00106      {  
00107          myGameObject.GetComponent<ChestInventory>().ToggleInventory(inv);  
00108          return true;  
00109      }
```

6.11.3.6 TexturePosition()

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

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

Parameters

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

Returns

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

REturns a trnsparent 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         }
```

6.11.3.7 ToString()

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

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

Returns

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

```
00127         {
00128             return $"{itemName}\nID{GetItemID()}";
00129         }
```

6.11.4 Member Data Documentation**6.11.4.1 ID**

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

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

6.11.4.2 myGameObject

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

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

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

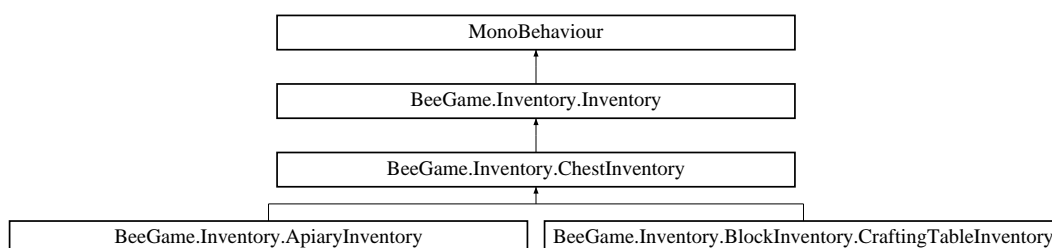
The documentation for this class was generated from the following file:

- `C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Chest.cs`

6.12 BeeGame.Inventory.ChestInventory Class Reference

Inventory 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 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`
Refernce 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

6.12.1 Detailed Description

Inventory for the chests

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

6.12.2 Member Function Documentation

6.12.2.1 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.
itemsInInventory.Length; i++)
00091             {
00092                 playerinventory.items.itemsInInventory[i] =
items.itemsInInventory[i + (inventorySize - 36)];
00093             }
00094
00095             playerinventory.SaveInv();
00096         }
```

6.12.2.2 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.Serialization.DeserializeInventory(this, inventoryName);
00071         }
```

6.12.2.3 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.
itemsInInventory.Length; i++)
00080     {
00081         items.itemsInInventory[i + (inventorySize - 36)] =
playerinventory.items.itemsInInventory[i];
00082     }
00083 }
```

6.12.2.4 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 immediately 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     }
```


6.12.2.5 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         }
```

6.12.2.6 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 &&  
floatingItem == null)  
00053                 ToggleInventory(playerinventory);  
00054         }
```

6.12.3 Member Data Documentation

6.12.3.1 inventory

```
GameObject BeeGame.Inventory.ChestInventory.inventory
```

The inventory gameobject that will be displayed

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

6.12.3.2 inventoryPosition

```
THVector3 BeeGame.Inventory.ChestInventory.inventoryPosition
```

Position in worldspace of the chest

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

6.12.3.3 inventorySize

```
int BeeGame.Inventory.ChestInventory.inventorySize
```

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

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

6.12.3.4 playerinventory

```
Inventory BeeGame.Inventory.ChestInventory.playerinventory
```

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

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

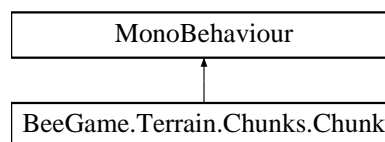
The documentation for this class was generated from the following file:

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

6.13 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 checkNeighbouringChunks=true)
Returns the Block in the given x, y, z
- void [SetBlock](#) (int x, int y, int z, [Block](#) block, bool checkNeighbouringChunks=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 [updateCollsionMesh](#) = 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 version 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 collidier

6.13.1 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](#).

6.13.2 Member Function Documentation

6.13.2.1 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 no verts dont do it as
their is no point
00240             if (this.mesh.verts.Count == 0)
00241                 return;
00242
00243         /** if the render and collision meshes should be shared set the render mesh to the collision
mesh otherwise make a collision mesh
00244             if (this.mesh.shareMeshes)
00245             {
00246                 world.chunkHasMadeCollisionMesh = true;
00247                 applyCollisionMesh = false;
00248                 meshCollider.sharedMesh = filter.mesh;
00249                 return;
00250             }
00251
00252             world.chunkHasMadeCollisionMesh = true;
00253             /** Applying the mesh takes the longest but nothing can be done with the mesh class in a
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     }

```

6.13.2.2 GetBlock()

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

Returns the Block in the given x, y, z

Parameters

<i>x</i>	X pos if the Block
<i>y</i>	Z pos if the Block
<i>z</i>	Y pos if the Block
<i>checkNeighbouringChunks</i>	Should this check neighbouring chunks? Only set to false when chunk mesh 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
           an air block (empty block)
00130         //if(checkNeighbouringChunks)
00131             return world.GetBlock(chunkWorldPos.x + x,
           chunkWorldPos.y + y, chunkWorldPos.z + z);
00132
00133         //return new Air();
00134     }

```

6.13.2.3 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     }

```

6.13.2.4 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
secondary thread...thanks unity
00216
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         /** redoes the normals incase they got messed up
00230         filter.mesh.RecalculateNormals();
00231         /** is this necessary as it causes alot of lag?
00232     }

```

6.13.2.5 SetBlock()

```

void BeeGame.Terrain.Chunks.Chunk.SetBlock (
    int x,
    int y,
    int z,
    Block block,
    bool checkNeighbouringChunks = 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 (checkNeighbouringChunks)
00153             /** if the block is not in the chunk find its chunk and set it there
00154             world.SetBlock(chunkWorldPos.x + x,
chunkWorldPos.y + y, chunkWorldPos.z + z, block);
00155     }

```

6.13.2.6 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     }
```

6.13.2.7 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     }
```

6.13.2.8 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     }
```

6.13.2.9 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 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     }
```

6.13.3 Member Data Documentation

6.13.3.1 applyCollisionMesh

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

Should the collision mesh be applied

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

6.13.3.2 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](#).

6.13.3.3 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 reduceing the number breaks things TODO: find

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

6.13.3.4 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`.

6.13.3.5 filter

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

This `Chunks` mesh filter

Definition at line 66 of file `Chunk.cs`.

6.13.3.6 mesh

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

`MeshData` of this chunk

Definition at line 61 of file `Chunk.cs`.

6.13.3.7 meshCollider

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

This `Chunks` mesh collider

Definition at line 70 of file `Chunk.cs`.

6.13.3.8 rendered

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

Is the `Chunk` rendered?

Definition at line 38 of file `Chunk.cs`.

6.13.3.9 update

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

Should the [Chunk](#) be updated?

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

6.13.3.10 updateCollisionMesh

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

Should the chunks collision mesh be updated?

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

6.13.3.11 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](#)

6.14 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`

6.14.1 Detailed Description

Serializable int version of THVector3

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

6.14.2 Constructor & Destructor Documentation

6.14.2.1 ChunkWorldPos()

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

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

Parameters

<code>x</code>	X Value
<code>y</code>	Y Value
<code>z</code>	Z Value

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

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

6.14.3 Member Function Documentation

6.14.3.1 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     }

```

6.14.3.2 GetHashCode()

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

Makes a unique hascode for the vector

Returns

unique int value for the vector

Possible that 2 defferent 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     }

```

6.14.3.3 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);
    }

```

6.14.3.4 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     }

```

6.14.3.5 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     }

```

6.14.4 Member Data Documentation

6.14.4.1 x

```

int BeeGame.Terrain.ChunkWorldPos.x

```

x, y, z values for the vector

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

6.14.4.2 y

```
int BeeGame.Terrain.ChunkWorldPos.y
```

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

6.14.4.3 z

```
int BeeGame.Terrain.ChunkWorldPos.z
```

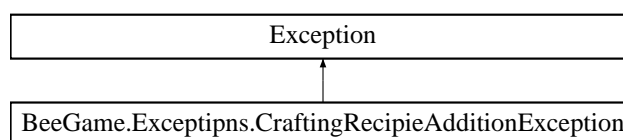
Definition at line 15 of file [ChunkWorldPos.cs](#).

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

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

6.15 BeeGame.Exceptipns.CraftingRecipieAdditionException Class Reference

Inheritance diagram for BeeGame.Exceptipns.CraftingRecipieAdditionException:



Public Member Functions

- [CraftingRecipieAdditionException](#) ()
- [CraftingRecipieAdditionException](#) (string message)
- [CraftingRecipieAdditionException](#) (string message, Exception innerException)

6.15.1 Detailed Description

Definition at line 8 of file [CraftingRecipieAdditionException.cs](#).

6.15.2 Constructor & Destructor Documentation

6.15.2.1 CraftingRecipieAdditionException() [1/3]

```
BeeGame.Exceptipns.CraftingRecipieAdditionException.CraftingRecipieAdditionException ( )
```

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

```

00010                                     : base ()
00011     {
00012
00013     }
```

6.15.2.2 CraftingRecipieAdditionException() [2/3]

```
BeeGame.Exceptipns.CraftingRecipieAdditionException.CraftingRecipieAdditionException (
    string message )
```

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

```
00015                                     : base(message)
00016     {
00017
00018     }
```

6.15.2.3 CraftingRecipieAdditionException() [3/3]

```
BeeGame.Exceptipns.CraftingRecipieAdditionException.CraftingRecipieAdditionException (
    string message,
    Exception innerException )
```

Definition at line 20 of file [CraftingRecipieAdditionException.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/Exceptipns/[CraftingRecipieAdditionException.cs](#)

6.16 BeeGame.Core.Dictionarys.CraftingRecipies Class Reference

Static Public Member Functions

- static void [AddShapedRecipie](#) (object[] recipie, [Item](#) result)
Will add a shaped crafting recipie to the game
- static [Item](#) [GetShapedRecipeltem](#) (string recipie)
Returns an Item from the [shapedCraftingRecipies](#) dictionary
- static void [AddShaplessRecipie](#) (object[] recipie, [Item](#) result)
Adds a Shapless recipie to the dictionary
- static string [GetShaplessRecipieString](#) ([Item](#)[] recipie)
Gets a shapless recipie string from a given recipie
- static [Item](#) [GetShaplessRecipieResult](#) (int[] recipie)
Trys to get a shapless recipie
- static [Item](#) [GetShaplessRecipieResult](#) (string recipie)
Trys to get a shapless recipie
- static [Item](#) [GetShaplessRecipieResult](#) ([Item](#)[] recipie)
Trys to get a shapless recipie

Static Private Attributes

- static Dictionary< string, [Item](#) > [shapedCraftingRecipies](#) = new Dictionary<string, [Item](#)>()
Contains all crafting recipies that require a certian layout in the crafting grid ([Blocks.CraftingTable](#)
- static Dictionary< string, [Item](#) > [shaplessRecipies](#)
All shapless recipies

6.16.1 Detailed Description

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

6.16.2 Member Function Documentation

6.16.2.1 AddShapedRecipie()

```
static void BeeGame.Core.Dictionarys.CraftingRecipies.AddShapedRecipie (
    object [] recipie,
    Item result ) [static]
```

Will add a shaped crafting recipe to the game

Parameters

<i>recipie</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 haracters to a 9 character string
00035         var stringRecipie = "";
00036
00037         for (int i = 0; i < 3; i++)
00038         {
00039             stringRecipie += recipie[i] as string;
00040         }
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, $"{itemID.ToString()}:");
00050     }
00051
00052     /** converts empty sots " " into "0:"
00053     stringRecipie = stringRecipie.Replace(" ", "0:");
00054
00055     /** if the recipe exists an exception is thrown as two recipies cannot be the same
00056     if (shapedCraftingRecipies.ContainsKey(stringRecipie))
00057         throw new CraftingRecipeAdditionException($"Shaped Recipe
already exists: {stringRecipie}");
00058
00059     /** adds the recipe to the dictionary
00060     shapedCraftingRecipies.Add(stringRecipie, result);
00061 }

```

6.16.2.2 AddShaplessRecipe()

```

static void BeeGame.Core.Dictionarys.CraftingRecipies.AddShaplessRecipe (
    object [] recipie,
    Item result ) [static]

```

Adds a Shapless recipe to the dictionary

Parameters

<i>recipie</i>	Recipe to add. Format as { Item, Number of items }
<i>result</i>	Result of the crafting recipe

2 Examples of adding a shapless recipe

```

void Main()
{
    CraftingRecipies.AddShaplessRecipe(new object[] { new Dirt(), 2 }, new Grass());
}

void Main()
{
    CraftingRecipies.AddShaplessRecipe(new object[] { new Stone(), 3, new Wood(), 3 }, new Apiary());
}

```

Definition at line 106 of file [CraftingRecipies.cs](#).

```

00107     {
00108         var itemList = new List<int>();
00109         var stringRecpie = "";
00110
00111         for (int i = 0; i < recipie.Length; i+=2)
00112         {
00113             for (int j = 0; j < (int)recipie[i+1]; j++)
00114             {
00115                 itemList.Add(int.Parse(((Item) recipie[i]).GetItemID()));
00116             }
00117         }
00118
00119         itemList.Sort();
00120
00121         for (int i = 0; i < itemList.Count; i++)
00122         {
00123             stringRecpie += $"{itemList[i]}:";
00124         }
00125
00126         if (shaplessRecipies.ContainsKey(stringRecpie))
00127             throw new CraftingRecipeAdditionException($"Shaped Recipe
already exists: {stringRecpie}");
00128
00129         shaplessRecipies.Add(stringRecpie, result);
00130     }

```

6.16.2.3 GetShapedRecipeltem()

```
static Item BeeGame.Core.Dictionarys.CraftingRecipies.GetShapedRecipeItem (
    string recipie ) [static]
```

Returns an Item from the [shapedCraftingRecipies](#) dictionary

Parameters

<i>recipie</i>	Recipie for Item
----------------	------------------

Returns

An Item or null is recipie was not found

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

```
00069      {
00070          shapedCraftingRecipies.TryGetValue(recipie, out var item);
00071      }
00072      return item;
00073  }
```

6.16.2.4 GetShaplessRecipeResult() [1/3]

```
static Item BeeGame.Core.Dictionarys.CraftingRecipies.GetShaplessRecipeResult (
    int [] recipie ) [static]
```

Trys to get a shapless recipe

Parameters

<i>recipie</i>	Recipie to get
----------------	----------------

Returns

Item for the recipie, null if recipie does not exist

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

```
00167      {
00168          var list = recipie.ToList();
00169          list.Sort();
00170      }
00171      var stringRecipe = "";
00172      for (int i = 0; i < list.Count; i++)
00173      {
00174          stringRecipe += $"{list[i]}:";
00175      }
00176      return GetShaplessRecipeResult(stringRecipe);
00177  }
```

6.16.2.5 GetShaplessRecipieResult() [2/3]

```
static Item BeeGame.Core.Dictionarys.CraftingRecipes.GetShaplessRecipieResult (
    string recipie ) [static]
```

Trys to get a shapless recipie

Parameters

<i>recipie</i>	Recipie to get
----------------	----------------

Returns

Item for the recipie, null if recipie does not exist

Definition at line 186 of file [CraftingRecipes.cs](#).

```
00187     {
00188         shaplessRecipes.TryGetValue(recipie, out var item);
00189
00190         return item;
00191     }
```

6.16.2.6 GetShaplessRecipieResult() [3/3]

```
static Item BeeGame.Core.Dictionarys.CraftingRecipes.GetShaplessRecipieResult (
    Item [] recipie ) [static]
```

Trys to get a shapless recipie

Parameters

<i>recipie</i>	Recipie to get
----------------	----------------

Returns

Item for the recipie, null if recipie does not exist

Definition at line 198 of file [CraftingRecipes.cs](#).

```
00199     {
00200         shaplessRecipes.TryGetValue(
00201             GetShaplessRecipieString(recipie), out var item);
00202         return item;
00203     }
```

6.16.2.7 GetShaplessRecipieString()

```
static string BeeGame.Core.Dictionarys.CraftingRecipes.GetShaplessRecipieString (
    Item [] recipie ) [static]
```

Gets a shapless recipie string from a given recipie

Parameters

<i>recipie</i>	Recipie for string
----------------	--------------------

Returns

A string of the given shapless recipie

Definition at line 137 of file [CraftingRecipies.cs](#).

```

00138     {
00139         var IDList = new List<int>();
00140         var stringRecipe = "";
00141
00142         /* converts tthe given item list to an ID list so it can be sorted
00143         for (int i = 0; i < recipie.Length; i++)
00144         {
00145             if(recipie[i] != null)
00146                 IDList.Add(recipie[i].GetHashCode());
00147         }
00148
00149         IDList.Sort();
00150
00151         /* converts the sorted ID list to a string so can be used as a dictionary key
00152         for (int i = 0; i < IDList.Count; i++)
00153         {
00154             /* : after each ID as it is possible for ID clashes without eg ID: 11 can be seen as 2 *
ID: 1
00155             stringRecipe += $"{IDList[i]}:";
00156         }
00157
00158         return stringRecipe;
00159     }

```

6.16.3 Member Data Documentation**6.16.3.1 shapedCraftingRecipies**

```
Dictionary<string, Item> BeeGame.Core.Dictionarys.CraftingRecipies.shapedCraftingRecipies =
new Dictionary<string, Item>() [static], [private]
```

Contains all crafting recipies that require a certian layout in the crafting grid ([Blocks.CraftingTable](#)

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

6.16.3.2 shaplessRecipies

```
Dictionary<string, Item> BeeGame.Core.Dictionarys.CraftingRecipies.shaplessRecipies [static],
[private]
```

Initial value:

```
= new Dictionary<string, Item>()
{
}

```

All shapless recipies

Definition at line 80 of file [CraftingRecipies.cs](#).

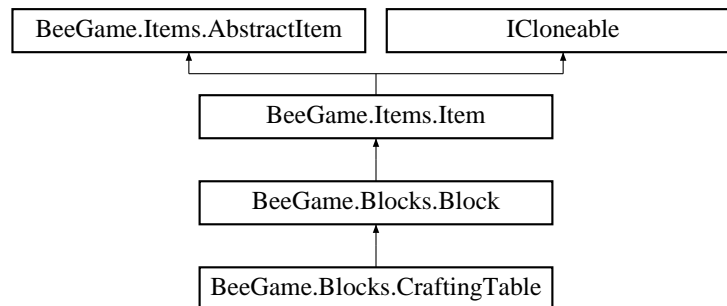
The documentation for this class was generated from the following file:

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

6.17 BeeGame.Blocks.CraftingTable Class Reference

The Workbench [Block](#) class

Inheritance diagram for BeeGame.Blocks.CraftingTable:



Public Member Functions

- [CraftingTable](#) ()
Constructor
- [Item ReturnShapedRecipeItem](#) ([Item](#)[] items)
Makes a shaped crafting recipe from the given items and return if it is a recipe
- virtual [Item ReturnShapelessRecipeItem](#) ([Item](#)[] items)
- virtual [Item ReturnShapedRecipeItem](#) (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 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 apiary [Block](#)
- override int [GetHashCode](#) ()
Returns the ID of the Item

Static Public Attributes

- static new int [ID](#) => 9
This blocks ID

Private Attributes

- [GameObject](#) [myGameobject](#)
The GameObject for this block

Additional Inherited Members

6.17.1 Detailed Description

The Workbench [Block](#) class

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

6.17.2 Constructor & Destructor Documentation

6.17.2.1 CraftingTable()

BeeGame.Blocks.CraftingTable.CraftingTable ()

Constructor

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

```
00034                                     : base("Workbench")
00035     {
00036         usesGameObject = true;
00037     }
```

6.17.3 Member Function Documentation

6.17.3.1 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

<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 colision mesh as the model is handled 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         }

```

6.17.3.2 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 131 of file [CraftingTable.cs](#).

```

00132         {
00133             /* removes the game object
00134             UnityEngine.Object.Destroy(myGameObject);
00135             /* removes the collision mesh from the chunk
00136             base.BreakBlock(pos);
00137         }

```

6.17.3.3 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         }

```

6.17.3.4 GetHashCode()

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

Returns the ID of the Item

Returns

ID

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

Definition at line 167 of file [CraftingTable.cs](#).

```
00168      {  
00169          return ID;  
00170      }
```

6.17.3.5 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 143 of file [CraftingTable.cs](#).

```
00144      {  
00145          return SpriteDictionary.GetSprite("TestSprite");  
00146      }
```

6.17.3.6 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>().
myblock = this;
00092             myGameObject.GetComponent<Inventory.BlockInventory.CraftingTableInventory>().
ToggleInventory(inv);
00093             return true;
00094         }

```

6.17.3.7 ReturnShapedRecipieItem() [1/2]

```

Item BeeGame.Blocks.CraftingTable.ReturnShapedRecipieItem (
    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         }

```

6.17.3.8 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 overridden by a different crafting system

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

```
00078         {
00079             return BeeGame.Core.Dictionarys.
CraftingRecipes.GetShapedRecipeItem(recipe);
00080         }
```

6.17.3.9 ReturnShapelessRecipeItem()

```
virtual Item BeeGame.Blocks.CraftingTable.ReturnShapelessRecipeItem (
    Item [] items ) [virtual]
```

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

```
00065         {
00066             return CraftingRecipes.GetShaplessRecipeResult (items)
;
00067         }
```

6.17.3.10 TexturePosition()

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

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

Parameters

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

Returns

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

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

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

Definition at line 156 of file [CraftingTable.cs](#).

```
00157     {  
00158         return new Tile() { x = 0, y = 9 };  
00159     }
```

6.17.4 Member Data Documentation

6.17.4.1 ID

```
new int BeeGame.Blocks.CraftingTable.ID => 9 [static]
```

This blocks ID

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

6.17.4.2 myGameObject

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

The GameObject for this block

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

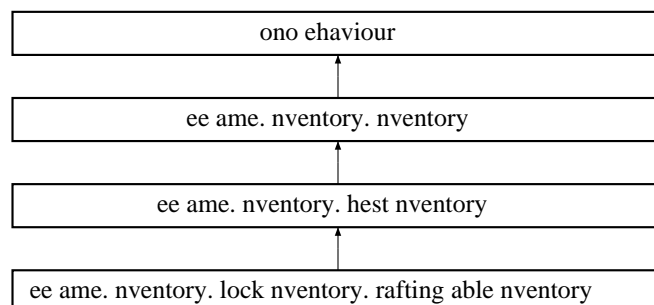
The documentation for this class was generated from the following file:

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

6.18 BeeGame.Inventory.BlockInventory.CraftingTableInventory Class Reference

Invnety for the CraftingTable Block

Inheritance diagram for BeeGame.Inventory.BlockInventory.CraftingTableInventory:



Public Member Functions

- delegate void [ItemRemovedFromResult](#) ()
Makes the delegate
- virtual void [CheckShapedRecipe](#) ()
Check in the recipe in the grid for a shaped crafting recipe
- virtual void [CheckShapelessRecipe](#) ()
Check in the recipe grid for a shapeless crafting recipe
- void [CraftedItemRemoved](#) ()
Removes the items from the crafting grid once an item has been removed from the crafting result slot, Called via the [result](#) 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](#) ()
Overridden so the inventory is not saved in any way

Public Attributes

- [ItemRemovedFromResult](#) result
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 recipes
- void [OnDestroy](#) ()
Ensuring no memory leaks occur due to the delegate

Additional Inherited Members

6.18.1 Detailed Description

Inventory for the CraftingTable Block

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

6.18.2 Member Function Documentation

6.18.2.1 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 179 of file [CraftingTableInventory.cs](#).

```
00180         {
00181             items.AddItem(slotIndex, item);
00182         }
```

6.18.2.2 CheckShapedRecipie()

```
virtual void BeeGame.Inventory.BlockInventory.CraftingTableInventory.CheckShapedRecipie ( )
[virtual]
```

Check in the recipie in the grid for a shaped crafting recipie

Definition at line 69 of file [CraftingTableInventory.cs](#).

```
00070         {
00071             var items = new Item[9];
00072
00073             for (int i = 0; i < items.Length; i++)
00074             {
00075                 items[i] = base.items.itemsInInventory[i];
00076             }
00077
00078             /* if it is a recipie put the result into the crafting result slot
00079             Item item = ((CraftingTable)myblock).ReturnShapedRecipieItem(items);
00080             if (item != base.items.itemsInInventory[9])
00081                 base.items.itemsInInventory[9] = item;
00082         }
```

6.18.2.3 CheckShapelessRecipie()

```
virtual void BeeGame.Inventory.BlockInventory.CraftingTableInventory.CheckShapelessRecipie ( )
[virtual]
```

Check in the recipie grid for a shapless crafting recipie

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

```
00088         {
00089             var items = new Item[9];
00090
00091             for (int i = 0; i < items.Length; i++)
00092             {
00093                 items[i] = base.items.itemsInInventory[i];
00094             }
00095
00096             Item item = ((CraftingTable)myblock).ReturnShapelessRecipieItem(items);
00097             if (item != base.items.itemsInInventory[9])
00098                 base.items.itemsInInventory[9] = item;
00099         }
```

6.18.2.4 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 [result](#) delegate from InventorySlot.OnPointerClick(UnityEngine.EventSystems.PointerEventData)

Definition at line 104 of file [CraftingTableInventory.cs](#).

```
00105         {
00106             if (items.itemsInInventory[9] != null)
00107             {
00108                 Events.CallShapedRecipieCraftedEvent (
00109                     items.itemsInInventory[9]);
00109                 for (int i = 0; i < 9; i++)
00110                 {
00111                     if (items.itemsInInventory[i] != null)
00112                         items.itemsInInventory[i].
00112                             itemStackCount -= 1;
00113                 }
00114             }
00115         }
```

6.18.2.5 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 125 of file [CraftingTableInventory.cs](#).

```
00126         {
00127             /* looks at every item in the crafting grid
00128             for (int i = 0; i < 9; i++)
00129             {
00130                 if (items.itemsInInventory[i] != null)
00131                 {
00132                     /* spwns it and removes it from the inventory if an items exists within
00133                     for (int j = 0; j < items.itemsInInventory[i].
00133                         itemStackCount; j++)
00134                     {
00135                         items.itemsInInventory[i].SpawnItem((
00135                         THVector3)this.transform.position + new THVector3(0, 1, 0));
00136                     }
00137                     items.itemsInInventory[i] = null;
00138                 }
00139             }
00140         }
```

6.18.2.6 ItemRemovedFromResult()

```
delegate void BeeGame.Inventory.BlockInventory.CraftingTableInventory.ItemRemovedFromResult (
)
```

Makes the delegate

6.18.2.7 OnDestroy()

```
void BeeGame.Inventory.BlockInventory.CraftingTableInventory.OnDestroy ( ) [protected]
```

Ensuring no memory leaks occur due to the delegate

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

```
00059     {
00060         /* just ensures no memory leaks occur
00061         result -= CraftedItemRemoved;
00062     }
```

6.18.2.8 SaveInv()

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

Oerriden so the inventory is not saved in any way

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

Definition at line 187 of file [CraftingTableInventory.cs](#).

```
00188     {
00189         /* does not need to be saved so overrided to do nothing
00190     }
```

6.18.2.9 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 attemp is made to deserialize the inventory helping with performance

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

Definition at line 164 of file [CraftingTableInventory.cs](#).

```
00165     {
00166         SetInventorySize(inventorySize);
00167         /* sets the UI to not be seen as inventorys cannot start open
00168         inventory.SetActive(false);
00169     }
```

6.18.2.10 Start()

```
void BeeGame.Inventory.BlockInventory.CraftingTableInventory.Start ( ) [protected]
```

Sets the size of the inventory

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

```
00032     {
00033         SetChestInventory();
00034         result = CraftedItemRemoved;
00035     }
```

6.18.2.11 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 148 of file [CraftingTableInventory.cs](#).

```
00149     {
00150         base.ToggleInventory(inv);
00151
00152         /* if the inventory was closed drop the items within
00153         if (!inventory.activeInHierarchy)
00154             DropItemsFromInventory();
00155     }
```

6.18.2.12 Update()

```
void BeeGame.Inventory.BlockInventory.CraftingTableInventory.Update ( ) [protected]
```

Updates the base and checks crafting recipies

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

```
00042     {
00043         UpdateChestInventory();
00044
00045         if (inventory.activeInHierarchy)
00046         {
00047             CheckShapedRecipe();
00048
00049             /* checks for shapless recipies second
00050             if(items.itemsInInventory[9] == null)
00051                 CheckShapelessRecipe();
00052         }
00053     }
```


6.18.3 Member Data Documentation

6.18.3.1 result

`ItemRemovedFromResult` `BeeGame.Inventory.BlockInventory.CraftingTableInventory.result`

Holds the method for the delegate to call

Definition at line 24 of file `CraftingTableInventory.cs`.

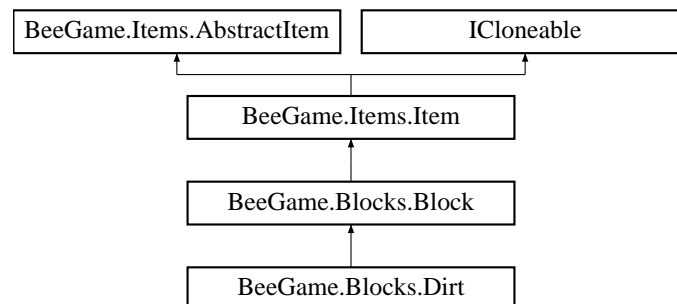
The documentation for this class was generated from the following file:

- `C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/BlockInventory/CraftingTableInventory.cs`

6.19 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

6.19.1 Detailed Description

Dirt Block

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

6.19.2 Constructor & Destructor Documentation

6.19.2.1 Dirt()

```
BeeGame.Blocks.Dirt.Dirt ( )
```

Constructor

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

```
00021 : base("Dirt") {}
```

6.19.3 Member Function Documentation

6.19.3.1 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     }
```

6.19.3.2 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      }
```

6.19.3.3 TexturePosition()

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

Position of the dirt texture in the atlas

Parameters

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

Returns

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

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

```
00038      {  
00039          return new Tile { x = 2, y = 9 };  
00040      }
```

6.19.3.4 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      }
```

6.19.4 Member Data Documentation

6.19.4.1 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](#)

6.20 BeeGame.Core.Events Class Reference

Public Member Functions

- delegate void [ItemCraftedEvent](#) ([Item](#) item)

Static Public Member Functions

- static void [CallShapedRecipieCraftedEvent](#) ([Item](#) item)
- static void [CallShaplessRecipirCraftedEvent](#) ([Item](#) item)
- static void [CallBeeCraftedEvent](#) ([Item](#) item)

Static Public Attributes

- static [ItemCraftedEvent](#) [shapedRecipieCrafted](#)
- static [ItemCraftedEvent](#) [shaplessRecipieCrafted](#)
- static [ItemCraftedEvent](#) [beeCraftedEvent](#)

6.20.1 Detailed Description

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

6.20.2 Member Function Documentation

6.20.2.1 CallBeeCraftedEvent()

```
static void BeeGame.Core.Events.CallBeeCraftedEvent (  
    Item item ) [static]
```

6.20.2.2 CallShapedRecipieCraftedEvent()

```
static void BeeGame.Core.Events.CallShapedRecipieCraftedEvent (
    Item item ) [static]
```

6.20.2.3 CallShaplessRecipirCraftedEvent()

```
static void BeeGame.Core.Events.CallShaplessRecipirCraftedEvent (
    Item item ) [static]
```

6.20.2.4 ItemCraftedEvent()

```
delegate void BeeGame.Core.Events.ItemCraftedEvent (
    Item item )
```

6.20.3 Member Data Documentation

6.20.3.1 beeCraftedEvent

```
ItemCraftedEvent BeeGame.Core.Events.beeCraftedEvent [static]
```

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

6.20.3.2 shapedRecipieCrafted

```
ItemCraftedEvent BeeGame.Core.Events.shapedRecipieCrafted [static]
```

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

6.20.3.3 shaplessRecipieCrafted

```
ItemCraftedEvent BeeGame.Core.Events.shaplessRecipieCrafted [static]
```

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

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

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

6.21 BeeGame.Core.Extensions Class Reference

Static Public Member Functions

- static T [CloneObject](#)< T > (this T obj)
Allows the copying of a class by value useing reflection
- static Sprite [ColourSprite](#) (this Sprite sprite, Color colour, Color[] coloursToAvoid=null, bool setTransparent↵ToWhite=false)
Will colour the sprite given a colour and optionally colours to avoid
- static void [SpawnItem](#) (this [Item](#) item, [THVector3](#) position, Quaternion rotation=new Quaternion())

6.21.1 Detailed Description

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

6.21.2 Member Function Documentation

6.21.2.1 CloneObject< T >()

```
static T BeeGame.Core.Extensions.CloneObject< T > (
    this T obj ) [static]
```

Allows the copying of a class by value useing reflection

Parameters

<i>obj</i>	Object to copy
------------	----------------

Returns

a new object with all values copyed

Mush faster than the serialize method however alot more complicated

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

```
00023     {
00024         /** gets the tyoe of the given object
00025         Type typeSource = obj.GetType();
00026
00027         /** makes a new object of type T
00028         T objTarget = (T)Activator.CreateInstance(typeSource);
00029
00030         /** gets the properties in T
00031         PropertyInfo[] propertyInfo = typeSource.GetProperties(BindingFlags.Public | BindingFlags.
NonPublic | BindingFlags.Instance);
00032
00033         /** applies the properties in T to the new type T object
00034         foreach (var property in propertyInfo)
00035         {
00036             if (property.CanWrite)
00037             {
00038                 /** if the propertly is a value just set it
```

```

00039         if (property.PropertyType.IsValueType || property.PropertyType.IsEnum || property.
PropertyType.Equals(typeof(string)))
00040         {
00041             property.SetValue(objTarget, property.GetValue(obj, null), null);
00042         }
00043         else
00044         {
00045             /* if the property is not a value type this function will need to be called
recursivly as it could also have non value type variables
00046             object propertyValue = property.GetValue(obj, null);
00047
00048             if (propertyValue == null)
00049             {
00050                 property.SetValue(objTarget, null, null);
00051             }
00052             else
00053             {
00054                 property.SetValue(objTarget, propertyValue.CloneObject(), null);
00055             }
00056         }
00057     }
00058 }
00059
00060     /* gets all of the field in T
00061     FieldInfo[] fieldInfo = typeSource.GetFields();
00062
00063     /* applies all of the fiels of T to the new object if type T in the same manor that the
properities are applied
00064     foreach (var field in fieldInfo)
00065     {
00066         if(field.FieldType.IsValueType || field.FieldType.IsEnum || field.FieldType.Equals(typeof(
string)))
00067         {
00068             field.SetValue(objTarget, field.GetValue(obj));
00069         }
00070         else
00071         {
00072             object fieldValue = field.GetValue(obj);
00073
00074             if(fieldValue == null)
00075             {
00076                 field.SetValue(objTarget, null);
00077             }
00078             else
00079             {
00080                 field.SetValue(objTarget, field.CloneObject());
00081             }
00082         }
00083     }
00084
00085     return objTarget;
00086 }

```

6.21.2.2 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 optional 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 96 of file [Extensions.cs](#).

```

00097     {
00098         Texture2D tex = new Texture2D((int)sprite.rect.width, (int)sprite.rect.height)
00099         {
00100             filterMode = FilterMode.Point,
00101             wrapMode = TextureWrapMode.Clamp
00102         };
00103
00104         /** sets the teture pixels to the pixels of teh sprite so the original sprite is not modified
00105         tex.SetPixels(sprite.texture.GetPixels());
00106
00107         for (int x = 0; x < tex.width; x++)
00108         {
00109             for (int y = 0; y < tex.height; y++)
00110             {
00111                 /** if we dont have to avoid any colours set the pixel
00112                 if (coloursToAvoid == null)
00113                 {
00114                     tex.SetPixel(x, y, tex.GetPixel(x, y) * colour);
00115                 }
00116                 else
00117                 {
00118                     for (int i = 0; i < coloursToAvoid.Length; i++)
00119                     {
00120                         /** if this colour should be avoided skip this iteration of the loop and move
00121                         on
00122                         if (tex.GetPixel(x, y) == coloursToAvoid[i])
00123                             goto Skip;
00124                     }
00125                     tex.SetPixel(x, y, tex.GetPixel(x, y) * colour);
00126                 }
00127
00128                 /** if transparent pixels should be set to white do that
00129                 if (setTransparentToWhite && tex.GetPixel(x, y).a == 0)
00130                     tex.SetPixel(x, y, Color.white);
00131
00132                 Skip:
00133                     continue;
00134             }
00135         }
00136
00137         /** apply the new texture with its colours
00138         tex.Apply();
00139
00140         /** return the Texture2D as a sprite
00141         return Sprite.Create(tex, new Rect(0, 0, tex.width, tex.height), new THVector2(0.5f, 0.5f));
00142     }

```

6.21.2.3 SpawnItem()

```

static void BeeGame.Core.Extensions.SpawnItem (
    this Item item,
    THVector3 position,
    Quaternion rotation = new Quaternion() ) [static]

```

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

```

00145     {
00146         GameObject go = MonoBehaviour.Instantiate(UnityEngine.Resources.Load("
Prefabs/ItemGameObject") as GameObject, position, rotation) as GameObject;
00147         go.GetComponent<ItemGameObject>().item = item;
00148     }

```

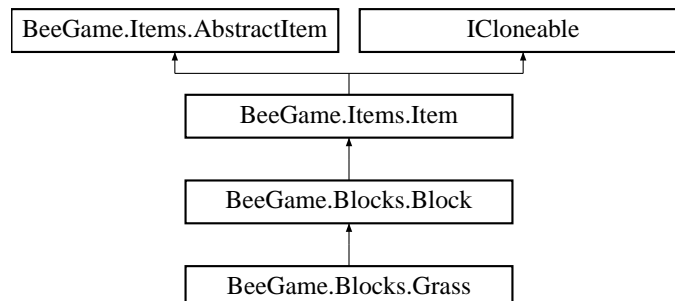
The documentation for this class was generated from the following file:

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

6.22 BeeGame.Blocks.Grass Class Reference

Grass Block

Inheritance diagram for BeeGame.Blocks.Grass:



Public Member Functions

- [Grass](#) ()
Constructor also sets teh 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

6.22.1 Detailed Description

Grass Block

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

6.22.2 Constructor & Destructor Documentation

6.22.2.1 Grass()

```
BeeGame.Blocks.Grass.Grass ( )
```

Constructor also sets teh items name

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

```
00022 : base("Grass") {}
```

6.22.3 Member Function Documentation

6.22.3.1 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     }
```

6.22.3.2 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     }
```

6.22.3.3 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 position 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     }

```

6.22.3.4 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     }

```

6.22.3.5 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     }

```

6.22.4 Member Data Documentation**6.22.4.1 ID**

```
new int BeeGame.Blocks.Grass.ID => 4 [static]
```

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

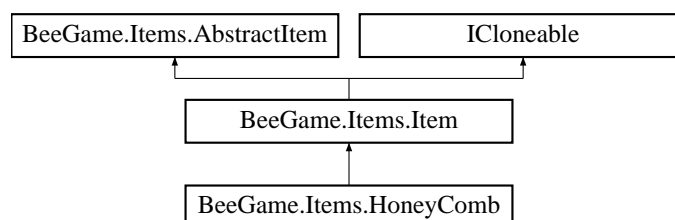
The documentation for this class was generated from the following file:

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

6.23 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](#) ()
Retuens the sprite for the 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 coumb, [BeeDictionary](#).[GetCombColour](#)([HoneyCombType](#))

Private Attributes

- [Sprite](#) [itemSprite](#)
The Sprite for this honey comb

Additional Inherited Members

6.23.1 Detailed Description

Honey comb item produced by bees

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

6.23.2 Constructor & Destructor Documentation

6.23.2.1 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($"
{HoneyCombType.HONEY} Comb".ToLower()))
00047     {
00048         usesGameObject = true;
00049         type = HoneyCombType.HONEY;
00050     }
```

6.23.2.2 HoneyComb() [2/2]

```
BeeGame.Items.HoneyComb.HoneyComb (
    HoneyCombType type )
```

Makes a [HoneyComb](#) for the given `HoneyCombType`

Parameters

<i>type</i>	that this comb is
-------------	-------------------

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

```
00056             : base(new CultureInfo("en-US", false).TextInfo.ToTitleCase($"
{type.ToString()} Comb".ToLower()))
00057     {
00058         usesGameObject = true;
00059         this.type = type;
00060     }
```

6.23.3 Member Function Documentation

6.23.3.1 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 obejct
00081         obj.GetComponent<ApplyColour>().colour = CombColour;
00082         return obj;
00083     }
```

6.23.3.2 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         }
```

6.23.3.3 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         }
```

6.23.3.4 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 =  
SpriteDictionary.GetSprite("HoneyComb").ColourSprite(  
CombColour));  
00071         }
```

6.23.4 Member Data Documentation

6.23.4.1 ID

```
new int BeeGame.Items.HoneyComb.ID => 12 [static]
```

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

6.23.4.2 itemSprite

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

The Sprite for this honey comb

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

6.23.5 Property Documentation

6.23.5.1 CombColour

```
Color BeeGame.Items.HoneyComb.CombColour [get]
```

The colour if this coumb, BeeDictionary.GetCombColour(HoneyCombType)

Definition at line 26 of file [HoneyComb.cs](#).

6.23.5.2 type

```
HoneyCombType BeeGame.Items.HoneyComb.type [get], [set]
```

The type of comb this is, HoneyCombType

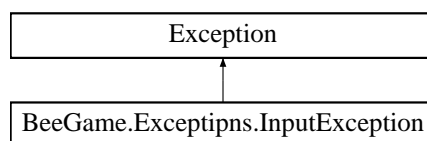
Definition at line 20 of file [HoneyComb.cs](#).

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

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

6.24 BeeGame.Exceptipns.InputException Class Reference

Inheritance diagram for BeeGame.Exceptipns.InputException:



Public Member Functions

- [InputException](#) ()
- [InputException](#) (string message)
- [InputException](#) (string message, Exception innerException)

6.24.1 Detailed Description

Definition at line 8 of file [InputException.cs](#).

6.24.2 Constructor & Destructor Documentation

6.24.2.1 InputException() [1/3]

```
BeeGame.Exceptipns.InputException.InputException ( )
```

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

```
00010                                     : base()
00011     {
00012
00013     }
```

6.24.2.2 InputException() [2/3]

```
BeeGame.Exceptipns.InputException.InputException (
    string message )
```

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

```
00015                                     : base(message)
00016     {
00017
00018     }
```

6.24.2.3 InputException() [3/3]

```
BeeGame.Exceptipns.InputException.InputException (
    string message,
    Exception innerException )
```

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

```
00020                                     : base(message, innerException)
00021     {
00022
00023     }
```

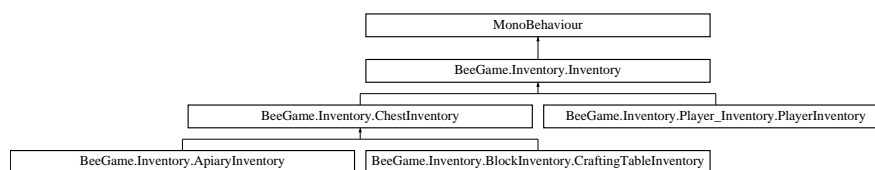
The documentation for this class was generated from the following file:

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

6.25 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 soze to the number of slots in the invnetory
- void [SetAllItems](#) ([ItemsInInventory](#) items)
Sets the [items](#) to the given [ItemsInInventory](#)
- void [UpdateBase](#) ()
Things in the inventory that should be updated
- virtual void [ToggleInventory](#) ([Inventory](#) inv)
- virtual void [SaveInv](#) ()
Saves the inventory
- [ItemsInInventory](#) [GetAllItems](#) ()
Gets all of the items in the invntory
- 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 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 [floatingItems](#) [Item.GetItemSprite\(\)](#) at the mouse position
- void [PutItemsInSlots](#) ()
Sets an Item in the [ItemsInInventory.itemsInInventory](#) array to a [InventorySlot.item](#)

Private Attributes

- GameObject [spriteAtCursor](#)
The sprite at the cursor

6.25.1 Detailed Description

Base class for all inventories in the game

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

6.25.2 Member Function Documentation

6.25.2.1 AddItemToInventory()

```
bool BeeGame.Inventory.Inventory.AddItemToInventory (  
    Item item )
```

Add an item to the inventory

Parameters

<i>item</i>	Item to add
-------------	-------------

Returns

true if item wasa added

Definition at line 176 of file [Inventory.cs](#).

```
00177      {
00178          return items.AddItem(item);
00179      }
```

6.25.2.2 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 164 of file [Inventory.cs](#).

```
00165      {
00166          items.AddItem(slotIndex, item);
00167          /* saves the inventory changes
00168          Serialization.Serialization.SerializeInventory(this, inventoryName);
00169      }
```

6.25.2.3 DrawItemAtCursor()

```
void BeeGame.Inventory.Inventory.DrawItemAtCursor ( ) [private]
```

Draws the [floatingItems](#) 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
00110                 spriteAtCursor.transform.GetChild(0).position = Input.mousePosition;
00111             }
00112             else
00113             {
00114                 Destroy(spriteAtCursor);
00115             }
00116         }

```

6.25.2.4 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 154 of file [Inventory.cs](#).

```

00155         {
00156             return items;
00157         }

```

6.25.2.5 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
00057             return false;
00058         }

```

6.25.2.6 PutItemsInSlots()

```
void BeeGame.Inventory.Inventory.PutItemsInSlots ( ) [private]
```

Sets an Item in the [ItemsInInventory.itemsInInventory](#) array to a [InventorySlot.item](#)

Definition at line [139](#) of file [Inventory.cs](#).

```
00140     {
00141         /** goes through all of the items in the array setting then all to a slot
00142         for (int i = 0; i < slots.Length; i++)
00143         {
00144             slots[i].slotIndex = i;
00145             slots[i].myInventory = this;
00146             slots[i].item = items.itemsInInventory[i];
00147         }
00148     }
```

6.25.2.7 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](#).

Definition at line [131](#) of file [Inventory.cs](#).

```
00132     {
00133         Serialization.Serialization.SerializeInventory(this, inventoryName);
00134     }
```

6.25.2.8 SetAllItems()

```
void BeeGame.Inventory.Inventory.SetAllItems (
    ItemsInInventory items )
```

Sets the [items](#) to the given [ItemsInInventory](#)

Parameters

<i>items</i>	Items to set this inventory to
--------------	--

remarks> Used during deserialization to restor the inventory /remarks>

Definition at line [76](#) of file [Inventory.cs](#).

```
00077     {
00078         this.items = items;
00079     }
```

6.25.2.9 SetInventorySize()

```
void BeeGame.Inventory.Inventory.SetInventorySize (
    int inventorySize )
```

Sets the inventory soze to the number of slots in the invnetory

Parameters

<i>inventorySize</i>	
----------------------	--

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

```
00065         {
00066             items = new ItemsInInventory(slots.Length);
00067         }
```

6.25.2.10 ToggleInventory()

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

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

Definition at line 120 of file [Inventory.cs](#).

```
00121         {
00122             throw new NotImplementedException();
00123         }
```

6.25.2.11 UpdateBase()

```
void BeeGame.Inventory.Inventory.UpdateBase ( )
```

Things in the inventory that should be updated

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

```
00087         {
00088             PutItemsInSlots();
00089             DrawItemAtCursor();
00090         }
```

6.25.3 Member Data Documentation

6.25.3.1 floatingItem

```
Item BeeGame.Inventory.Inventory.floatingItem [package]
```

Item that is currently being moved

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

6.25.3.2 inventoryName

```
string BeeGame.Inventory.Inventory.inventoryName = ""
```

Name of this inventory

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

6.25.3.3 items

```
ItemsInInventory BeeGame.Inventory.Inventory.items
```

[Items](#) in the inventory

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

6.25.3.4 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](#).

6.25.3.5 slots

```
InventorySlot [ ] BeeGame.Inventory.Inventory.slots
```

Slots in the inventory

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

6.25.3.6 spriteAtCursor

GameObject BeeGame.Inventory.Inventory.spriteAtCursor [private]

The sprite at the cursor

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

6.25.3.7 thisInventoryOpen

bool BeeGame.Inventory.Inventory.thisInventoryOpen = false [protected]

is this inventory open?

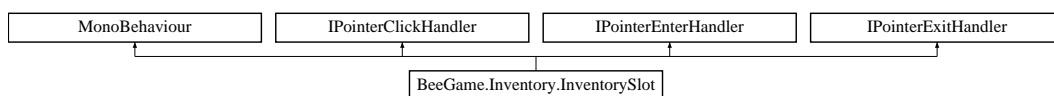
Definition at line 33 of file [Inventory.cs](#).

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

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

6.26 BeeGame.Inventory.InventorySlot Class Reference

Inheritance diagram for BeeGame.Inventory.InventorySlot:



Public Member Functions

- 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

Package Attributes

- int [slotIndex](#)

The slot in the inventory this is

Private Member Functions

- void [Update](#) ()
Updates the slot
- void [UpdateIcon](#) ()
Applies the correct icon to the slot depending on what is in the slot
- 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 [CheckItem](#) ()
checks that the item is valid
- void [OnDisable](#) ()
Destroys the item text when the inventory is closed

6.26.1 Detailed Description

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

6.26.2 Member Function Documentation

6.26.2.1 AddToFloatingItem()

```
void BeeGame.Inventory.InventorySlot.AddToFloatingItem ( ) [private]
```

Add items from the slot to the [Inventory.floatingItem](#)

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

```

00185         {
00186             /** if the whole stack can be added do it and move on
00187             if(myInventory.floatingItem.itemStackCount +
00188             item.itemStackCount <= item.maxStackCount)
00189             {
00190                 myInventory.floatingItem.itemStackCount +=
00191                 item.itemStackCount;
00192                 item = null;
00193                 myInventory.AddItemToSlots(slotIndex,
00194                 item);
00195                 return;
00196             }
00197             /** if the whole stack cannot be added calculate how many need to be removed from the slots
00198             item stack
00199             item.itemStackCount -= (item.maxStackCount -
00200             myInventory.floatingItem.itemStackCount);
00201             /** set the floating item to the max stack count
00202             myInventory.floatingItem.itemStackCount =
00203             item.maxStackCount;
00204             myInventory.AddItemToSlots(slotIndex,
00205             item);
00206         }

```

6.26.2.2 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 210 of file [InventorySlot.cs](#).

```

00211         {
00212             /** if the item in the slot is null create it
00213             if (item == null)
00214             {
00215                 item = myInventory.floatingItem.CloneObject();
00216                 item.itemStackCount = 0;
00217             }
00218             /** add to number to add to the stack count
00219             item.itemStackCount += numerToAdd;
00220             /** if the stack count is now larger than it should be dont let it be
00221             if (item.itemStackCount > item.maxStackCount)
00222             {
00223                 item.itemStackCount = item.maxStackCount;
00224             }
00225             /** remove the numebr if items form the floating item then check the floating item is not null
00226             myInventory.floatingItem.itemStackCount -= numerToAdd;
00227             CheckFloatingItem();
00228             /** save the inventory changes
00229             myInventory.AddItemToSlots(slotIndex,
00230             item);
00231         }

```

6.26.2.3 CheckFloatingItem()

```
void BeeGame.Inventory.InventorySlot.CheckFloatingItem ( ) [private]
```

Checks if the [Inventory.floatingItem](#) should be null

Definition at line [275](#) of file [InventorySlot.cs](#).

```
00276     {
00277         if(myInventory.floatingItem.itemStackCount <= 0)
00278         {
00279             myInventory.floatingItem = null;
00280         }
00281     }
```

6.26.2.4 CheckItem()

```
void BeeGame.Inventory.InventorySlot.CheckItem ( ) [private]
```

checks that the item is valid

Definition at line [287](#) of file [InventorySlot.cs](#).

```
00288     {
00289         if (item != null && myInventory != null)
00290         {
00291             if (item.itemStackCount == 0 || item.
00292                 itemName == "TestItem")
00293             {
00294                 myInventory.items.itemsInInventory[
00295                     slotIndex] = null;
00296                 Destroy(itemText);
00297             }
00298         }
```

6.26.2.5 OnDisable()

```
void BeeGame.Inventory.InventorySlot.OnDisable ( ) [private]
```

Destroys the item text when the inventory is closed

Definition at line [329](#) of file [InventorySlot.cs](#).

```
00330     {
00331         Destroy(itemText);
00332     }
```

6.26.2.6 OnPointerClick()

```
void BeeGame.Inventory.InventorySlot.OnPointerClick (
    PointerEventData eventData )
```

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

Definition at line 83 of file [InventorySlot.cs](#).

```

00084         {
00085             if (myInventory.floatingItem != null)
00086             {
00087                 /* Left click moves whole stacks of items
00088                 if (eventData.button == PointerEventData.InputButton.Left)
00089                 {
00090                     /* If the item in the slot is empty put the floating item into it then clear it and
the slot can have items inserted
00091                     if (item == null && itemsCanBeInserted)
00092                     {
00093                         item = myInventory.floatingItem;
00094                         myInventory.floatingItem = null;
00095                         myInventory.AddItemToSlots (
slotIndex, item);
00096                     }
00097                     return;
00098                 }
00099                 /* if the items are the same
if(myInventory.floatingItem == item &&
itemsCanBeInserted)
00100                 {
00101                     /* if the item in the inventoys stack count + the floating items stack count is
less than the max stack count
00102                     if (myInventory.floatingItem.
itemStackCount + item.itemStackCount <= item.
maxStackCount)
00103                     {
00104                         AddToSlot(myInventory.
floatingItem.itemStackCount);
00105                         return;
00106                     }
00107                     /* if the item stack added is larger than the max count add as many as you can and
move on
00108                     else
00109                     {
00110                         AddToSlot(item.maxStackCount -
item.itemStackCount);
00111                         return;
00112                     }
00113                 }
00114                 /* if the tiems are the same but items cannot be inserted into the slot add as many
items as you
00115                 /* can from the slot to the floating item
00116                 else if(myInventory.floatingItem ==
item && !itemsCanBeInserted)
00117                 {
00118                     AddToFloatingItem();
00119                     {
00120                         if (myInventory is BlockInventory.CraftingTableInventory c)
00121                             c.result.Invoke();
00122                     }
00123                     return;
00124                 }
00125                 /* If the items were not == swap them
00126                 else
00127                 {
00128                     /* only if items can be inserted into the slot
00129                     if(itemsCanBeInserted)
00130                         SwapItems();
00131                     return;
00132                 }
00133             }
00134             else if(eventData.button == PointerEventData.InputButton.Right)
00135             {
00136                 /* if the item in slot is null add 1 from the floating item to it
00137                 if(item == null && itemsCanBeInserted)
00138                 {
00139                     AddToSlot(1);
00140                     return;
00141                 }
00142                 /* if the items are the same add 1 from the floating item to this item
00143                 else if(item == myInventory.floatingItem &&
itemsCanBeInserted)
00144                 {

```

```

00145             AddToSlot (1);
00146             return;
00147         }
00148     }
00149 }
00150 /** if the floating item is null
00151 else
00152 {
00153     /** add 1/2 of the stack into the floating item if right click was pressed
00154     if(eventData.button == PointerEventData.InputButton.Right)
00155     {
00156         SplitStack();
00157
00158         /** blocks removed some weird name confliction
00159         {
00160             if (myInventory is BlockInventory.CraftingTableInventory c)
00161                 c.result.Invoke();
00162         }
00163
00164         return;
00165     }
00166
00167     /** otherwise add the items into the floating item slot
00168     SwapItems();
00169     /** ^ does not need to check that the slot cannot be inserted into as null be being
    inserted because the floating item is null
00170
00171     {
00172         if (myInventory is BlockInventory.CraftingTableInventory c)
00173             c.result.Invoke();
00174     }
00175
00176     return;
00177 }
00178
00179 }
```

6.26.2.7 OnPointerEnter()

```
void BeeGame.Inventory.InventorySlot.OnPointerEnter (
    PointerEventData eventData )
```

Makes the text object when the cursor is over the slot

Parameters

<i>eventData</i>	Not used but required for the interface
------------------	---

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

```

00305     {
00306         /** if the item is null or the floating item has something in it dont display the item text as
    it is not necessary
00307         if (item != null && myInventory.floatingItem == null)
00308         {
00309             itemText = Instantiate(PrefabDictionary.
GetPrefab("ItemDetails"));
00310             /** sets the text to the correct postion
00311             itemText.transform.GetChild(0).position = Input.mousePosition;
00312             /** puts the correct text in the box
00313             itemText.transform.GetChild(0).GetChild(0).GetComponent<Text>().text = $"
{item.GetItemName()} \nStack: {item.itemStackCount}";
00314         }
00315     }
```

6.26.2.8 OnPointerExit()

```
void BeeGame.Inventory.InventorySlot.OnPointerExit (
    PointerEventData eventData )
```

Destroys the text object when the cursor is not over the slot anymore

Parameters

<i>eventData</i>	Not used but required for the interface
------------------	---

Definition at line 321 of file [InventorySlot.cs](#).

```
00322     {
00323         Destroy(itemText);
00324     }
```

6.26.2.9 SplitStack()

```
void BeeGame.Inventory.InventorySlot.SplitStack ( ) [private]
```

Halves a Item.itemStackCount between the slot and the [Inventory.floatingItem](#)

If the stack count of 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 makes the item in that slot go into the floating item

Definition at line 241 of file [InventorySlot.cs](#).

```
00242     {
00243         myInventory.floatingItem = item.CloneObject();
00244         int give = (item.itemStackCount + 1) / 2;
00245         myInventory.floatingItem.itemStackCount = give;
00246         item.itemStackCount -= give;
00247
00248         if (item.itemStackCount <= 0)
00249             item = null;
00250
00251         myInventory.AddItemToSlots(slotIndex,
00252             item);
00252         Destroy(itemText);
00253     }
```

6.26.2.10 SwapItems()

```
void BeeGame.Inventory.InventorySlot.SwapItems ( ) [private]
```

Swaps the Item in the [Inventory.floatingItem](#) with the slot's item

Definition at line 258 of file [InventorySlot.cs](#).

```
00259     {
00260         /* temp copy of the item
00261         Item temp = myInventory.floatingItem;
00262         /* sets the floating item
00263         myInventory.floatingItem = item;
00264         /* sets the item that was in the floating item to the item in the slot
00265         item = temp;
00266         /* Saves the changes to the inventory
00267         myInventory.AddItemToSlots(slotIndex,
00268             item);
00268         /* destroys the text as it is not needed anymore
00269         Destroy(itemText);
00270     }
```

6.26.2.11 Update()

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

Updates the slot

Definition at line 42 of file [InventorySlot.cs](#).

```
00043     {
00044         CheckItem();
00045         UpdateIcon();
00046     }
```

6.26.2.12 UpdateIcon()

```
void BeeGame.Inventory.InventorySlot.UpdateIcon ( ) [private]
```

Applies the correct icon to the slot depending on what is in the slot

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

```
00053     {
00054         if(item == null)
00055         {
00056             GetComponent<Image>().sprite = null;
00057         }
00058         else
00059         {
00060             if(!item.Equals(new Item()))
00061                 GetComponent<Image>().sprite = item.GetItemSprite();
00062         }
00063
00064         /* if the slot is selected in the hotbar give the player some indication by colouring it grey
00065         if (selectedSlot)
00066         {
00067             GetComponent<Image>().color = Color.gray;
00068         }
00069         else
00070         {
00071             GetComponent<Image>().color = Color.white;
00072         }
00073     }
```

6.26.3 Member Data Documentation

6.26.3.1 item

[Item](#) BeeGame.Inventory.InventorySlot.item

The item this slot has in it

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

6.26.3.2 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](#).

6.26.3.3 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](#).

6.26.3.4 myInventory

```
Inventory BeeGame.Inventory.InventorySlot.myInventory
```

The [Inventory](#) this slot is in

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

6.26.3.5 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](#).

6.26.3.6 slotIndex

```
int BeeGame.Inventory.InventorySlot.slotIndex [package]
```

The slot in the inventory this is

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

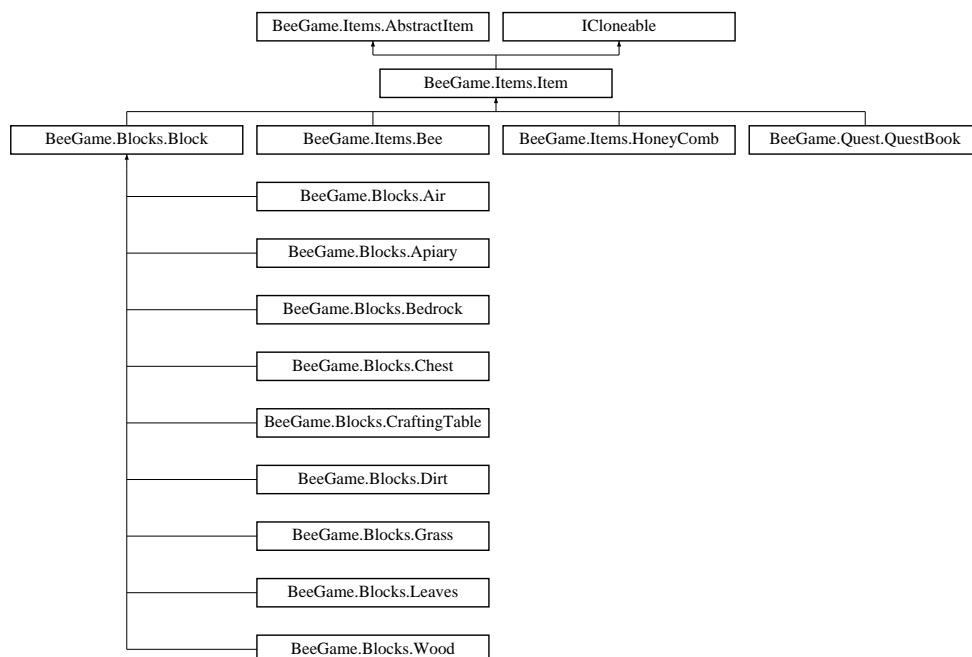
The documentation for this class was generated from the following file:

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

6.27 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 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
- 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
- int `itemStackCount` [get, set]
Number of items in the stack

Private Attributes

- const float `tileSize` = 0.1f
How big are the texture tiles in the texture map (1/tile number x)
- int `count` = 1

6.27.1 Detailed Description

Base class for all `Items` and `Blocks` in the game

Definition at line 16 of file `Item.cs`.

6.27.2 Constructor & Destructor Documentation

6.27.2.1 `Item()` [1/2]

```
BeeGame.Items.Item.Item ( )
```

Definition at line 51 of file `Item.cs`.

```
00052         {  
00053             itemName = "TestItem";  
00054         }
```

6.27.2.2 `Item()` [2/2]

```
BeeGame.Items.Item.Item (  
    string name )
```

Definition at line 56 of file `Item.cs`.

```
00057         {  
00058             itemName = name;  
00059         }
```

6.27.3 Member Function Documentation

6.27.3.1 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 330 of file [Item.cs](#).

```
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     }
```

6.27.3.2 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 367 of file [Item.cs](#).

```
00368     {
00369         if (!(obj is Item))
00370             return false;
00371
00372         return this == (obj as Item);
00373     }
```

6.27.3.3 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 194 of file [Item.cs](#).

```
00195     {
00196         /* Adds vertices in a anti-clockwise order
00197         meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z -
00198         blockSize), addToRenderMesh);
00199         meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z -
00200         blockSize), addToRenderMesh);
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         /* adds teh tirs for the quad
00206         meshData.AddQuadTriangles(addToRenderMesh);
00207         /* if the data should be added to the render mesh also add the uvs to the mesh
00208         if (addToRenderMesh)
00209             meshData.uv.AddRange(FaceUVs(Direction.DOWNN));
00210         return meshData;
00211     }
```

6.27.3.4 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 250 of file [Item.cs](#).

```

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         /* adds teh tirs for the quad
00262         meshData.AddQuadTriangles (addToRenderMesh);
00263         /* if the data should be added to the render mesh also add the uvs to the mesh
00264         if (addToRenderMesh)
00265             meshData.uv.AddRange(FaceUVs(Direction.EAST));
00266         return meshData;
00267     }

```

6.27.3.5 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 222 of file [Item.cs](#).

```

00223     {
00224         /* Adds vertices in a anti-clockwise order
00225         meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z +
blockSize), addToRenderMesh);
00226         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z +
blockSize), addToRenderMesh);
00227         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z +
blockSize), addToRenderMesh);
00228         meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z +
blockSize), addToRenderMesh);
00229
00230         /* adds teh tirs for the quad
00231         meshData.AddQuadTriangles (addToRenderMesh);
00232
00233         /* if the data should be added to the render mesh also add the uvs to the mesh
00234         if (addToRenderMesh)
00235             meshData.uv.AddRange(FaceUVs(Direction.NORTH));
00236
00237         return meshData;
00238     }

```

6.27.3.6 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 278 of file [Item.cs](#).

```

00279     {
00280         /* Adds vertices in a anti-clockwise order
00281         meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z -
blockSize), addToRenderMesh);

```



```

00282         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z -
blockSize), addToRenderMesh);
00283         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z -
blockSize), addToRenderMesh);
00284         meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z -
blockSize), addToRenderMesh);
00285
00286         /* adds teh tirs for the quad
00287         meshData.AddQuadTriangles (addToRenderMesh);
00288
00289         /* if the data should be added to the render mesh also add the uvs to the mesh
00290         if (addToRenderMesh)
00291             meshData.uv.AddRange (FaceUVs (Direction.SOUTH));
00292
00293         return meshData;
00294     }

```

6.27.3.7 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 166 of file [Item.cs](#).

```

00167     {
00168         /* Adds vertices in a anti-clockwise order
00169         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z +
blockSize), addToRenderMesh, Direction.UP);
00170         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z +
blockSize), addToRenderMesh, Direction.UP);
00171         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z -
blockSize), addToRenderMesh, Direction.UP);
00172         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z -
blockSize), addToRenderMesh, Direction.UP);
00173
00174         /* adds teh tirs for the quad
00175         meshData.AddQuadTriangles (addToRenderMesh);
00176
00177         /* if the data should be added to the render mesh also add the uvs to the mesh
00178         if (addToRenderMesh)
00179             meshData.uv.AddRange (FaceUVs (Direction.UP));
00180
00181         return meshData;
00182     }

```

6.27.3.8 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 306 of file [Item.cs](#).

```
00307     {
00308         /** Adds vertices in a anti-clockwise order
00309         blockSize), meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z +
00310         blockSize), addToRenderMesh);
00311         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z +
00312         blockSize), addToRenderMesh);
00313         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z -
00314         blockSize), addToRenderMesh);
00315         meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z -
00316         blockSize), addToRenderMesh);
00317         /** adds teh tirs for the quad
00318         meshData.AddQuadTriangles(addToRenderMesh);
00319         /** if the data should be added to the render mesh also add the uvs to the mesh
00320         if (addToRenderMesh)
00321             meshData.uv.AddRange(FaceUVs(Direction.WEST));
00322         return meshData;
00323     }
```

6.27.3.9 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 UVsreturns>

Definition at line 141 of file [Item.cs](#).

```

00142     {
00143         /** only 4 uvs per face
00144         Vector2[] UVs = new Vector2[4];
00145         Tile tilePos = TexturePosition(direction);
00146
00147         /** sets the UVs for each vertex
00148         UVs[0] = new THVector2(tileSize * tilePos.x +
00149     tileSize - 0.01f, tileSize * tilePos.y + 0.01f);
00149         UVs[1] = new THVector2(tileSize * tilePos.x +
00150     tileSize - 0.01f, tileSize * tilePos.y + tileSize - 0.01f);
00150         UVs[2] = new THVector2(tileSize * tilePos.x + 0.01f,
00151     tileSize * tilePos.y + tileSize - 0.01f);
00151         UVs[3] = new THVector2(tileSize * tilePos.x + 0.01f,
00152     tileSize * tilePos.y + 0.01f);
00152
00153         return UVs;
00154     }

```

6.27.3.10 GetGameObject()

```
virtual GameObject BeeGame.Items.Item.GetGameObject ( ) [virtual]
```

Returns the GameObject for the item of it has one

Returns

GameObject for the item

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

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

```
00074 { return null; }
```

6.27.3.11 GetHashCode()

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

Returns the hashcode for the item

Returns

1

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

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

Definition at line 357 of file [Item.cs](#).

```

00358     {
00359         return ID;
00360     }

```

6.27.3.12 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 80 of file [Item.cs](#).

```
00081     {  
00082         return $"{GetHashCode()}";  
00083     }
```

6.27.3.13 GetItemName()

```
override string BeeGame.Items.Item.GetItemName ( ) [virtual]
```

Returns the items name

Returns

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

Definition at line 98 of file [Item.cs](#).

```
00099     {  
00100         return $"{itemName}";  
00101     }
```

6.27.3.14 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.Bee](#), [BeeGame.Items.HoneyComb](#), [BeeGame.Blocks.Block](#), [BeeGame.Blocks.Grass](#), [BeeGame.Blocks.Dirt](#), [BeeGame.Quest.QuestBook](#), [BeeGame.Blocks.Wood](#), and [BeeGame.Blocks.Leaves](#).

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

```
00090     {  
00091         return SpriteDictionary.GetSprite("TestSprite");  
00092     }
```

6.27.3.15 InteractWithObject()

```
virtual bool BeeGame.Items.Item.InteractWithObject ( ) [virtual]
```

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

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

```
00064     {
00065         return false;
00066     }
```

6.27.3.16 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 123 of file [Item.cs](#).

```
00124     {
00125         /* adds all faces of the item to the mesh as all faces could be seen at any time
00126         meshData = FaceDataUp(x, y, z, meshData, true, 0.25f);
00127         meshData = FaceDataDown(x, y, z, meshData, true, 0.25f);
00128         meshData = FaceDataNorth(x, y, z, meshData, true, 0.25f);
00129         meshData = FaceDataEast(x, y, z, meshData, true, 0.25f);
00130         meshData = FaceDataSouth(x, y, z, meshData, true, 0.25f);
00131         meshData = FaceDataWest(x, y, z, meshData, true, 0.25f);
00132
00133         return meshData;
00134     }
```

6.27.3.17 operator!=(())

```
static bool BeeGame.Items.Item.operator!= (
    Item a,
    Item b ) [static]
```

Inverse of ==

Parameters

<i>a</i>	Item
<i>b</i>	Item

Returns

True if $a \neq b$

Definition at line [400](#) of file [Item.cs](#).

```
00401      {
00402          return !(a == b);
00403      }
```

6.27.3.18 operator==()

```
static bool BeeGame.Items.Item.operator== (
    Item a,
    Item b ) [static]
```

Overrides the default == operator as different things need to be checked

Parameters

<i>a</i>	Item
<i>b</i>	Item

Returns

true if $a == b$

Definition at line [381](#) of file [Item.cs](#).

```
00382      {
00383          if (ReferenceEquals(a, null) && ReferenceEquals(b, null))
00384              return true;
00385          if (ReferenceEquals(a, null) || ReferenceEquals(b, null))
00386              return false;
00387          if (a.GetItemID() == b.GetItemID())
00388              return true;
00389          return false;
00390      }
```

6.27.3.19 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.Wood](#), and [BeeGame.Blocks.Leaves](#).

Definition at line 110 of file [Item.cs](#).

```
00111         {
00112             return new Tile() { x = 1, y = 9 };
00113         }
```

6.27.3.20 ToString()

```
override string BeeGame.Items.Item.ToString ( )
```

Returns the item name and id formatted nicely

Returns

Definition at line 348 of file [Item.cs](#).

```
00349         {
00350             return $"{itemName} \nID: {GetItemID()}";
00351         }
```

6.27.4 Member Data Documentation**6.27.4.1 count**

```
int BeeGame.Items.Item.count = 1 [private]
```

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

6.27.4.2 ID

```
int BeeGame.Items.Item.ID => 0 [static]
```

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

6.27.4.3 maxStackCount

```
virtual int BeeGame.Items.Item.maxStackCount => 64
```

Max number of items in a stack

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

6.27.4.4 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](#).

6.27.4.5 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](#).

6.27.5 Property Documentation

6.27.5.1 itemName

```
string BeeGame.Items.Item.itemName [get], [set], [package]
```

Name of the item

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

6.27.5.2 itemStackCount

```
int BeeGame.Items.Item.itemStackCount [get], [set]
```

Number of items in the stack

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

6.27.5.3 usesGameObject

```
bool BeeGame.Items.Item.usesGameObject [get], [set]
```

Does the item use a gameobject

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

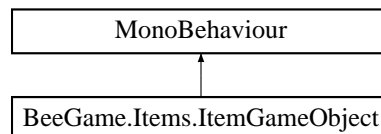
The documentation for this class was generated from the following file:

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

6.28 BeeGame.Items.ItemGameObject Class Reference

Interface between item and inity gameobjects

Inheritance diagram for BeeGame.Items.ItemGameObject:



Public Attributes

- [Item](#) [item](#)
[Item](#) that this gameobject represents
- [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

6.28.1 Detailed Description

Interface between item and inity gameobjects

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

6.28.2 Member Function Documentation

6.28.2.1 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     }
```

6.28.2.2 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     }
```

6.28.2.3 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         }
```

6.28.3 Member Data Documentation

6.28.3.1 go

```
GameObject BeeGame.Items.ItemGameObject.go
```

GameObject to make

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

6.28.3.2 item

```
Item BeeGame.Items.ItemGameObject.item
```

[Item](#) that this gameobject represents

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/[ItemGameObject.cs](#)

6.29 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

6.29.1 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](#).

6.29.2 Constructor & Destructor Documentation

6.29.2.1 ItemsInInventory()

```
BeeGame.Inventory.ItemsInInventory.ItemsInInventory (
    int numberOfInventorySlots )
```

Sets the size of the inventory

Parameters

<i>numberOfInventorySlots</i>	
-------------------------------	--

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

```
00022     {
00023         itemsInInventory = new Item(numberOfInventorySlots);
00024     }
```

6.29.3 Member Function Documentation

6.29.3.1 AddItem() [1/2]

```
void BeeGame.Inventory.ItemsInInventory.AddItem (
    int index,
    Item item )
```

Add an Item to a specific index in the inventory

Parameters

<i>index</i>	Were to add the item
<i>item</i>	What Item to put in the inventory

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

```
00032     {
00033         itemsInInventory[index] = item;
00034     }
```

6.29.3.2 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;
00057     }
```

6.29.4 Member Data Documentation

6.29.4.1 itemsInInventory

```
Item [] BeeGame.Inventory.ItemsInInventory.itemsInInventory
```

All of the items in the inventory

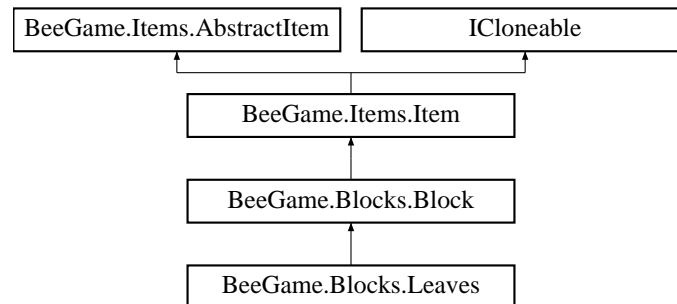
Definition at line 15 of file [ItemsInInventory.cs](#).

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

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

6.30 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 postion 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

6.30.1 Detailed Description

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

6.30.2 Constructor & Destructor Documentation

6.30.2.1 Leaves()

`BeeGame.Blocks.Leaves.Leaves ()`

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

```

00014         : base ("Leaves")
00015     {
00016
00017     }
```

6.30.3 Member Function Documentation

6.30.3.1 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     }
```

6.30.3.2 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     }
```

6.30.3.3 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         }
```

6.30.3.4 TexturePosition()

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

Texture postion 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         }
```

6.30.3.5 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         }
```


6.30.4 Member Data Documentation

6.30.4.1 ID

```
new int BeeGame.Blocks.Leaves.ID => 6 [static]
```

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

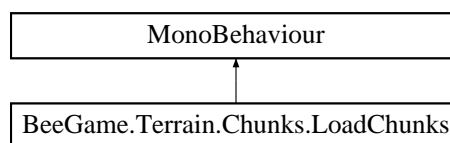
The documentation for this class was generated from the following file:

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

6.31 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 Remmoves [Chunks](#)
- void [ApplyCollsionMeshToNearbyChunks](#) ()
Makes a collsion mesh for the [Chunks](#) nearest to the player to reduce lag created by PhysX mesh bakeing
- void [LoadAndRenderChunks](#) ()
Gets the chunks that could be built and renders then renders them
- void [FindChunksToLoad](#) ()
Finds the [Chunks](#) that should be rendered
- void [BuildChunk](#) ([ChunkWorldPos](#) pos)
Makes a chunk in the given positon 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

6.31.1 Detailed Description

Loads the [Chunks](#) around the player

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

6.31.2 Member Function Documentation

6.31.2.1 ApplyCollsionMeshToNearbyChunks()

```
void BeeGame.Terrain.Chunks.LoadChunks.ApplyCollsionMeshToNearbyChunks ( ) [private]
```

Makes a collision mesh for the [Chunks](#) nearest to the player to reduce lag created by PhysX mesh baking

We dont need to worry about removeing [Chunk](#) collision meshes as once PhysX has baked then they have minimal performance impact Doing things this wayt 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.
chunkSize) * Chunk.chunkSize, Mathf.FloorToInt(transform.position.y / Chunk.chunkSize) * Chunk.chunkSize, Mathf.
FloorToInt(transform.position.z / Chunk.chunkSize) * Chunk.chunkSize);
00115
00116         for (int i = 0; i < nearbyChunks.Length; i++)
00117         {
00118             ChunkWorldPos chunkPos = new ChunkWorldPos(nearbyChunks[i].x * Chunk.chunkSize
+ playerPos.x, 0, nearbyChunks[i].z * Chunk.chunkSize + playerPos.z);
00119
00120             for (int j = -1; j < 2; j++)
00121             {
00122                 Chunk nearbyChunk = world.GetChunk(chunkPos.x, j * Chunk.chunkSize,
chunkPos.z);
00123
00124                 if (nearbyChunk != null)
00125                     nearbyChunk.applyCollisionMesh = true;
00126             }
00127         }
00128     }
```

6.31.2.2 BuildChunk()

```
void BeeGame.Terrain.Chunks.LoadChunks.BuildChunk (
    ChunkWorldPos pos ) [private]
```

Makes a chunk in the given positon if it does not already exist

Parameters

<i>pos</i>	hte positon 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     }

```

6.31.2.3 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     }

```

6.31.2.4 FindChunksToLoad()

```
void BeeGame.Terrain.Chunks.LoadChunks.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.
chunkSize) * Chunk.chunkSize, Mathf.FloorToInt(transform.position.y / Chunk.chunkSize) * Chunk.chunkSize,
Mathf.FloorToInt(transform.position.z / Chunk.chunkSize) * Chunk.chunkSize);
00156
00157             /** check all of the chunk positions and if that position does not have a chunk in it make
it
00158             for (int i = 0; i < chunkPositions.Length; i++)
00159             {
00160                 ChunkWorldPos newChunkPos = new ChunkWorldPos(chunkPositions[i].x * Chunk
.chunkSize + playerPos.x, 0, chunkPositions[i].z * Chunk.chunkSize + playerPos.z);
00161
00162                 Chunk newChunk = world.GetChunk(newChunkPos.x, newChunkPos.y, newChunkPos.
z);
00163
00164                 if (newChunk != null && (newChunk.rendered || buildList.Contains(newChunkPos))
)
00165                     continue;
00166
00167                 for (int y = -1; y < 2; y++)
00168                 {
00169                     for (int x = newChunkPos.x - Chunk.chunkSize; x < newChunkPos.x + Chunk.chunkSize;
x += Chunk.chunkSize)
00170                     {
00171                         for (int z = newChunkPos.z - Chunk.chunkSize; z < newChunkPos.z + Chunk.
chunkSize; z += Chunk.chunkSize)
00172                         {
00173                             buildList.Add(new ChunkWorldPos(x, y * Chunk.chunkSize, z));
00174                         }
00175                     }
00176                 }
00177                 return;
00178             }
00179         }
00180     }
```

6.31.2.5 LoadAndRenderChunks()

```
void BeeGame.Terrain.Chunks.LoadChunks.LoadAndRenderChunks ( ) [private]
```

Gets the chunks that could be built and renders then 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
chunk is missed because chunks are removed from the list as they are made
00139             for (int i = buildList.Count - 1, j = 0; i >= 0 && j < 8; i--, j++)
00140             {
00141                 BuildChunk(buildList[0]);
00142                 buildList.RemoveAt(0);
00143             }
00144         }
00145     }
```

6.31.2.6 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         }
```

6.31.2.7 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         }
```

6.31.3 Member Data Documentation

6.31.3.1 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](#).

6.31.3.2 chunkPositions

```
ChunkWorldPos [ ] BeeGame.Terrain.Chunks.LoadChunks.chunkPositions [static], [private]
```

Positions to make chunks around the player ///

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

6.31.3.3 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](#).

6.31.3.4 timer

`int` BeeGame.Terrain.Chunks.LoadChunks.timer = 0 [static], [private]

Timer for chunk removal

Definition at line 76 of file [LoadChunks.cs](#).

6.31.3.5 world

`World` BeeGame.Terrain.Chunks.LoadChunks.world

The world the player is in

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

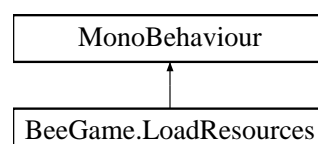
The documentation for this class was generated from the following file:

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

6.32 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

6.32.1 Detailed Description

Loads all of the resources in the game

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

6.32.2 Member Function Documentation

6.32.2.1 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
00018             Serialization.Serialization.LoadPlayerPosition (GameObject.Find ("Player") .GetComponent<Transform
00019 >());
00019
00020             SpriteDictionary.LoadSprites ();
00021             PrefabDictionary.LoadPrefabs ();
00022         }
```

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

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

6.33 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 collidson 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 is collider and render Meshes
- bool **done** = false

6.33.1 Detailed Description

The data for a [Chunks](#)'s Mesh

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

6.33.2 Member Function Documentation

6.33.2.1 AddQuadTriangles()

```
void BeeGame.Terrain.Chunks.MeshData.AddQuadTriangles (
    bool addToRenderMesh = true )
```

Adds 2 triangles to the triangle list

Parameters

<i>addToRenderMesh</i>	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     }

```

6.33.2.2 AddTriangle()

```

void BeeGame.Terrain.Chunks.MeshData.AddTriangle (
    int tri )

```

Adds a triangle to both the render and collidson meshes

Parameters

<i>tri</i>	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     }

```

6.33.2.3 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     }

```

6.33.3 Member Data Documentation

6.33.3.1 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](#).

6.33.3.2 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](#).

6.33.3.3 done

```
bool BeeGame.Terrain.Chunks.MeshData.done = false
```

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

6.33.3.4 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](#).

6.33.3.5 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](#).

6.33.3.6 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](#).

6.33.3.7 verts

```
List<Vector3> BeeGame.Terrain.Chunks.MeshData.verts = new List<Vector3>()
```

Vertices 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](#)

6.34 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](#)

6.34.1 Detailed Description

Definition at line 240 of file [Bee.cs](#).

6.34.2 Member Function Documentation

6.34.2.1 GetHashCode()

```
override int BeeGame.Items.NormalBee.GetHashCode ( )
```

Definition at line 292 of file [Bee.cs](#).

```
00293         {
00294             unchecked
00295             {
00296                 //int hashCode = 13;
00297                 var temp = $"
00298 { (int)pSpecies}{ (int)sSpecies}{ (int)pLifespan}{ (int)sLifespan}{ (int)pFertility}{ (int)sFertility}{ (int)pEffect}{ (int)sEf
00299
00300                 var hashCode = (int)(Int64.Parse(temp) ^ (127 * 13) / 159);
00301
00302                 //hashCode += ((int)pSpecies ^ (int)pLifespan ^ (int)pFertility ^ (int)pEffect ^
00303 (int)pProdSpeed) * 127;
00303                 //hashCode += ((int)sSpecies ^ (int)sLifespan ^ (int)sFertility ^ (int)sEffect ^
00304 (int)sProdSpeed) * 307;
00304
00305                 return hashCode;
00306             }
00307         }
```

6.34.3 Member Data Documentation

6.34.3.1 pEffect

```
BeeEffect BeeGame.Items.NormalBee.pEffect
```

Primary BeeEffect of the [Bee](#)

Definition at line 260 of file [Bee.cs](#).

6.34.3.2 pFertility

```
uint BeeGame.Items.NormalBee.pFertility
```

Primary Fertility of the [Bee](#)

Definition at line 256 of file [Bee.cs](#).

6.34.3.3 pLifespan

`BeeLifeSpan` `BeeGame.Items.NormalBee.pLifespan`

Primary BeeLifeSpan of the [Bee](#)

Definition at line 252 of file [Bee.cs](#).

6.34.3.4 pProdSpeed

`BeeProductionSpeed` `BeeGame.Items.NormalBee.pProdSpeed`

Primary BeeProductionSpeed of the [Bee](#)

Definition at line 264 of file [Bee.cs](#).

6.34.3.5 pSpecies

`BeeSpecies` `BeeGame.Items.NormalBee.pSpecies`

Primary BeeSpecies of the [Bee](#)

Definition at line 248 of file [Bee.cs](#).

6.34.3.6 sEffect

`BeeEffect` `BeeGame.Items.NormalBee.sEffect`

Secondary BeeGame.Enums.BeeEffect of the [Bee](#)

Definition at line 285 of file [Bee.cs](#).

6.34.3.7 sFertility

`uint` `BeeGame.Items.NormalBee.sFertility`

Secondary Fertility of the [Bee](#)

Definition at line 281 of file [Bee.cs](#).

6.34.3.8 sLifespan

`BeeLifeSpan` `BeeGame.Items.NormalBee.sLifespan`

Secondary `BeeGame.Enums.BeeLifeSpan` of the [Bee](#)

Definition at line 277 of file [Bee.cs](#).

6.34.3.9 sProdSpeed

`BeeProductionSpeed` `BeeGame.Items.NormalBee.sProdSpeed`

Secondary `BeeGame.Enums.BeeProductionSpeed` of the [Bee](#)

Definition at line 289 of file [Bee.cs](#).

6.34.3.10 sSpecies

`BeeSpecies` `BeeGame.Items.NormalBee.sSpecies`

Secondary `BeeGame.Enums.BeeSpecies` of the [Bee](#)

Definition at line 273 of file [Bee.cs](#).

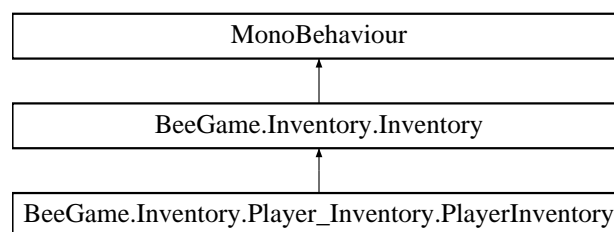
The documentation for this class was generated from the following file:

- `C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/Bee.cs`

6.35 BeeGame.Inventory.Player_Inventory.PlayerInventory Class Reference

Controls the player inventory

Inheritance diagram for `BeeGame.Inventory.Player_Inventory.PlayerInventory`:



Public Member Functions

- 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

Private Member Functions

- void [Awake](#) ()
Sets all required params for the inventory and loads ant saved versions of it
- void [SetPlayerInventory](#) ()
Set the size of the player inventory
- void [Update](#) ()
Goves the inventory update ticks
- void [OpenPlayerInventory](#) ()
Show/Hide the player inventory
- void [PickupItem](#) (ItemGameObject item)
Pickup an item and put it into the [Inventory](#)

Additional Inherited Members

6.35.1 Detailed Description

Controlls the player inventory

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

6.35.2 Member Function Documentation

6.35.2.1 Awake()

```
void BeeGame.Inventory.Player_Inventory.PlayerInventory.Awake ( ) [private]
```

Sets all required params for the inventory and loads ant saved versions of it

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

```
00024     {
00025         SetPlayerInventory();
00026         inventoryName = "PlayerInventory";
00027         Serialization.Serialization.DeSerializeInventory(this, inventoryName);
00028     }
```

6.35.2.2 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 97 of file [PlayerInventory.cs](#).

```

00098     {
00099         /** get the item
00100         outItem = GetAllItems().itemsInInventory[slotIndex];
00101
00102         if (outItem == null)
00103             return false;
00104
00105         /** if the item is placeable and is not null remove 1 from the inventory as it is assumed it is
about to be placed in the world
00106         if(outItem.placeable)
00107             RemoveItemFromInventory(slotIndex);
00108
00109         return outItem.placeable;
00110     }

```

6.35.2.3 OpenPlayerInventory()

```
void BeeGame.Inventory.Player_Inventory.PlayerInventory.OpenPlayerInventory ( ) [private]
```

Show/Hide the player inventory

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

```

00118     {
00119         if (floatingItem != null)
00120             return;
00121         thisInventoryOpen = !thisInventoryOpen;
00122         playerInventory.SetActive(!playerInventory.activeInHierarchy);
00123         THInput.isAnotherInventoryOpen = !
THInput.isAnotherInventoryOpen;
00124
00125         /** hides/shows the mouse depending on if te inventory is open or not
00126         if (playerInventory.activeInHierarchy)
00127         {
00128             Cursor.lockState = CursorLockMode.None;
00129             Cursor.visible = true;
00130         }
00131         else
00132         {
00133             Cursor.visible = false;
00134             Cursor.lockState = CursorLockMode.Locked;
00135         }
00136     }

```

6.35.2.4 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 161 of file [PlayerInventory.cs](#).

```

00162         {
00163             item.item.itemStackCount = 1;
00164
00165             /* if the item can be added to the inventory do that
00166             if (AddItemToInventory(item.item))
00167             {
00168                 /* if the item was added destroy its gameobject and save the inventory
00169                 Destroy(item.gameObject);
00170                 Serialization.Serialization.SerializeInventory(this,
inventoryName);
00171             }
00172         }

```

6.35.2.5 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 142 of file [PlayerInventory.cs](#).

```

00143         {
00144             /* if the item is already null nothing needs to be removed
00145             if (GetAllItems().itemsInInventory[index] != null)
00146             {
00147                 /* remove 1 item and if that was the last in the stack remove the item from the inventory
00148                 GetAllItems().itemsInInventory[index].
itemStackCount -= 1;
00149
00150                 if (GetAllItems().itemsInInventory[index].itemStackCount <= 0)
00151                     GetAllItems().itemsInInventory[index] = null;
00152
00153                 Serialization.Serialization.SerializeInventory(this,
inventoryName);
00154             }
00155         }

```

6.35.2.6 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 81 of file [PlayerInventory.cs](#).

```

00082         {
00083             for (int i = 0; i < slots.Length; i++)
00084             {
00085                 slots[i].selectedSlot = false;
00086             }
00087
00088             slots[index].selectedSlot = true;
00089         }

```

6.35.2.7 SetPlayerInventory()

```
void BeeGame.Inventory.Player_Inventory.PlayerInventory.SetPlayerInventory ( ) [private]
```

Set the size of the player inventory

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

```

00034         {
00035             if (!InventorySet())
00036                 SetInventorySize(36);
00037         }

```

6.35.2.8 Update()

```
void BeeGame.Inventory.Player_Inventory.PlayerInventory.Update ( ) [private]
```

Goves the inventory update ticks

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

```

00044         {
00045             UpdateBase();
00046
00047             /* checks if the inventory should be opened/closed
00048             if ((thisInventoryOpen || !playerInventory.activeInHierarchy)
00049             && !THInput.chestOpen && THInput.GetButtonDown("Player Inventory"))
00049             {
00050                 if (THInput.blockInventoryJustClosed)
00051                 {
00052                     THInput.blockInventoryJustClosed = false;
00053                     return;
00054                 }
00055                 else
00056                 {
00057                     OpenPlayerInventory();
00058                 }
00059             }
00060
00061             /* dont pickup items if the inventory is open
00062             if (THInput.isAnotherInventoryOpen)
00063                 return;
00064
00065             /* checks if somethig should be picked up and put into the inventory
00066             RaycastHit[] hit = Physics.SphereCastAll(transform.position, 1f, transform.forward);
00067
00068             for (int i = hit.Length - 1; i >= 0; i--)
00069             {
00070                 if (hit[i].collider.GetComponent<ItemGameObject>())
00071                     PickupItem(hit[i].collider.GetComponent<
00072                     ItemGameObject>());
00073             }
00074         }

```

6.35.3 Member Data Documentation

6.35.3.1 playerInventory

GameObject BeeGame.Inventory.Player_Inventory.PlayerInventory.playerInventory

Object that the inventory is

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

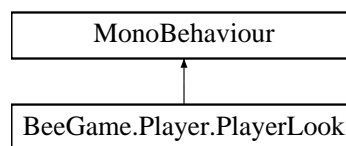
The documentation for this class was generated from the following file:

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

6.36 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

6.36.1 Detailed Description

The look for the player

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

6.36.2 Member Function Documentation

6.36.2.1 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,
rotationLock);
00074
00075         myTransform.rotation = Quaternion.Euler(0, yRot, 0);
00076         cameraTransform.localRotation = Quaternion.Euler(xRot, 0, 0);
00077     }
```

6.36.2.2 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     }
```

6.36.2.3 Update()

```
void BeeGame.Player.PlayerLook.Update ( ) [private]
```

Every fixed update check if the look should 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     }
```

6.36.3 Member Data Documentation

6.36.3.1 cameraTransform

```
Transform BeeGame.Player.PlayerLook.cameraTransform
```

Camera transform

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

6.36.3.2 myTransform

```
Transform BeeGame.Player.PlayerLook.myTransform
```

[Player](#) transform

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

6.36.3.3 rotationLock

```
float BeeGame.Player.PlayerLook.rotationLock
```

Lock for camera X rotation

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

6.36.3.4 speed

```
float BeeGame.Player.PlayerLook.speed = 5
```

Look move speed

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

6.36.3.5 xRot

```
float BeeGame.Player.PlayerLook.xRot = 0 [private]
```

Current X rotation

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

6.36.3.6 yRot

```
float BeeGame.Player.PlayerLook.yRot = 0 [private]
```

Current Y rotation

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

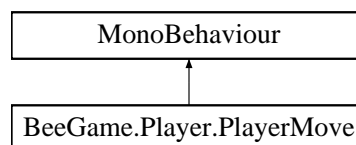
The documentation for this class was generated from the following file:

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

6.37 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

6.37.1 Detailed Description

Moves the player

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

6.37.2 Member Function Documentation

6.37.2.1 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     }
```

6.37.2.2 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 * -
gravity, 0));
00071     }
```

6.37.2.3 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,
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,
maxVelocity);
00100         velocityChange.z = Mathf.Clamp(velocityChange.z, -maxVelocity,
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,
VerticalJumpSpeed(), velocity.z);
00111         }
00112     }
```

6.37.2.4 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         }
```

6.37.2.5 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         }
```

6.37.3 Member Data Documentation

6.37.3.1 canJump

```
bool BeeGame.Player.PlayerMove.canJump = false [private]
```

Can the player jump?

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

6.37.3.2 gravity

```
float BeeGame.Player.PlayerMove.gravity = 9.81f
```

Gravity of the player

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

6.37.3.3 jumpHeight

```
float BeeGame.Player.PlayerMove.jumpHeight = 2f
```

How high can the player jump

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

6.37.3.4 maxVelocity

```
float BeeGame.Player.PlayerMove.maxVelocity = 10f
```

Max velocity of the player

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

6.37.3.5 myRigidBody

```
Rigidbody BeeGame.Player.PlayerMove.myRigidBody [private]
```

Rigidbody for the player

Definition at line 42 of file [PlayerMove.cs](#).

6.37.3.6 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](#)

6.38 BeeGame.Core.Dictionarys.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

6.38.1 Detailed Description

The prefabs available to the game

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

6.38.2 Member Function Documentation

6.38.2.1 GetPrefab()

```
static GameObject BeeGame.Core.Dictionarys.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     }
```

6.38.2.2 LoadPrefabs()

```
static void BeeGame.Core.Dictionarys.PrefabDictionary.LoadPrefabs ( ) [static]
```

Loads the prefabs into the Dictionary

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

```
00020     {
00021         prefabDictionary = Resources.Resources.GetPrefabs();
00022     }
```

6.38.3 Member Data Documentation

6.38.3.1 prefabDictionary

```
Dictionary<string, GameObject> BeeGame.Core.Dictionarys.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/Dictionarys/[PrefabDictionary.cs](#)

6.39 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

6.39.1 Detailed Description

Definition at line 210 of file [Bee.cs](#).

6.39.2 Constructor & Destructor Documentation

6.39.2.1 QueenBee() [1/2]

```
BeeGame.Items.QueenBee.QueenBee ( )
```

Definition at line 222 of file [Bee.cs](#).

```
00222 { }
```

6.39.2.2 QueenBee() [2/2]

```
BeeGame.Items.QueenBee.QueenBee (
    NormalBee princess,
    NormalBee drone )
```

Definition at line 224 of file [Bee.cs](#).

```
00225     {
00226         this.queen = princess;
00227         this.drone = drone;
00228     }
```

6.39.3 Member Function Documentation

6.39.3.1 GetHashCode()

```
override int BeeGame.Items.QueenBee.GetHashCode ( )
```

Definition at line 230 of file [Bee.cs](#).

```
00231     {
00232         unchecked
00233         {
00234             return (int)Int64.Parse($"{queen.GetHashCode()} {drone.GetHashCode()}");
00235         }
00236     }
```

6.39.4 Property Documentation

6.39.4.1 drone

[NormalBee](#) BeeGame.Items.QueenBee.drone [get], [set]

Paired drone traits

Definition at line 220 of file [Bee.cs](#).

6.39.4.2 queen

[NormalBee](#) BeeGame.Items.QueenBee.queen [get], [set]

Original princess traits

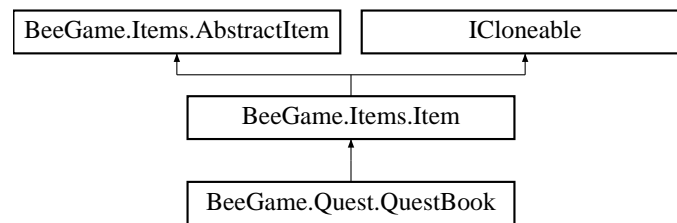
Definition at line 216 of file [Bee.cs](#).

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

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

6.40 BeeGame.Quest.QuestBook Class Reference

Inheritance diagram for BeeGame.Quest.QuestBook:



Public Member Functions

- [QuestBook](#) ()
- override bool [InteractWithObject](#) ()
- override Sprite [GetItemSprite](#) ()
Returns the sprite for the item
- override int [GetHashCode](#) ()
Returns the hashcode for this [Item](#)

Public Attributes

- override int [maxStackCount](#) => 1

Additional Inherited Members

6.40.1 Detailed Description

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

6.40.2 Constructor & Destructor Documentation

6.40.2.1 QuestBook()

`BeeGame.Quest.QuestBook.QuestBook ()`

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

```

00015                                     : base("Quest Book")
00016     {
00017
00018     }
```

6.40.3 Member Function Documentation

6.40.3.1 GetHashCode()

```
override int BeeGame.Quest.QuestBook.GetHashCode ( ) [virtual]
```

Returns the hashcode for this [Item](#)

Returns

10

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

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

```
00035         {  
00036             return 10;  
00037         }
```

6.40.3.2 GetItemSprite()

```
override Sprite BeeGame.Quest.QuestBook.GetItemSprite ( ) [virtual]
```

Returns the sprite for the item

Returns

Sprite for this item

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

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

```
00026         {  
00027             return SpriteDictionary.GetSprite("TestSprite");  
00028         }
```

6.40.3.3 InteractWithObject()

```
override bool BeeGame.Quest.QuestBook.InteractWithObject ( ) [virtual]
```

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

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

```
00021         {  
00022             return true;  
00023         }
```

6.40.4 Member Data Documentation

6.40.4.1 maxStackCount

```
override int BeeGame.Quest.QuestBook.maxStackCount => 1
```

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

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

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

6.41 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](#) ()
- static Dictionary< string, GameObject > [GetPrefabs](#) ()

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](#)

6.41.1 Detailed Description

A strongly-typed resource class, for looking up localized strings, etc.

Definition at line 26 of file [Resources.Designer.cs](#).

6.41.2 Constructor & Destructor Documentation

6.41.2.1 Resources()

BeeGame.Resources.Resources.Resources () [package]

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

```
00033         {
00034     }
```

6.41.3 Member Function Documentation

6.41.3.1 GetPrefabs()

static Dictionary<string, GameObject> BeeGame.Resources.Resources.GetPrefabs () [static],
[package]

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

```
00119     {
00120         string[] splitCharacters = new string[] { ",", " " };
00121         object obj = ResourceManager.GetObject("Prefabs",
resourceCulture);
00122
00123         string text = System.Text.Encoding.Default.GetString((byte[])obj);
00124         text = text.Remove(0, 3);
00125         string lineText = "";
00126         string[] splitText;
00127         Dictionary<string, GameObject> objects = new Dictionary<string, GameObject>();
00128
00129         for (int i = 0; i < text.Length; i++)
00130         {
00131             if(text[i] != '\n')
00132             {
00133                 lineText += text[i];
00134             }
00135             else
00136             {
00137                 splitText = lineText.Split(splitCharacters, StringSplitOptions.RemoveEmptyEntries);
00138                 lineText = "";
00139                 objects.Add(splitText[0], UnityEngine.Resources.Load("Prefabs/" + splitText[
1].Remove(splitText[1].Length - 1, 1)) as GameObject);
00140             }
00141         }
00142
00143         splitText = lineText.Split(splitCharacters, StringSplitOptions.RemoveEmptyEntries);
00144         lineText = "";
00145         objects.Add(splitText[0], UnityEngine.Resources.Load("Prefabs/" + splitText[1]) as
GameObject);
00146
00147         return objects;
00148     }
```

6.41.3.2 GetSprites()

static Dictionary<string, Sprite> BeeGame.Resources.Resources.GetSprites () [static], [package]

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

```

00085     {
00086         string[] splitCharacters = new string[] { ",", " " };
00087         object obj = ResourceManager.GetObject("Sprites",
resourceCulture);
00088
00089         string text = System.Text.Encoding.Default.GetString((byte[])obj);
00090         string lineText = "";
00091         string[] splitText;
00092         Texture2D tex;
00093         Dictionary<string, Sprite> sprites = new Dictionary<string, Sprite>();
00094
00095         for (int i = 0; i < text.Length; i++)
00096         {
00097             if (text[i] != '\n')
00098             {
00099                 lineText += text[i];
00100             }
00101             else
00102             {
00103                 splitText = lineText.Split(splitCharacters, StringSplitOptions.RemoveEmptyEntries);
00104                 lineText = "";
00105                 tex = UnityEngine.Resources.Load("Sprites/" + splitText[1].Remove(splitText[
1].Length - 1, 1)) as Texture2D;
00106                 sprites.Add(splitText[0], Sprite.Create(tex, new UnityEngine.Rect(0, 0, tex.
width, tex.height), Vector2.zero));
00107             }
00108         }
00109
00110         splitText = lineText.Split(splitCharacters, StringSplitOptions.RemoveEmptyEntries);
00111         lineText = "";
00112         tex = UnityEngine.Resources.Load("Sprites/" + splitText[1]) as Texture2D;
00113         sprites.Add(splitText[0], Sprite.Create(tex, new UnityEngine.Rect(0, 0, tex.width,
tex.height), Vector2.zero));
00114
00115         return sprites;
00116     }

```

6.41.4 Member Data Documentation

6.41.4.1 resourceCulture

global.System.Globalization.CultureInfo BeeGame.Resources.Resources.resourceCulture [static], [private]

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

6.41.4.2 resourceMan

global.System.Resources.ResourceManager BeeGame.Resources.Resources.resourceMan [static], [private]

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

6.41.5 Property Documentation

6.41.5.1 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](#).

6.41.5.2 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](#).

6.41.5.3 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](#).

6.41.5.4 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](#)↵

6.42 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

6.42.1 Detailed Description

Saves a [Chunks](#) modified Blocks for save optimisation

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

6.42.2 Constructor & Destructor Documentation

6.42.2.1 SaveChunk()

```
BeeGame.Terrain.Chunks.SaveChunk.SaveChunk (
    Block blockArray[, ] )
```

Will search all the the given Blocks for modified blocks

Parameters

blockArray	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     }
```

6.42.3 Member Data Documentation

6.42.3.1 blocks

```
Dictionary<ChunkWorldPos, Block> BeeGame.Terrain.Chunks.SaveChunk.blocks = new Dictionary<Chunk↵  
WorldPos, Block>()
```

Blocks to be saved

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

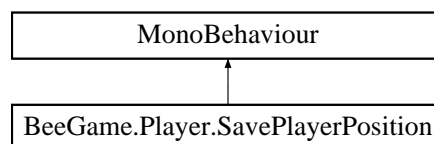
The documentation for this class was generated from the following file:

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

6.43 BeeGame.Player.SavePlayerPosition Class Reference

Saves the player postion

Inheritance diagram for BeeGame.Player.SavePlayerPosition:



Private Member Functions

- void [Update](#) ()
Saves the player every 1000 frames

Private Attributes

- int [counter](#) = 0
Timer for saveing the player

6.43.1 Detailed Description

Saves the player postion

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

6.43.2 Member Function Documentation

6.43.2.1 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
00028          counter--;
00029      }
```

6.43.3 Member Data Documentation

6.43.3.1 counter

```
int BeeGame.Player.SavePlayerPosition.counter = 0 [private]
```

Timer for saving the player

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

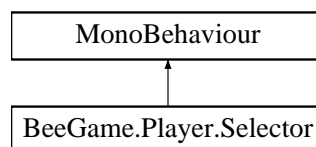
The documentation for this class was generated from the following file:

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

6.44 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](#) ()
Chanages 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

6.44.1 Detailed Description

Moves the Block selector

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

6.44.2 Member Function Documentation

6.44.2.1 Awake()

```
void BeeGame.Player.Selector.Awake ( ) [private]
```

Make the selector

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

```
00048         {
00049             selector = Instantiate(selector);
00050         }
```

6.44.2.2 BreakBlock()

```
void BeeGame.Player.Selector.BreakBlock ( ) [private]
```

Breaks the Block in the selectors postion

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)
selector.transform.position.y, (int)selector.transform.position.z);
00128
00129         if (!block.breakable)
00130             return;
00131
00132         chunk.world.SetBlock((int)selector.transform.position.x, (int)
selector.transform.position.y, (int)selector.transform.position.z, new
Air(), true);
00133         /* set to changed so when block is placed down again it will be saved
00134         block.changed = true;
00135         block.BreakBlock(selector.transform.position);
00136     }
```

6.44.2.3 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     }
```

6.44.2.4 PlaceBlock()

```
void BeeGame.Player.Selector.PlaceBlock ( ) [private]
```

Places s Block in the selector postion

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         if (!chunk.GetBlock((int)selector.transform.position.x - chunk.
chunkWorldPos.x, (int)selector.transform.position.y - chunk.
chunkWorldPos.y, (int)selector.transform.position.z - chunk.
chunkWorldPos.z).InteractWithBlock(
playerInventory))
00149             /* gets the item in the hotbar and if the item is placeable place it
00150             if (transform.parent.GetComponentInChildren<PlayerInventory>().
GetItemFromHotBar(selectedHotbarSlot, out
Item blockToPlace))
00151                 chunk.world.SetBlock((int)(selector.transform.position.x +
hit.normal.x), (int)(selector.transform.position.y + hit.normal.y), (int)(
selector.transform.position.z + hit.normal.z), (Block)blockToPlace.CloneObject(), true);
00152     }
```


6.44.2.5 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);
00116     }
```

6.44.2.6 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     }
```

6.44.2.7 UpdateSelector()

```
void BeeGame.Player.Selector.UpdateSelector ( ) [private]
```

Updates the 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         SelectedSlot();
00095     }
```

6.44.3 Member Data Documentation

6.44.3.1 hit

RaycastHit BeeGame.Player.Selector.hit [private]

Where the raycast hit

Definition at line 35 of file [Selector.cs](#).

6.44.3.2 layers

LayerMask BeeGame.Player.Selector.layers

Layers for the selector to look at

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

6.44.3.3 playerInventory

[PlayerInventory](#) BeeGame.Player.Selector.playerInventory

[Player Inventory](#)

Definition at line 26 of file [Selector.cs](#).

6.44.3.4 selectedHotbarSlot

int BeeGame.Player.Selector.selectedHotbarSlot = 27

What slot in the hotbar is selected

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

6.44.3.5 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](#)

6.45 BeeGame.Serialization.Serialization Class Reference

Serializes and Deserialises things

Static Public Member Functions

- static void [MakeDirectorys](#) ()
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 playerTransfom)
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 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

6.45.1 Detailed Description

Serializes and Deserialises things

Binary serialization is SLOW try to only serialize only what is absolutly necessary

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

6.45.2 Member Function Documentation

6.45.2.1 DeleteFile()

```
static void BeeGame.Serialization.Serialization.DeleteFile (
    string fileName ) [static]
```

Deletes the given file if it exists, Starts in Application.dataPath

Parameters

<i>fileName</i>	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.
RemoveEmptyEntries);
00060
00061             if(temp[temp.Length - 1] == fileName)
00062             {
00063                 File.Delete(file[i]);
00064
00065                 return;
00066             }
00067         }
00068     }
```

6.45.2.2 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 }
```

6.45.2.3 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     }
```

6.45.2.4 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     }

```

6.45.2.5 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 241 of file [Serialization.cs](#).

```

00242     {
00243         BinaryFormatter bf = new BinaryFormatter();
00244         FileStream fs = new FileStream(file, FileMode.Open);
00245
00246         try
00247         {
00248             return bf.Deserialize(fs);
00249         }
00250         catch (SerializationException e)
00251         {
00252             Debug.Log($"Deserialization Exception {e}");
00253             throw new SerializationException();
00254         }
00255         finally
00256         {
00257             fs.Close();
00258         }
00259     }

```

6.45.2.6 LoadPlayerPosition()

```
static void BeeGame.Serialization.Serialization.LoadPlayerPosition (
    Transform playerTransform ) [static]
```

Loads the players position, rotation, 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         playerTransform.position = pos[0];
00102         playerTransform.rotation = (Quaternion)pos[1];
00103         playerTransform.localScale = pos[2];
00104     }
```

6.45.2.7 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     }
```

6.45.2.8 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     }

```

6.45.2.9 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 216 of file [Serialization.cs](#).

```

00217     {
00218         BinaryFormatter bf = new BinaryFormatter();
00219         FileStream fs = new FileStream(file, FileMode.OpenOrCreate);
00220
00221         try
00222         {
00223             bf.Serialize(fs, obj);
00224         }
00225         catch (SerializationException e)
00226         {
00227             Debug.Log($"Serialization Exception: {e}");
00228             throw new SerializationException();
00229         }
00230         finally
00231         {
00232             fs.Close();
00233         }
00234     }

```

6.45.2.10 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     }

```

6.45.2.11 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     }

```

6.45.3 Member Data Documentation

6.45.3.1 saveFolderName

```
string BeeGame.Serialization.Serialization.saveFolderName = "Saves" [static]
```

Save folder

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

6.45.3.2 savePath

```
string BeeGame.Serialization.Serialization.savePath [static], [private]
```

Path to save things

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

6.45.3.3 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](#)

6.46 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)

6.46.1 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](#).

6.46.2 Member Function Documentation

6.46.2.1 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     }
```

6.46.2.2 Generate() [1/3]

```
static float BeeGame.Terrain.LandGeneration.Noise.SimplexNoise.Generate (
    float x ) [static]
```

1D simplex noise

Parameters

<i>x</i>	
----------	--

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     }

```

6.46.2.3 Generate() [2/3]

```

static float BeeGame.Terrain.LandGeneration.Noise.SimplexNoise.Generate (
    float x,
    float y ) [static]

```

2D simplex noise

Parameters

<i>x</i>	
<i>y</i>	

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 + i1 + 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     }

```

6.46.2.4 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 = 0.3333333333f;
00145         const float G3 = 0.1666666667f;
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 + 1 + perm[jj + 1 + 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 }

```

6.46.2.5 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     }

```

6.46.2.6 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     }
```

6.46.2.7 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     }
```

6.46.2.8 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     }
```

6.46.2.9 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     }
```

6.46.3 Member Data Documentation

6.46.3.1 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
}
```

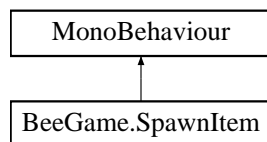
Definition at line 244 of file [SimplexNoise.cs](#).

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](#)

6.47 BeeGame.SpawnItem Class Reference

Inheritance diagram for BeeGame.SpawnItem:



Private Member Functions

- void [Start](#) ()
- void [OnDrawGizmos](#) ()

6.47.1 Detailed Description

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

6.47.2 Member Function Documentation

6.47.2.1 OnDrawGizmos()

```
void BeeGame.SpawnItem.OnDrawGizmos ( ) [private]
```

Definition at line 49 of file [SpawnItem.cs](#).

```
00050     {
00051         //Gizmos.color = Color.green;
00052         //Gizmos.DrawSphere(transform.position, 0.5f);
00053     }
```

6.47.2.2 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
GameObject, transform.position, Quaternion.identity) as GameObject;
00017             go.GetComponent<ItemGameObject>().item = new Bee(
BeeType.DRONE, new NormalBee() { pSpecies = BeeSpecies.FOREST });
00018
00019             go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
transform.position, Quaternion.identity) as GameObject;
00020             go.GetComponent<ItemGameObject>().item = new Bee(
BeeType.PRINCESS, new NormalBee() { pSpecies = BeeSpecies.FOREST });
00021
00022             go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
transform.position, Quaternion.identity) as GameObject;
00023             go.GetComponent<ItemGameObject>().item = new Bee(
BeeType.DRONE, new NormalBee() { pSpecies = BeeSpecies.COMMON, sSpecies =
BeeSpecies.COMMON });
00024
00025             go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
transform.position, Quaternion.identity) as GameObject;
00026             go.GetComponent<ItemGameObject>().item = new Bee(
BeeType.PRINCESS, new NormalBee() { pSpecies = BeeSpecies.COMMON, sSpecies =
BeeSpecies.COMMON });
00027
00028             //go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
transform.position, Quaternion.identity) as GameObject;
00029             //go.GetComponent<ItemGameObject>().item = new Bee(BeeType.QUEEN, new QueenBee());
00030
00031             go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
transform.position, Quaternion.identity) as GameObject;
00032             go.GetComponent<ItemGameObject>().item = new HoneyComb(
HoneyCombType.ICEY);
00033
00034             go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
transform.position, Quaternion.identity) as GameObject;
00035             go.GetComponent<ItemGameObject>().item = new HoneyComb(
HoneyCombType.HONEY);
00036
00037             go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
transform.position, Quaternion.identity) as GameObject;
00038             go.GetComponent<ItemGameObject>().item = new Chest();
00039             go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
transform.position, Quaternion.identity) as GameObject;
00040             go.GetComponent<ItemGameObject>().item = new Chest();
00041
00042             go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
transform.position, Quaternion.identity) as GameObject;
00043             go.GetComponent<ItemGameObject>().item = new Apiary();
00044
00045             go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
transform.position, Quaternion.identity) as GameObject;
00046             go.GetComponent<ItemGameObject>().item = new
CraftingTable();
00047         }

```

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

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

6.48 BeeGame.Core.Dictionary.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

6.48.1 Detailed Description

All of the sprites available to the game

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

6.48.2 Member Function Documentation

6.48.2.1 GetSprite()

```
static Sprite BeeGame.Core.Dictionarys.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         return sprite;
00028     }
00029 }
```

6.48.2.2 LoadSprites()

```
static void BeeGame.Core.Dictionarys.SpriteDictionary.LoadSprites ( ) [static]
```

Loads the sprites into the dictionary

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

```
00035     {
00036         itemSpriteDictionary = Resources.Resources.GetSprites();
00037     }
```

6.48.3 Member Data Documentation

6.48.3.1 itemSpriteDictionary

```
Dictionary<string, Sprite> BeeGame.Core.Dictionarys.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/Dictionarys/[SpriteDictionary.cs](#)

6.49 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 postion from a THVector3
- static [THVector3](#) [GetBlockPos](#) ([RaycastHit](#) hit)
Returns the positon 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 positon
- 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

6.49.1 Detailed Description

Should use as an interface between the rest of the game and the terrain

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

6.49.2 Member Function Documentation

6.49.2.1 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     }
```

6.49.2.2 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     }

```

6.49.2.3 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     }

```

6.49.2.4 GetBlockPos() [1/3]

```

static ChunkWorldPos BeeGame.Terrain.LandGeneration.Terrain.GetBlockPos (
    THVector3 pos ) [static]

```

Gets a block position from a THVector3

Parameters

<i>pos</i>	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 }

```

6.49.2.5 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 adjacent 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     }

```

6.49.2.6 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     }
```

6.49.2.7 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>	
------------	--

Returns

Definition 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     }
```

6.49.2.8 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     }
```


6.49.2.9 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         return pos;
00193     }
00194 }
```

6.49.2.10 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 really 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
00102             /** if negative nomrmal
00103             if (normal < 0)
00104             {
00105                 pos = (int)pos;
00106                 return pos;
00107             }
00108             /** if their is no normal
00109
00110             /** if pos is greater than 0.5 we are in the next block so go to it
00111             if ((-pos - (int)-pos) > 0.5)
00112             {
00113                 return (int)pos - 1;
00114             }
00115
00116             return (int)pos;
00117         }
00118     }

```

6.49.2.11 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 their 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             if((int)pos % 2 == 0)
00141                 return Mathf.RoundToInt((float)Math.Round(pos, 1)) - normal;
00142             return Mathf.RoundToInt((float)Math.Round(pos, 1));
00143         }
00144         if(pos <= 0)
00145         {
00146             if (normal > 0)
00147             {
00148                 if ((int)pos % 2 == 0)
00149                     /* the Math.Round removes strange rounding errors shown with Mathf.Round eg
00150                     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         return Mathf.RoundToInt((float)Math.Round(pos, 1));
00159     }
00160 }

```

6.49.2.12 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 tryign 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     }

```

6.49.3 Member Data Documentation**6.49.3.1 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`

6.50 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 heigh 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
Desity of trees
- float [caveFrequency](#) = 0.025f
How often do caves happen
- int [caveSize](#) = 8
Threashold for makeing a cave

6.50.1 Detailed Description

Generates the terrain for the game

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

6.50.2 Member Function Documentation

6.50.2.1 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
00086                 thread.Start();
00087                 return outChunk;
00088             }
00089         }

```

6.50.2.2 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 = GenChunkColumn(chunk, x, z);
00104                 }
00105             }
00106
00107             chunk.SetBlocksUnmodified();
00108             outChunk = chunk;
00109         }

```

6.50.2.3 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     }
```

6.50.2.4 GenChunkColumn()

```
Chunk BeeGame.Terrain.LandGeneration.TerrainGeneration.GenChunkColumn (
    Chunk chunk,
    int x,
    int z )
```

Generates a column 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 column of 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 column is currently too 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 column 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     }

```

6.50.2.5 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 should be 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 *
scale) + 1f) * (max / 2f));
00176     }
```

6.50.2.6 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 there then sets it
00195         if (Chunk.InRange(x) && Chunk.InRange(y) &&
Chunk.InRange(z))
00196             if (replacesBlocks || chunk.blocks[x, y, z] == null)
00197                 chunk.SetBlock(x, y, z, block, false);
00198     }
```

6.50.3 Member Data Documentation

6.50.3.1 caveFrequency

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.caveFrequency = 0.025f [private]
```

How often do caves happen

Definition at line 67 of file [TerrainGeneration.cs](#).

6.50.3.2 caveSize

```
int BeeGame.Terrain.LandGeneration.TerrainGeneration.caveSize = 8 [private]
```

Threshold for makeing a cave

Definition at line 71 of file [TerrainGeneration.cs](#).

6.50.3.3 dirtBaseHeight

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.dirtBaseHeight = 1 [private]
```

Where does dirt start

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

6.50.3.4 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](#).

6.50.3.5 dirtNoiseHeight

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.dirtNoiseHeight = 3 [private]
```

How tall dirt can be

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

6.50.3.6 stoneBaseHeight

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.stoneBaseHeight = -24 [private]
```

Base height of stone

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

6.50.3.7 stoneBaseNoise

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.stoneBaseNoise = 0.05f [private]
```

Base noise of stone

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

6.50.3.8 stoneBaseNoiseHeight

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.stoneBaseNoiseHeight = 4 [private]
```

Base noise heigh for stone

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

6.50.3.9 stoneMinHeight

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.stoneMinHeight = -12 [private]
```

Minimun height for stone

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

6.50.3.10 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](#).

6.50.3.11 stoneMountainHeight

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.stoneMountainHeight = 48 [private]
```

Base height for a mountain

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

6.50.3.12 treeDensity

```
int BeeGame.Terrain.LandGeneration.TerrainGeneration.treeDensity = 3 [private]
```

Desity of trees

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

6.50.3.13 treeFrequency

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.treeFrequency = 0.2f [private]
```

Frequency of trees

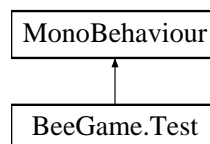
Definition at line 58 of file [TerrainGeneration.cs](#).

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

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

6.51 BeeGame.Test Class Reference

Inheritance diagram for BeeGame.Test:



Public Member Functions

- void [Print](#) ([Item](#) item)

Private Member Functions

- void [Start](#) ()

6.51.1 Detailed Description

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

6.51.2 Member Function Documentation

6.51.2.1 Print()

```
void BeeGame.Test.Print (
    Item item )
```

6.51.2.2 Start()

```
void BeeGame.Test.Start ( ) [private]
```

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

```
00017         {
00018             CraftingRecipies.AddShapedRecipie(new object[] { " ", " X ",
00019 " ", "X", Dirt.ID }, new Grass());
00019             CraftingRecipies.AddShaplessRecipie(new object[] { new
Grass(), 1 }, new Dirt());
00020
00021             Events.shapedRecipieCrafted += Print;
00022         }
```

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

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

6.52 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

6.52.1 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](#).

6.52.2 Member Function Documentation

6.52.2.1.GetAxis()

```
static int BeeGame.Core.THInput.GetAxis (
    string axis ) [static]
```

Gets the axis of a button press

Parameters

<i>axis</i>	Axis to check, Horizontal or Vertical
-------------	---------------------------------------

Returns

+1 or -1

Definition at line 140 of file [THInput.cs](#).

```
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     }

```

6.52.2.2 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 eg "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[] array:
00090                 /*for each possible key, check if it was pressed and if it was return that it was, if
00091                 none of them was pressed return false
00092                 foreach (var item in array)
00093                 {
00094                     if (Input.GetKey(item))
00095                     {
00096                         return true;
00097                     }
00098                 }
00099                 return false;
00100             default:
00101                 return Input.GetKey((KeyCode)inputButtons[button]);
00102         }
00103     }

```


6.52.2.3 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         switch (inputButtons[button])
00057         {
00058             case KeyCode[] array:
00059                 /*for each possible key, check if it was pressed and if it was return that it was, if
00060                 none of them was pressed return false
00061                 foreach (var item in array)
00062                 {
00063                     if (Input.GetKeyDown(item))
00064                     {
00065                         return true;
00066                     }
00067                 }
00068                 return false;
00069             default:
00070                 return Input.GetKeyDown((KeyCode)inputButtons[button]);
00071         }
00072     }
00073 }
```

6.52.2.4 GetButtonUp()

```
static bool BeeGame.Core.THInput.GetButtonUp (
    string button ) [static]
```

Has the given button been released this update

Parameters

<i>button</i>	Button name eg "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[] array:
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 array)
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         }

```

6.52.3 Member Data Documentation**6.52.3.1 blockInventoryJustClosed**

```
bool BeeGame.Core.THInput.blockInventoryJustClosed [static]
```

Was a Block inventory just closed

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

6.52.3.2 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](#).

6.52.3.3 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](#).

6.52.3.4 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/[THInput.cs](#)

6.53 BeeGame.Core.UnityTypeReplacements.THQuaternion Struct Reference

Public Attributes

- float [x](#)
- float [y](#)
- float [z](#)
- float [w](#)

6.53.1 Detailed Description

Definition at line 8 of file [THQuaternion.cs](#).

6.53.2 Member Data Documentation

6.53.2.1 w

`float BeeGame.Core.UnityTypeReplacements.THQuaternion.w`

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

6.53.2.2 x

`float BeeGame.Core.UnityTypeReplacements.THQuaternion.x`

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

6.53.2.3 y

`float BeeGame.Core.UnityTypeReplacements.THQuaternion.y`

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

6.53.2.4 z

`float BeeGame.Core.UnityTypeReplacements.THQuaternion.z`

Definition at line 12 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/THQuaternion.cs](#)

6.54 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

6.54.1 Detailed Description

Serializable version of Vector2

Definition at line 10 of file THVector2.cs.

6.54.2 Constructor & Destructor Documentation

6.54.2.1 THVector2() [1/3]

```
BeeGame.Core.THVector2.THVector2 (
    float x,
    float y )
```

Constructor from 2 floats

Parameters

<code>x</code>	X position
<code>y</code>	Y position

Definition at line 29 of file [THVector2.cs](#).

```
00030      {
00031          this.x = x;
00032          this.y = y;
00033      }
```

6.54.2.2 THVector2() [2/3]

```
BeeGame.Core.THVector2.THVector2 (
    THVector2 vec2 )
```

Constructor from another [THVector2](#)

Parameters

<code>vec2</code>	Vector to make this from
-------------------	--------------------------

Definition at line 39 of file [THVector2.cs](#).

```
00040      {
00041          this = vec2;
00042      }
```

6.54.2.3 THVector2() [3/3]

```
BeeGame.Core.THVector2.THVector2 (
    Vector2 vec2 )
```

Constructor from Vector2

Parameters

<code>vec2</code>	Vector to make this from
-------------------	--------------------------

Definition at line 48 of file [THVector2.cs](#).

```
00049      {
00050          this = vec2;
00051      }
```

6.54.3 Member Function Documentation

6.54.3.1 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     }
```

6.54.3.2 GetHashCode()

```
override int BeeGame.Core.THVector2.GetHashCode ( )
```

Definition at line 64 of file [THVector2.cs](#).

```
00065     {
00066         unchecked
00067         {
00068             int hash = 13;
00069
00070             hash *= 443 * x.GetHashCode();
00071             hash *= 373 * y.GetHashCode();
00072
00073             return hash;
00074         }
00075     }
```

6.54.3.3 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     }
```

6.54.3.4 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     }
```

6.54.3.5 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     }
```

6.54.3.6 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     }
```

6.54.3.7 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     }
```

6.54.3.8 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     }
```


6.54.3.9 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     }
```

6.54.3.10 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     }
```

6.54.3.11 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     }
```

6.54.3.12 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     }
```

6.54.3.13 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     }
```

6.54.3.14 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     }
```

6.54.3.15 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
00150         return a;
00151     }
```

6.54.3.16 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
00157         return a;
00158     }
```

6.54.3.17 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     }
```

6.54.3.18 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     }
```

6.54.3.19 ToString()

```
override string BeeGame.Core.THVector2.ToString ( )
```

Definition at line 77 of file [THVector2.cs](#).

```
00078     {
00079         return $"{x}, {y}";
00080     }
```

6.54.4 Member Data Documentation

6.54.4.1 x

```
float BeeGame.Core.THVector2.x
```

X position

Definition at line 16 of file [THVector2.cs](#).

6.54.4.2 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](#)

6.55 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)
- Multiplies vector a and b*
- static [THVector3 operator*](#) ([THVector3](#) a, float b)
- Multiplies b to vector a*
- static [THVector3 operator*](#) (float a, [THVector3](#) b)
- Multiplies a to vector b*
- static [THVector3 operator/](#) ([THVector3](#) a, [THVector3](#) b)
- Divides vector a and b*
- static [THVector3 operator/](#) ([THVector3](#) a, float b)
- Divides a by b*
- static [THVector3 operator/](#) (float a, [THVector3](#) b)
- Divides b by 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

6.55.1 Detailed Description

Serializable version of Vector3

Definition at line 10 of file [THVector3.cs](#).

6.55.2 Constructor & Destructor Documentation

6.55.2.1 [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      }
```

6.55.2.2 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      }
```

6.55.2.3 THVector3() [3/4]

```
BeeGame.Core.THVector3.THVector3 (
    Vector3 vec3 )
```

Constructor from another [Vector3](#)

Parameters

<i>vec3</i>	Vector to make this from
-------------	--------------------------

Definition at line 54 of file [THVector3.cs](#).

```
00055      {
00056          this = vec3;
00057      }
```

6.55.2.4 THVector3() [4/4]

```
BeeGame.Core.THVector3.THVector3 (
    Terrain.ChunkWorldPos vec3 )
```

Constructor from another [Terrain.ChunkWorldPos](#)

Parameters

<i>vec3</i>	Vector to make this from
-------------	--------------------------

Definition at line 63 of file [THVector3.cs](#).

```
00064     {
00065         this = vec3;
00066     }
```

6.55.3 Member Function Documentation

6.55.3.1 Distance()

```
static float BeeGame.Core.THVector3.Distance (
    THVector3 a,
    THVector3 b ) [static]
```

Distance between 2 vectors

Parameters

<i>a</i>	First Vector
<i>b</i>	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));
00079     }
```

6.55.3.2 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      }

```

6.55.3.3 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              return hash;
00112          }
00113      }

```

6.55.3.4 operator Quaternion()

```
static BeeGame.Core.THVector3.operator Quaternion (
    THVector3 vec3 ) [explicit], [static]
```

Converts a [THVector3](#) to a Quaternion

Parameters

<i>vec3</i>	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         }
```

6.55.3.5 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         }
```

6.55.3.6 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         }
```

6.55.3.7 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 $a \neq b$

Definition at line 140 of file [THVector3.cs](#).

```
00141      {
00142          return !(a == b);
00143      }
```

6.55.3.8 operator*() [1/3]

```
static THVector3 BeeGame.Core.THVector3.operator* (
    THVector3 a,
    THVector3 b ) [static]
```

Multiplies vector a and b

Parameters

<i>a</i>	Vector a
<i>b</i>	Vector b

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      }
```

6.55.3.9 operator*() [2/3]

```
static THVector3 BeeGame.Core.THVector3.operator* (
    THVector3 a,
    float b ) [static]
```

Multiplies b to vector a

Parameters

<i>a</i>	Vector a
<i>b</i>	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     }
```

6.55.3.10 operator*() [3/3]

```
static THVector3 BeeGame.Core.THVector3.operator* (
    float a,
    THVector3 b ) [static]
```

Multiplies a to vector b

Parameters

<i>a</i>	Vector a
<i>b</i>	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     }
```

6.55.3.11 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     }

```

6.55.3.12 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     }

```

6.55.3.13 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         }
```

6.55.3.14 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 and 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             return a;  
00195         }
```

6.55.3.15 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 and 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  }
```

6.55.3.16 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 and b

Definition at line 217 of file [THVector3.cs](#).

```
00218      {  
00219          return new THVector3(a - b.x, a - b.y, a - b.z);  
00220      }
```

6.55.3.17 operator/() [1/3]

```
static THVector3 BeeGame.Core.THVector3.operator/ (  
    THVector3 a,  
    THVector3 b ) [static]
```

Divides vector a and b

Parameters

<i>a</i>	Vector a
<i>b</i>	Vector b

Returns

returns new vector that is the division of a and 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     }
```

6.55.3.18 operator/() [2/3]

```

static THVector3 BeeGame.Core.THVector3.operator/ (
    THVector3 a,
    float b ) [static]
```

Divides a by b

Parameters

<i>a</i>	Vector a
<i>b</i>	float b

Returns

returns new vector that is the division of a and 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     }
```

6.55.3.19 operator/() [3/3]

```

static THVector3 BeeGame.Core.THVector3.operator/ (
    float a,
    THVector3 b ) [static]
```

Divides b by a

Parameters

<i>a</i>	Vector a
<i>b</i>	float b

Returns

returns new vector that is the division of a and b

Definition at line 293 of file [THVector3.cs](#).

```
00294         {  
00295             return new THVector3(a / b.x, a / b.y, a / b.z);  
00296         }
```

6.55.3.20 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         }
```

6.55.3.21 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         }
```


6.55.4 Member Data Documentation

6.55.4.1 x

```
float BeeGame.Core.THVector3.x
```

X position

Definition at line 16 of file [THVector3.cs](#).

6.55.4.2 y

```
float BeeGame.Core.THVector3.y
```

Y postion

Definition at line 20 of file [THVector3.cs](#).

6.55.4.3 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/[THVector3.cs](#)

6.56 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

6.56.1 Detailed Description

Position of the items texture

Definition at line 411 of file [Item.cs](#).

6.56.2 Member Data Documentation

6.56.2.1 x

```
int BeeGame.Items.Tile.x
```

X pos of the texture

Definition at line 416 of file [Item.cs](#).

6.56.2.2 y

```
int BeeGame.Items.Tile.y
```

Y pos of the texture

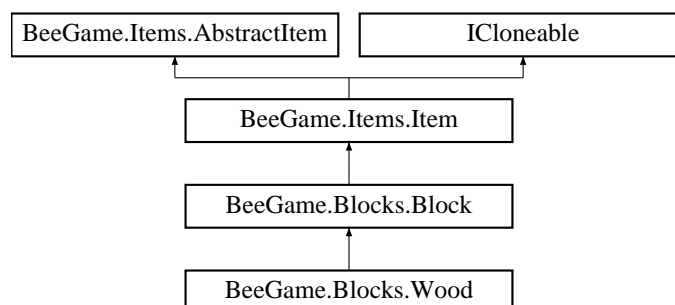
Definition at line 420 of file [Item.cs](#).

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

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

6.57 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 postion 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

6.57.1 Detailed Description

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

6.57.2 Constructor & Destructor Documentation

6.57.2.1 Wood()

```
BeeGame.Blocks.Wood.Wood ( )
```

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

```
00017                                     : base("Wood")
00018     {
00019
00020     }
```

6.57.3 Member Function Documentation

6.57.3.1 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      }
```

6.57.3.2 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      }
```

6.57.3.3 TexturePosition()

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

Texture postion 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         }
```

6.57.3.4 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         }
```

6.57.4 Member Data Documentation**6.57.4.1 ID**

```
new int BeeGame.Blocks.Wood.ID => 5 [static]
```

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

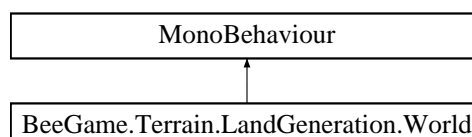
The documentation for this class was generated from the following file:

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

6.58 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 at 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 the 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

6.58.1 Detailed Description

Allows inter Chunk communication as it stores a list of active chunks

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

6.58.2 Member Function Documentation

6.58.2.1 CreateChunk()

```
void BeeGame.Terrain.LandGeneration.World.CreateChunk (
    int x,
    int y,
    int z )
```

Creates a chunk at the given x, y, z

Parameters

x	X pos to make the new chunk
y	Y pos to make the new chunk
z	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.
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 nwe 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 unecisary 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     }

```

6.58.2.2 DestroyChunk()

```

void BeeGame.Terrain.LandGeneration.World.DestroyChunk (
    int x,
    int y,
    int z )

```

Destroys a Chunk st the given x, y, z postion

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     }
```

6.58.2.3 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     }
```


6.58.2.4 GetChunk()

```

Chunk BeeGame.Terrain.LandGeneration.World.GetChunk (
    int x,
    int y,
    int z )

```

Gets a chunk at eh given x, y, z

Parameters

<i>x</i>	X pos of the chunk
<i>y</i>	Y pos of the chunk
<i>z</i>	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     }

```

6.58.2.5 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

<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 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.
x, y - chunk.chunkWorldPos.y, z - chunk.chunkWorldPos.
z].breakable)
00125             {
00126
00127                 chunk.SetBlock(x - chunk.chunkWorldPos.x, y - chunk.
chunkWorldPos.y, z - chunk.chunkWorldPos.z, block);
00128                 chunk.update = true;
00129
00130                 /**updates the neighbouring chunks as when a block is broken it may be in the edge of the
chunk so their meshes also need to be updated
00131                 /**only updates chunks that need to be updated as not every chunk will need to be and
sometimes none of them will need to be
00132
00133                 /**checks if the block chaged is in the edge if the x value for the chunk
00134                 UpdateIfEqual(x - chunk.chunkWorldPos.
x, 0, new ChunkWorldPos(x - 1, y, z));
00135                 UpdateIfEqual(x - chunk.chunkWorldPos.
x, Chunk.chunkSize - 1, new ChunkWorldPos(x + 1, y, z));
00136
00137                 /**checks if the block chaged is in the edge if the y value for the chunk
00138                 UpdateIfEqual(y - chunk.chunkWorldPos.
y, 0, new ChunkWorldPos(x, y - 1, z));
00139                 UpdateIfEqual(y - chunk.chunkWorldPos.
y, Chunk.chunkSize - 1, new ChunkWorldPos(x, y + 1, z));
00140
00141                 /**checks if the block chaged is in the edge if the z value for the chunk
00142                 UpdateIfEqual(z - chunk.chunkWorldPos.
z, 0, new ChunkWorldPos(x, y, z - 1));
00143                 UpdateIfEqual(z - chunk.chunkWorldPos.
z, Chunk.chunkSize - 1, new ChunkWorldPos(x, y, z + 1));
00144
00145                 if (saveChunk)
00146                     Serialization.Serialization.SaveChunk(chunk);
00147             }
00148         }

```

6.58.2.6 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         }

```

6.58.3 Member Data Documentation

6.58.3.1 chunkHasMadeCollisionMesh

```
bool BeeGame.Terrain.LandGeneration.World.chunkHasMadeCollisionMesh = false
```

Has a Chunk made a collision mesh?

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

6.58.3.2 chunkPrefab

```
GameObject BeeGame.Terrain.LandGeneration.World.chunkPrefab
```

The chunk prefab

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

6.58.3.3 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,
MeshData meshData, bool addRoRenderMesh = true)
00034         {
00035             return meshData;
00036         }
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 System.Linq;
00003 using UnityEngine;
00004 using BeeGame.Core;
00005 using BeeGame.Items;
00006 using BeeGame.Inventory;
00007 using BeeGame.Core.Enums;
00008 using BeeGame.Terrain.Chunks;
00009 using BeeGame.Core.Dictionarys;
00010

```

```

00011 namespace BeeGame.Blocks
00012 {
00016     [Serializable]
00017     public class Apiary : Block
00018     {
00019         [NonSerialized]
00020         private GameObject myGameObject;
00021
00022         public int mutationMultiplier;
00023
00024         public new static int ID => 10;
00025
00026         #region Constructor
00027         public Apiary() : base("Apiary")
00031         {
00032             usesGameObject = true;
00033         }
00034         #endregion
00035
00036         #region Block Overrides
00037         public override GameObject GetGameObject()
00042         {
00043             return PrefabDictionary.GetPrefab("Apiary");
00044         }
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,
MeshData meshData, bool addToRenderMesh = true)
00073         {
00074             if (myGameObject == null)
00075             {
00076                 myGameObject = UnityEngine.Object.Instantiate(
PrefabDictionary.GetPrefab("Apiary"), new THVector3(x, y, z) + chunk.
chunkWorldPos, Quaternion.identity, chunk.transform);
00077                 myGameObject.GetComponent<ChestInventory>().inventoryPosition = new
THVector3(x, y, z) + chunk.chunkWorldPos;
00078                 myGameObject.GetComponent<ChestInventory>().SetChestInventory();
00079             }
00080             return base.BlockData(chunk, x, y, z, meshData, true);
00081         }
00082
00087         public override void BreakBlock(THVector3 pos)
00088         {
00089             /* removes the blocks blocks inventory save file and destroys the game object
00090             Serialization.Serialization.DeleteFile(myGameObject.GetComponent<
ApiaryInventory>().inventoryName);
00091             UnityEngine.Object.Destroy(myGameObject);
00092             /* removes the collision mesh from the chunk
00093             base.BreakBlock(pos);
00094         }
00095         #endregion
00096
00097         #region Overrides
00098         public override int GetHashCode()
00103         {
00104             return ID;
00105         }
00106
00111         public override string ToString()
00112         {
00113             return $"{itemName} \nID: {GetItemID()}";
00114         }
00115         #endregion
00116
00122         public override bool InteractWithBlock(Inventory.Inventory inv)
00123         {
00124             myGameObject.GetComponent<ApiaryInventory>().myblock = this;
00125             myGameObject.GetComponent<ApiaryInventory>().ToggleInventory(inv);
00126             return true;
00127         }
00128
00129         #region Bee Combineing Stuff
00130         public void MakeBees(Bee queen, ref Item[] inventory)
00139         {
00140             Item[] producedItems = new Item[9];
00141
00142             /* will always return a new princess and drone
00143             producedItems[0] = MakeBee(BeeType.PRINCESS, queen.queenBee);
00144             producedItems[1] = MakeBee(BeeType.DRONE, queen.queenBee);
00145
00146             var repeats = UnityEngine.Random.Range(0, queen.queenBee.
queen.pFertility);
00147

```

```

00148         /** produces as many other children as the bee staats will allow
00149         for (int i = 0; i < repeats; i++)
00150         {
00151             producedItems[i + 2] = MakeBee(queen.queenBee.queen.
pFertility > 6 ? (BeeType)UnityEngine.Random.Range(1, 3) :
BeeType.DRONE, queen.queenBee);
00152
00153             if (producedItems[i + 2] is Bee b && b.beeType !=
BeeType.PRINCESS)
00154                 producedItems[i + 2].itemStackCount =
UnityEngine.Random.Range(1, (int)queen.queenBee.queen.
pFertility + 1);
00155         }
00156
00157         /** gets the produced items
00158         var beeProduce = BeeDictionary.sGetBeeProduce(queen.
queenBee.queen.pSpecies);
00159
00160         /** chnages the stack count of the produced items to the correct number
00161         for (int i = 0; i < beeProduce.Length; i++)
00162         {
00163             beeProduce[i].itemStackCount += UnityEngine.Random.Range(1, (int)
queen.queenBee.queen.sProdSpeed + 1);
00164         }
00165
00166         /** adds the itmes that the bee species produces into the procued item array
00167         for (int i = (int)queen.queenBee.queen.pFertility + 2, prod = 0; prod <
beeProduce.Length; i++, prod++)
00168         {
00169             producedItems[i] = beeProduce[prod];
00170         }
00171
00172         /** puts the items into the inventory
00173         for (int i = 0; i < 9; i++)
00174         {
00175             if (inventory[i + 2] != null)
00176             {
00177                 /** if the slot has the same item in it and it wont be more than the max stack out but
the new item into it
00178                 if (producedItems[i] == inventory[i + 2] && inventory[i + 2].itemStackCount + 1 <=
inventory[i + 2].maxStackCount)
00179                     inventory[i + 2].itemStackCount++;
00180                 else
00181                     /** otherwise find a new slot to put the item into
00182                     for (int j = i; j < (9 - i); j++)
00183                     {
00184                         if (inventory[j + 2] == null)
00185                         {
00186                             inventory[j + 2] = producedItems[i];
00187                             break;
00188                         }
00189                         else if (producedItems[i] == inventory[j + 2] && inventory[j + 2].
itemStackCount + 1 <= inventory[j + 2].maxStackCount)
00190                         {
00191                             inventory[j + 2].itemStackCount++;
00192                             break;
00193                         }
00194                     }
00195                 }
00196                 /** if the slot is empty put the item into it
00197                 else
00198                     inventory[i + 2] = producedItems[i];
00199             }
00200         }
00201
00202         public Bee MakeBee(BeeType beeType, QueenBee queen)
00203         {
00204             /** gives all of the primary and secondary stats to the bee
00205             NormalBee nb = new NormalBee()
00206             {
00207                 {
00208                     pSpecies = CombineSpecies(queen.queen.sSpecies, queen.
drone.sSpecies),
00209                     sSpecies = CombineSpecies(queen.queen.sSpecies, queen.
drone.sSpecies),
00210                     pEffect = CombineEffect(queen.queen.sEffect, queen.
drone.sEffect),
00211                     sEffect = CombineEffect(queen.queen.sEffect, queen.
drone.sEffect),
00212                     pFertility = CombineFertility(queen.queen.sFertility, queen.
drone.sFertility),
00213                     sFertility = CombineFertility(queen.queen.sFertility, queen.
drone.sFertility),
00214                     pLifespan = CombineLifespan(queen.queen.sLifespan, queen.
drone.sLifespan),

```

```

00223         sLifespan = CombineLifespan(queen.queen.sLifespan, queen.
00224         drone.sLifespan),
00225         pProdSpeed = CombineProductionSpeed(queen.queen.sProdSpeed, queen.
00226         drone.sProdSpeed),
00227         sProdSpeed = CombineProductionSpeed(queen.queen.sProdSpeed, queen.
00228         drone.sProdSpeed)
00229     };
00230     /** returns the new bee
00231     return new Bee(beeType, nb);
00232     }
00233
00234     private BeeSpecies CombineSpecies(BeeSpecies s1,
00235     BeeSpecies s2)
00236     {
00237         BeeSpecies[] possibleSpecies = BeeDictionary.
00238         GetCombinations(s1, s2);
00239         float[] weights = possibleSpecies.Length > 2 ? BeeDictionary.
00240         GetWeights(possibleSpecies) : new float[] { 0.5f, 0.5f };
00241
00242         var randomNum = Rand(weights);
00243         var weightsSum = 0f;
00244
00245         /** when the number generated is less than the current sum of the weights return that bee
00246         for (int i = 0; i < weights.Length; i++)
00247         {
00248             if(randomNum <= weightsSum)
00249             {
00250                 return possibleSpecies[i];
00251             }
00252             weightsSum += weights[i];
00253         }
00254
00255         /** if for some reason the weights cannot work return the first bee in the combination list
00256         return possibleSpecies[0];
00257     }
00258
00259     private float Rand(float[] weights)
00260     {
00261         var totalWeights = 0f;
00262
00263         /** sums the weights
00264         for (int i = 0; i < weights.Length; i++)
00265         {
00266             totalWeights += weights[i];
00267         }
00268
00269         return (float)Math.Round(UnityEngine.Random.Range(0, totalWeights), 2);
00270     }
00271
00272     private BeeLifeSpan CombineLifespan(
00273     BeeLifeSpan b1, BeeLifeSpan b2)
00274     {
00275         return (BeeLifeSpan)ReturnChange((int)b1, (int)b2, (int)
00276     BeeLifeSpan.SEATURTLE);
00277     }
00278
00279     private uint CombineFertility(uint b1, uint b2)
00280     {
00281         return (uint)ReturnChange((int)b1, (int)b2, 5, 1);
00282     }
00283
00284     private BeeEffect CombineEffect(BeeEffect b1,
00285     BeeEffect b2)
00286     {
00287         return (BeeEffect)ReturnChange((int)b1, (int)b2, (int)
00288     BeeEffect.POSION);
00289     }
00290
00291     public BeeProductionSpeed CombineProductionSpeed(
00292     BeeProductionSpeed b1, BeeProductionSpeed b2)
00293     {
00294         return (BeeProductionSpeed)ReturnChange((int)b1, (int)b2, (int)
00295     BeeProductionSpeed.FAST);
00296     }
00297
00298     private int ReturnChange(int b1, int b2, int maxChange, int minChange = 0)
00299     {
00300         /** b1 and b2 are checked for which one is bigger than the other here as the
00301         /** queen may have a lower stat than the drone and the drone is always passed in second
00302         var change = UnityEngine.Random.Range(b1 < b2 ? b1 : b2, (b2 > b1 ? b2 : b1) + 2);
00303
00304         /** this will make it possible for the bees to mutate during combination of the stats are the
00305         same
00306
00307         /** it will also cause more random mutation more mimicing nature

```

```

00343         change += UnityEngine.Random.Range(-mutationMultiplier, mutationMultiplier);
00344
00345         /* as all but one of the stats are enums they have a min/max value so need to check that this
is not exceeded
00346         if (change > maxChange)
00347             change = maxChange;
00348         else if (minChange > change)
00349             change = minChange;
00350
00351         return change;
00352     }
00353 }
00354 #endregion
00355 }
00356 }

```

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 {
00011     [Serializable]
00012     public class Bedrock : Block
00013     {
00014         #region Data
00015         public new static int ID => -1;
00016         #endregion
00017
00018         #region Constructor
00019         public Bedrock() : base("Bedrock")
00023         {
00024             breakable = false;
00025         }
00026         #endregion
00027
00028         #region Break Block
00029         public override void BreakBlock(THVector3 pos)
00034         {
00035             return;
00036         }
00037         #endregion
00038
00039         #region Mesh
00040         public override Tile TexturePosition(Direction direction)
00046         {
00047             return new Tile() { x = 0, y = 0 };
00048         }
00049         #endregion
00050
00051         #region Overrides
00052         public override int GetHashCode()
00057         {
00058             return ID;
00059         }
00060
00065         public override string ToString()
00066         {
00067             return $"{itemName} \nID: {GetItemID()}";
00068         }
00069         #endregion
00070     }
00071 }

```


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.Dictionarys;
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("
00042 Prefabs/ItemGameObject") as GameObject, pos, Quaternion.identity) as GameObject;
00043             go.GetComponent<ItemGameObject>().item = this;
00044         }
00045
00046         public virtual void UpdateBlock(int x, int y, int z, Chunk chunk) { }
00047
00048         public virtual bool InteractWithBlock(BeeGame.
00049 Inventory.Inventory inv)
00050         {
00051             return false;
00052         }
00053         #endregion
00054
00055         #region Mesh
00056         public virtual MeshData BlockData(Chunk chunk, int x, int y, int z,
00057 MeshData meshData, bool addToRenderMesh = true)
00058         {
00059             /* Adds the Top face of the block
00060             if (!chunk.GetBlock(x, y + 1, z, false).IsSolid(Direction.DOWN))
00061             {

```

```

00107         meshData = FaceDataUp(x, y, z, meshData, addToRenderMesh);
00108     }
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 }
00142
00143 public virtual bool IsSolid(Direction direction)
00144 {
00145     return true;
00146 }
00147 #endregion
00148
00149 #region Overrides
00150 public override int GetHashCode()
00151 {
00152     return ID;
00153 }
00154
00155 public override string ToString()
00156 {
00157     return $"{itemName} \nID: {GetHashCode()}";
00158 }
00159 #endregion
00160 }
00161 }

```

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.Dictionarys;
00009
00010 namespace BeeGame.Blocks
00011 {
00012     [Serializable]
00013     public class Chest : Block
00014     {
00015         #region Data
00016         [NonSerialized]
00017         private GameObject myGameObject;
00018
00019         public new static int ID => 8;
00020         #endregion
00021
00022         #region Constructors
00023         public Chest() : base("Chest")
00024         {
00025             usesGameObject = true;
00026         }
00027         #endregion
00028
00029         #region Block Overrides
00030         public override GameObject GetGameObject()
00031         {
00032             return PrefabDictionary.GetPrefab("Chest");
00033         }
00034
00035         public override Tile TexturePosition(Direction direction)
00036         {
00037             return new Tile() { x = 0, y = 9 };
00038         }
00039
00040         public override MeshData BlockData(Chunk chunk, int x, int y, int z,
00041 MeshData meshData, bool addToRenderMesh = true)
00042         {
00043             if (myGameObject == null)
00044             {
00045                 myGameObject = UnityEngine.Object.Instantiate(
00046 PrefabDictionary.GetPrefab("Chest"), new THVector3(x, y, z) + chunk.
00047 chunkWorldPos, Quaternion.identity, chunk.transform);
00048                 myGameObject.GetComponent<ChestInventory>().inventoryPosition = new
00049 THVector3(x, y, z) + chunk.chunkWorldPos;
00050                 myGameObject.GetComponent<ChestInventory>().SetChestInventory();
00051             }
00052             return base.BlockData(chunk, x, y, z, meshData, true);
00053         }
00054
00055         public override void BreakBlock(THVector3 pos)
00056         {
00057             /* removes the blocks blocks inventory save file and destroys the game object
00058             Serialization.Serialization.DeleteFile(myGameObject.GetComponent<
00059 ChestInventory>().inventoryName);
00060             UnityEngine.Object.Destroy(myGameObject);
00061             /* removes the collision mesh from the chunk
00062             base.BreakBlock(pos);
00063         }
00064         #endregion
00065
00066         #region Inventory Suff
00067         public override bool InteractWithBlock(BeeGame.Inventory.
00068 Inventory inv)
00069         {
00070             myGameObject.GetComponent<ChestInventory>().ToggleInventory(inv);
00071             return true;
00072         }
00073         #endregion
00074
00075         #region Overrides
00076         public override int GetHashCode()
00077         {
00078             return ID;
00079         }
00080
00081         public override string ToString()
00082         {
00083             return $"{itemName}\nID{GetItemID()}";
00084         }
00085     }

```

```

00130         #endregion
00131     }
00132 }

```

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 ReturnShapedRecipeItem(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()}:";
00042             }
00043
00044             return ReturnShapedRecipeItem(recipe);
00045         }
00046
00047         public virtual Item ReturnShapelessRecipeItem(
00048             Item[] items)
00049         {
00050             return CraftingRecipes.GetShaplessRecipeResult(items)
00051 ;
00052         }
00053
00054         public virtual Item ReturnShapedRecipeItem(string recipe)
00055

```

```

00078     {
00079         return BeeGame.Core.Dictionarys.
CraftingRecipies.GetShapedRecipeItem(recipe);
00080     }
00081     #endregion
00082
00083     #region Block Overrides
00084     public override bool InteractWithBlock(Inventory.Inventory inv)
00090     {
00091         myGameObject.GetComponent<Inventory.BlockInventory.CraftingTableInventory>().myblock = this;
00092         myGameObject.GetComponent<Inventory.BlockInventory.CraftingTableInventory>().ToggleInventory(
inv);
00093         return true;
00094     }
00095
00100     public override GameObject GetGameObject()
00101     {
00102         return PrefabDictionary.GetPrefab("CraftingTable");
00103     }
00104
00118     public override MeshData BlockData(Chunk chunk, int x, int y, int z,
MeshData meshData, bool addToRenderMesh = true)
00119     {
00120         if (myGameObject == null)
00121         {
00122             myGameObject = UnityEngine.Object.Instantiate(
PrefabDictionary.GetPrefab("CraftingTable"), new
THVector3(x, y, z) + chunk.chunkWorldPos, Quaternion.identity, chunk.transform);
00123         }
00124         return base.BlockData(chunk, x, y, z, meshData, true);
00125     }
00126
00131     public override void BreakBlock(THVector3 pos)
00132     {
00133         /* removes the game object
00134         UnityEngine.Object.Destroy(myGameObject);
00135         /* removes the collision mesh from the chunk
00136         base.BreakBlock(pos);
00137     }
00138
00143     public override Sprite GetItemSprite()
00144     {
00145         return SpriteDictionary.GetSprite("TestSprite");
00146     }
00147
00156     public override Tile TexturePosition(Direction direction)
00157     {
00158         return new Tile() { x = 0, y = 9 };
00159     }
00160     #endregion
00161
00162     #region Overrides
00163     public override int GetHashCode()
00168     {
00169         return ID;
00170     }
00171     #endregion
00172 }
00173 }

```

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.Dictionarys;
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)
00033         {
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.Dictionarys;
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") {}
00020         #endregion
00021
00022         #region Item Stuff
00023         public override Sprite GetItemSprite()
00024         {
00025             return SpriteDictionary.GetSprite("Grass");
00026         }
00027         #endregion
00028
00029         #region Mesh
00030         public override void UpdateBlock(int x, int y, int z, Chunk chunk)
00031         {
00032             if (chunk.GetBlock(x, y + 1, z, false).IsSolid(Direction.DOWN))
00033                 chunk.blocks[x, y, z] = new Dirt() { changed = changed };
00034         }
00035
00036         public override Tile TexturePosition(Direction direction)
00037         {
00038             //All textures are on the same Y value for the texture atlas so Y can be set
00039             Tile tile = new Tile()
00040             {
00041                 y = 9
00042             };
00043
00044             switch (direction)
00045             {
00046                 //if we want the top face return the full grass texture
00047                 case Direction.UP:
00048                     tile.x = 3;
00049                     return tile;
00050                 //if we want the bottom face return the dirt texture
00051                 case Direction.DOWN:
00052                     tile.x = 2;
00053                     return tile;
00054                 //return the 1/2 grass texture if a side face is wanted
00055                 default:
00056                     tile.x = 4;
00057                     return tile;
00058             }
00059         }
00060         #endregion
00061
00062         #region Overrides
00063         public override int GetHashCode()
00064         {
00065             return ID;
00066         }
00067
00068         public override string ToString()
00069         {
00070             return $"{itemName} \nID: {GetItemID()}";
00071         }
00072         #endregion
00073     }
00074 }

```

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.Dictionarys;
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()
00042         {
00043             return ID;
00044         }
00045
00050         public override string ToString()
00051         {
00052             return $"{itemName} \nID: {GetItemID()}";
00053         }
00054         #endregion
00055     }
00056 }

```

7.19 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Wood.cs File Reference

Classes

- class [BeeGame.Blocks.Wood](#)

Namespaces

- namespace [BeeGame.Blocks](#)

7.20 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.Dictionarys;
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 }
```

7.21 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/Bee↵ Dictionarys.cs File Reference

Classes

- class [BeeGame.Core.Dictionarys.BeeDictionarys](#)

Namespaces

- namespace [BeeGame.Core.Dictionarys](#)

7.22 BeeDictionarys.cs

```
00001 using System.Collections.Generic;
00002 using System.Linq;
00003 using BeeGame.Core.Enums;
00004 using UnityEngine;
00005 using BeeGame.Core.Dictionarys;
00006
00007 namespace BeeGame.Core.Dictionarys
00008 {
00009     public static class BeeDictionarys
00010     {
00011         #region Bee Combination Weights
00012         private static Dictionary<BeeSpecies, float> beeCombinationWeights = new Dictionary<BeeSpecies,
00013 float>()
00014         {
00015             {BeeSpecies.COMMON, 0.15f },
00016             {BeeSpecies.HEROIC, 0.06f }
00017         };
00018         public static float[] GetWeights(BeeSpecies[] species)
```

```

00019         {
00020             var returnArray = new float[species.Length];
00021
00022             for (int i = 0; i < species.Length; i++)
00023             {
00024                 if (beeCombinationWeights.ContainsKey(species[i]))
00025                     returnArray[i] = beeCombinationWeights[species[i]];
00026                 else
00027                     returnArray[i] = 0.5f;
00028             }
00029
00030             return returnArray;
00031         }
00032     #endregion
00033
00034     #region Bee Combinations
00035     public static Dictionary<BeeSpecies[], BeeSpecies[]> beeCombinations = new Dictionary<BeeSpecies[],
BeeSpecies[]>(new BeeCombinationDictionaryEqualityComparer())
00036     {
00037         { new BeeSpecies[6] { BeeSpecies.FOREST,
BeeSpecies.MEADOWS, BeeSpecies.TROPICAL, BeeSpecies.WINTRY,
BeeSpecies.MODEST, BeeSpecies.MARSHY }, new BeeSpecies[1] {
BeeSpecies.COMMON } }
00038     };
00039
00040     public static BeeSpecies[] GetCombinations(
BeeSpecies s1, BeeSpecies s2)
00041     {
00042         var beeSpecies = new BeeSpecies[2] { s1, s2 };
00043         var returnBeeList = new List<BeeSpecies>();
00044
00045         var keys = beeCombinations.Keys.ToArray();
00046         var comparor = new BeeCombinationDictionaryEqualityComparer
00047         ();
00048
00049         for (int i = 0; i < keys.Length; i++)
00050         {
00051             if (comparor.Equals(keys[i], beeSpecies))
00052             {
00053                 var temp = beeCombinations[keys[i]];
00054
00055                 for (int j = 0; j < temp.Length; j++)
00056                 {
00057                     returnBeeList.Add(temp[j]);
00058                 }
00059             }
00060
00061             returnBeeList.Add(s1);
00062             returnBeeList.Add(s2);
00063
00064             return returnBeeList.ToArray();
00065         }
00066     #endregion
00067
00068     #region Bee Produce
00069     private static Dictionary<BeeSpecies, Items.Item[]> beeProduce = new Dictionary<
BeeSpecies, Items.Item[]>()
00070     {
00071         {BeeSpecies.FOREST, new Items.Item[] {new Items.HoneyComb(
HoneyCombType.HONEY) } },
00072         {BeeSpecies.COMMON, new Items.Item[] {new Items.HoneyComb(
HoneyCombType.HONEY) } }
00073     };
00074
00075     public static Items.Item[] GetBeeProduce(BeeSpecies species)
00076     {
00077         beeProduce.TryGetValue(species, out Items.Item[] produce);
00078
00079         /* of the produce cant be found then return a honey comb as it is probly a bug
00080         return produce ?? new Items.Item[1] { new Items.HoneyComb(
HoneyCombType.HONEY) };
00081     }
00082     #endregion
00083
00084     #region Bee Colours
00085     private static Dictionary<BeeSpecies, Color> beeColour = new Dictionary<BeeSpecies, Color>()
00086     {
00087         {BeeSpecies.FOREST, CombColour(0, 255, 0) },
00088         {BeeSpecies.COMMON, CombColour(255, 0, 0) }
00089     };
00090
00091     public static Color GetBeeColour(BeeSpecies species)
00092     {
00093         beeColour.TryGetValue(species, out Color colour);
00094
00095         return colour != null ? colour : new Color();

```

```

00096     }
00097     #endregion
00098
00099     #region Comb Colours
00100     private static Dictionary<HoneyCombType, Color> honeyCoubColour = new Dictionary<HoneyCombType,
Color>()
00104     {
00105         {HoneyCombType.HONEY, CombColour(255, 164, 56) },
00106         {HoneyCombType.ICEY, CombColour(78, 231, 231) }
00107     };
00108
00118     private static Color CombColour(float r, float g, float b, float a = 255f)
00119     {
00120         return new Color(r / 255f, g / 255f, b / 255f);
00121     }
00122
00128     public static Color GetCombColour(HoneyCombType type)
00129     {
00130         honeyCoubColour.TryGetValue(type, out var temp);
00131
00132         if (temp == null)
00133             return new Color(1, 0, 0);
00134
00135         return temp;
00136     }
00137     #endregion
00138 }
00139 }

```

7.23 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/↵

CraftingRecipies.cs File Reference

Classes

- class [BeeGame.Core.Dictionarys.CraftingRecipies](#)

Namespaces

- namespace [BeeGame.Core.Dictionarys](#)

7.24 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.Exceptipns;
00007
00008 namespace BeeGame.Core.Dictionarys
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,
Item result)
00033         {
00034             /** converts the given blocks of 3 haracters to a 9 character string
00035             var stringRecipie = "";
00036
00037             for (int i = 0; i < 3; i++)
00038             {
00039                 stringRecipie += recipie[i] as string;
00040             }
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         stringRecipe = stringRecipe.Replace(character, $"{itemID.ToString()}:");
00050     }
00051
00052     /** converts empty sots " " into "0:"
00053     stringRecipe = stringRecipe.Replace(" ", "0:");
00054
00055     /** if the recipe exists an exception is thrown as two recipies cannot be the same
00056     if (shapedCraftingRecipies.ContainsKey(stringRecipe))
00057         throw new CraftingRecipeAdditionException($"Shaped Recipe
already exists: {stringRecipe}");
00058
00059     /** adds the recipe to the dictionary
00060     shapedCraftingRecipies.Add(stringRecipe, result);
00061 }
00062
00063 public static Item GetShapedRecipeItem(string recipe)
00064 {
00065     shapedCraftingRecipies.TryGetValue(recipe, out var item);
00066
00067     return item;
00068 }
00069 #endregion
00070
00071 #region Shapless Crafting
00072 private static Dictionary<string, Item> shaplessRecipies = new Dictionary<string, Item>()
00073 {
00074
00075 };
00076
00077 public static void AddShaplessRecipe(object[] recipe,
Item result)
00078 {
00079     var itemList = new List<int>();
00080     var stringRecipe = "";
00081
00082     for (int i = 0; i < recipe.Length; i+=2)
00083     {
00084         for (int j = 0; j < (int)recipe[i+1]; j++)
00085         {
00086             itemList.Add(int.Parse(((Item)recipe[i]).GetItemID()));
00087         }
00088     }
00089
00090     itemList.Sort();
00091
00092     for (int i = 0; i < itemList.Count; i++)
00093     {
00094         stringRecipe += $"{itemList[i]}:";
00095     }
00096
00097     if (shaplessRecipies.ContainsKey(stringRecipe))
00098         throw new CraftingRecipeAdditionException($"Shaped Recipe
already exists: {stringRecipe}");
00099
00100     shaplessRecipies.Add(stringRecipe, result);
00101 }
00102
00103 public static string GetShaplessRecipeString(
Item[] recipe)
00104 {
00105     var IDList = new List<int>();
00106     var stringRecipe = "";
00107
00108     /** converts tthe given item list to an ID list so it can be sorted
00109     for (int i = 0; i < recipe.Length; i++)
00110     {
00111         if(recipe[i] != null)
00112             IDList.Add(recipe[i].GetHashCode());
00113     }
00114
00115     IDList.Sort();
00116
00117     /** converts the sorted ID list to a string so can be used as a dictionary key
00118     for (int i = 0; i < IDList.Count; i++)
00119     {
00120         /** : after each ID as it is possible for ID clashes without eg ID: 11 can be seen as 2 *
ID: 1
00121         stringRecipe += $"{IDList[i]}:";
00122     }
00123
00124     return stringRecipe;
00125 }
00126
00127 public static Item GetShaplessRecipeResult(int[] recipe)
00128 {

```

```

00168         var list = recipie.ToList();
00169         list.Sort();
00170
00171         var stringRecipe = "";
00172
00173         for (int i = 0; i < list.Count; i++)
00174         {
00175             stringRecipe += $"{list[i]}:";
00176         }
00177
00178         return GetShaplessRecipieResult(stringRecipe);
00179     }
00180
00181     public static Item GetShaplessRecipieResult(string recipie)
00182     {
00183         shaplessRecipies.TryGetValue(recipie, out var item);
00184
00185         return item;
00186     }
00187
00188     public static Item GetShaplessRecipieResult(
00189     Item[] recipie)
00190     {
00191         shaplessRecipies.TryGetValue(GetShaplessRecipieString(recipie), out var item);
00192
00193         return item;
00194     }
00195 }
00196 #endregion
00197 }
00198 }
00199 }

```

7.25 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/↵

EqualityComperors.cs File Reference

Classes

- class [BeeGame.Core.Dictionarys.BeeCombinationDictionaryEqualityComparer](#)

Namespaces

- namespace [BeeGame.Core.Dictionarys](#)

7.26 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.Dictionarys
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                 return true;
00026             }
00027         }
00028
00029         return false;
00030     }
00031 }

```

```

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 }

```

7.27 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/↔ PrefabDictionary.cs File Reference

Classes

- class [BeeGame.Core.Dictionarys.PrefabDictionary](#)
The prefabs available to the game

Namespaces

- namespace [BeeGame.Core.Dictionarys](#)

7.28 PrefabDictionary.cs

```

00001 using System.Collections.Generic;
00002 using UnityEngine;
00003
00004 namespace BeeGame.Core.Dictionarys
00005 {
00009     public static class PrefabDictionary
00010     {
00014         private static Dictionary<string, GameObject> prefabDictionary = new Dictionary<string, GameObject>
00015         ();
00019         public static void LoadPrefabs()
00020         {
00021             prefabDictionary = Resources.Resources.GetPrefabs();
00022         }
00023
00029         public static GameObject GetPrefab(string prefab)
00030         {
00031             return prefabDictionary[prefab];
00032         }
00033     }
00034 }

```

7.29 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/Sprite↔ Dictionary.cs File Reference

Classes

- class [BeeGame.Core.Dictionarys.SpriteDictionary](#)
All of the sprites available to the game

Namespaces

- namespace [BeeGame.Core.Dictionarys](#)

7.30 SpriteDictionary.cs

```

00001 using System.Collections.Generic;
00002 using UnityEngine;
00003
00004 namespace BeeGame.Core.Dictionarys
00005 {
00009     public static class SpriteDictionary
00010     {
00014         private static Dictionary<string, Sprite> itemSpriteDictionary = new Dictionary<string, Sprite>();
00015
00021         public static Sprite GetSprite(string spriteName)
00022         {
00023             itemSpriteDictionary.TryGetValue(spriteName, out Sprite sprite);
00024
00025             if (sprite == null)
00026                 return new Sprite();
00027
00028             return sprite;
00029         }
00030
00034         public static void LoadSprites()
00035         {
00036             itemSpriteDictionary = Resources.Resources.GetSprites();
00037         }
00038     }
00039 }

```

7.31 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](#) }
- enum [BeeGame.Core.Enums.BeeSpecies](#) { [BeeGame.Core.Enums.BeeSpecies.FOREST](#), [BeeGame.Core.Enums.BeeSpecies.MEADOWS](#), [BeeGame.Core.Enums.BeeSpecies.TROPICAL](#), [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.SECCLUDED](#), [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.POSION }

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

7.32 Enums.cs

```
00001 namespace BeeGame.Core.Enums
00002 {
00006     public enum HoneyCombType
00007     {
00008         HONEY, ICEY
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, FORMERLY,
        AGRARIAN, UNWEARY, INDUSTRIOUS, ICY, GLACIAL,
        NOBLE, IMPERIAL, MAJESTIC, MIRY, BOGGY, HERIOC,
        PHANTASMAL, SPECTRAL, HERMETIC, SECLUDED,
        SINISTER, FIENDISH, DEMONIC, FRUGAL, AUSTER,
        VINDICTIVE, EXOTIC, ENDEMIC, VENGEFUL, AVENGING,
```



```

    SETADFAST, HEROIC
00018     };
00019
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,
    LONGEST, SEATURTLE
00042     };
00043
00047     public enum BeeProductionSpeed
00048     {
00049         SLOW, NORMAL, FAST
00050     };
00051
00055     public enum BeeEffect
00056     {
00057         NONE, POSION
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 }

```

7.33 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Events.cs File Reference

Classes

- class [BeeGame.Core.Events](#)

Namespaces

- namespace [BeeGame.Core](#)

7.34 Events.cs

```

00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using System.Text;
00005 using BeeGame.Items;
00006 using BeeGame.Blocks;
00007
00008 namespace BeeGame.Core
00009 {
00010     public static class Events
00011     {
00012         public delegate void ItemCraftedEvent(Item item);
00013         public static ItemCraftedEvent shapedRecipieCrafted;
00014         public static ItemCraftedEvent shaplessRecipieCrafted;
00015         public static ItemCraftedEvent beeCraftedEvent;
00016
00017         public static void CallShapedRecipieCraftedEvent(Item item) => shapedRecipieCrafted?.Invoke(
00018             item);
00019         public static void CallShaplessRecipirCraftedEvent(Item item) => shaplessRecipieCrafted?.Invoke(
00020             item);
00021         public static void CallBeeCraftedEvent(Item item) => beeCraftedEvent?.Invoke(item);
00022     }
00023 }

```

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.Collections.Generic;
00003 using System.Linq;
00004 using System.Reflection;
00005 using System.Text;
00006 using UnityEngine;
00007 using System.Threading;
00008 using BeeGame.Items;
00009
00010 namespace BeeGame.Core
00011 {
00012     public static class Extensions
00013     {
00022         public static T CloneObject<T>(this T obj)
00023         {
00024             /** gets the tyoe of the given object
00025             Type typeSource = obj.GetType();
00026
00027             /** makes a new object of type T
00028             T objTarget = (T)Activator.CreateInstance(typeSource);
00029
00030             /** gets the properties in T
00031             PropertyInfo[] propertyInfo = typeSource.GetProperties(BindingFlags.Public | BindingFlags.
NonPublic | BindingFlags.Instance);
00032
00033             /** applies the properties in T to the new type T object
00034             foreach (var property in propertyInfo)
00035             {
00036                 if (property.CanWrite)
00037                 {
00038                     /** if the property is a value just set it
00039                     if (property.PropertyType.IsValueType || property.PropertyType.IsEnum || property.
PropertyType.Equals(typeof(string)))
00040                     {
00041                         property.SetValue(objTarget, property.GetValue(obj, null), null);
00042                     }
00043                     else
00044                     {
00045                         /** if the property is not a value type this function will need to be called
recursivly as it could also have non value type variables
00046                         object propertyValue = property.GetValue(obj, null);
00047
00048                         if (propertyValue == null)
00049                         {
00050                             property.SetValue(objTarget, null, null);
00051                         }
00052                         else
00053                         {
00054                             property.SetValue(objTarget, propertyValue.CloneObject(), null);
00055                         }
00056                     }
00057                 }
00058             }
00059
00060             /** gets all of the field in T
00061             FieldInfo[] fieldInfo = typeSource.GetFields();
00062
00063             /** applies all of the fiels of T to the new object if type T in the same manor that the
properites are applied
00064             foreach (var field in fieldInfo)
00065             {
00066                 if(field.FieldType.IsValueType || field.FieldType.IsEnum || field.FieldType.Equals(typeof(

```

```

        string)))
00067         {
00068             field.SetValue(objTarget, field.GetValue(obj));
00069         }
00070         else
00071         {
00072             object fieldValue = field.GetValue(obj);
00073
00074             if(fieldValue == null)
00075             {
00076                 field.SetValue(objTarget, null);
00077             }
00078             else
00079             {
00080                 field.SetValue(objTarget, field.CloneObject());
00081             }
00082         }
00083     }
00084
00085     return objTarget;
00086 }
00087
00096 public static Sprite ColourSprite(this Sprite sprite, Color colour, Color[]
coloursToAvoid = null, bool setTransparentToWhite = false)
00097 {
00098     Texture2D tex = new Texture2D((int)sprite.rect.width, (int)sprite.rect.height)
00099     {
00100         filterMode = FilterMode.Point,
00101         wrapMode = TextureWrapMode.Clamp
00102     };
00103
00104     /* sets the teture pixels to the pixels of teh sprite so the original sprite is not modified
00105     tex.SetPixels(sprite.texture.GetPixels());
00106
00107     for (int x = 0; x < tex.width; x++)
00108     {
00109         for (int y = 0; y < tex.height; y++)
00110         {
00111             /* if we dont have to avoid any colours set the pixel
00112             if (coloursToAvoid == null)
00113             {
00114                 tex.SetPixel(x, y, tex.GetPixel(x, y) * colour);
00115             }
00116             else
00117             {
00118                 for (int i = 0; i < coloursToAvoid.Length; i++)
00119                 {
00120                     /* if this colour should be avoided skip this iteration of the loop and move
00121                     on
00122                     if (tex.GetPixel(x, y) == coloursToAvoid[i])
00123                         goto Skip;
00124
00125                     tex.SetPixel(x, y, tex.GetPixel(x, y) * colour);
00126                 }
00127
00128                 /* if transparent pixels should be set to white do that
00129                 if (setTransparentToWhite && tex.GetPixel(x, y).a == 0)
00130                     tex.SetPixel(x, y, Color.white);
00131
00132                 Skip:
00133                     continue;
00134             }
00135         }
00136
00137         /* apply the new texture with its colours
00138         tex.Apply();
00139
00140         /* return the Texture2D as a sprite
00141         return Sprite.Create(tex, new Rect(0, 0, tex.width, tex.height), new
THVector2(0.5f, 0.5f));
00142     }
00143
00144     public static void SpawnItem(this Item item, THVector3 position, Quaternion
rotation = new Quaternion())
00145     {
00146         GameObject go = MonoBehaviour.Instantiate(UnityEngine.Resources.Load("
Prefabs/ItemGameObject") as GameObject, position, rotation) as GameObject;
00147         go.GetComponent<ItemGameObject>().item = item;
00148     }
00149 }
00150 }

```

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.Exceptipns;
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[] array:
00060                     /*for each possible key, check if it was pressed and if it was return that it was, if
none of them was poressed return false
00061                     foreach (var item in array)
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($"Key input name not defined: {button}");
00085     }
00086
00087     switch (inputButtons[button])
00088     {
00089         case KeyCode[] array:
00090             /**for each possible key, check if it was pressed and if it was return that it was, if
none of them was poressed return false
00091             foreach (var item in array)
00092             {
00093                 if (Input.GetKey(item))
00094                 {
00095                     return true;
00096                 }
00097             }
00098             return false;
00099         default:
00100             return Input.GetKey((KeyCode) inputButtons[button]);
00101     }
00102 }
00103
00104 public static bool GetButtonUp(string button)
00105 {
00106     if (!inputButtons.ContainsKey(button))
00107     {
00108         throw new InputException($"Key input name not defined: {button}");
00109     }
00110
00111     switch (inputButtons[button])
00112     {
00113         case KeyCode[] array:
00114             /**for each possible key, check if it was pressed and if it was return that it was, if
none of them was poressed return false
00115             foreach (var item in array)
00116             {
00117                 if (Input.GetKeyUp(item))
00118                 {
00119                     return true;
00120                 }
00121             }
00122             return false;
00123         default:
00124             return Input.GetKeyUp((KeyCode) inputButtons[button]);
00125     }
00126 }
00127
00128 public static int GetAxis(string axis)
00129 {
00130     int returnAxis = 0;
00131
00132     if (axis == "Horizontal")
00133     {
00134         if (GetButton("Right"))
00135         {
00136             returnAxis += 1;
00137         }
00138
00139         if (GetButton("Left"))
00140         {
00141             returnAxis -= 1;
00142         }
00143     }
00144     else if (axis == "Vertical")
00145     {
00146         if (GetButton("Forward"))
00147         {
00148             returnAxis += 1;
00149         }
00150
00151         if (GetButton("Backward"))
00152         {
00153             returnAxis -= 1;
00154         }
00155     }
00156
00157     return returnAxis;
00158 }
00159 }
00160 }
00161 }
00162 }

```

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 System.Collections.Generic;
00003 using System.Linq;
00004 using System.Text;
00005
00006 namespace BeeGame.Core.UnityTypeReplacements
00007 {
00008     public struct THQuaternion
00009     {
00010         public float x;
00011         public float y;
00012         public float z;
00013         public float w;
00014     }
00015 }
```

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 {
00006     [Serializable]
00007     public struct THVector2
00008     {
00009         #region Data
00010         public float x;
00011         public float y;
00012         #endregion
00013
00014         #region Constructor
00015         public THVector2(float x, float y)
00016         {
00017             this.x = x;
00018         }
00019     }
00020 }
```

```

00032         this.y = y;
00033     }
00034
00039     public THVector2(THVector2 vec2)
00040     {
00041         this = vec2;
00042     }
00043
00048     public THVector2(Vector2 vec2)
00049     {
00050         this = vec2;
00051     }
00052     #endregion
00053
00054     #region Overrides
00055     public override bool Equals(object obj)
00056     {
00057         if (!(obj is THVector2))
00058             return false;
00059         if (obj.GetHashCode() == GetHashCode())
00060             return true;
00061         return false;
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,
THVector2 b)
00092     {
00093         a.x += b.x;
00094         a.y += b.y;
00095
00096         return a;
00097     }
00098     public static THVector2 operator +(THVector2 a, float b)
00099     {
00100         a.x += b;
00101         a.y += b;
00102
00103         return a;
00104     }
00105     public static THVector2 operator +(float a, THVector2 b)
00106     {
00107         return new THVector2(a + b.x, a + b.y);
00108     }
00109     public static THVector2 operator -(THVector2 a,
THVector2 b)
00110     {
00111         a.x -= b.x;
00112         a.y -= b.y;
00113
00114         return a;
00115     }
00116     public static THVector2 operator -(THVector2 a, float b)
00117     {
00118         a.x -= b;
00119         a.y -= b;
00120
00121         return a;
00122     }
00123     public static THVector2 operator -(float a, THVector2 b)
00124     {

```

```

00125         return new THVector2(a - b.x, a - b.y);
00126     }
00127     public static THVector2 operator *(THVector2 a,
THVector2 b)
00128     {
00129         a.x *= b.x;
00130         a.y *= b.y;
00131
00132         return a;
00133     }
00134     public static THVector2 operator *(THVector2 a, float b)
00135     {
00136         a.x *= b;
00137         a.y *= b;
00138
00139         return a;
00140     }
00141     public static THVector2 operator *(float a, THVector2 b)
00142     {
00143         return new THVector2(a * b.x, a * b.y);
00144     }
00145     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 {
00006     [Serializable]
00010     public struct THVector3
00011     {
00012         #region Data
00013         public float x;
00020         public float y;
00024         public float z;
00025         #endregion
00026
00027         #region Constructors
00028         public THVector3(float x, float y, float z)
00035         {
00036             this.x = x;
00037             this.y = y;
00038             this.z = z;
00039         }
00040
00045         public THVector3(THVector3 vec3)
00046         {
00047             this = vec3;
00048         }
00049
00054         public THVector3(Vector3 vec3)
00055         {
00056             this = vec3;
00057         }
00058
00063         public THVector3(Terrain.ChunkWorldPos vec3)
00064         {
00065             this = vec3;
00066         }
00067         #endregion
00068
00069         #region Methods
00070         public static float Distance(THVector3 a, THVector3 b)
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.z), 2));
00079         }
00080         #endregion
00081
00082         #region Overrides
00083         public override bool Equals(object obj)
00089         {
00090             if (!(obj is THVector3))
00091                 return false;
00092             if (obj.GetHashCode() == GetHashCode())
00093                 return true;
00094             return false;
00095         }
00096
00101         public override int GetHashCode()
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                 return hash;
00112             }
00113         }
00114
00119         public override string ToString()
00120         {
00121             return $"{x}, {y}, {z}";
00122         }
00123
00130         public static bool operator ==(THVector3 a, THVector3 b)
00131         {
00132             return a.Equals(b);
00133         }
00140         public static bool operator !=(THVector3 a, THVector3 b)
00141         {
00142             return !(a == b);
00143         }
00144

```

```

00151     public static THVector3 operator +(THVector3 a,
THVector3 b)
00152     {
00153         a.x += b.x;
00154         a.y += b.y;
00155         a.z += b.z;
00156
00157         return a;
00158     }
00165     public static THVector3 operator +(THVector3 a, float b)
00166     {
00167         a.x += b;
00168         a.y += b;
00169         a.z += b;
00170
00171         return a;
00172     }
00179     public static THVector3 operator +(float a, THVector3 b)
00180     {
00181         return new THVector3(a + b.x, a + b.y, a + b.z);
00182     }
00189     public static THVector3 operator -(THVector3 a,
THVector3 b)
00190     {
00191         a.x -= b.x;
00192         a.y -= b.y;
00193         a.z -= b.z;
00194
00195         return a;
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,
THVector3 b)
00228     {
00229         a.x *= b.x;
00230         a.y *= b.y;
00231         a.z *= b.z;
00232
00233         return a;
00234     }
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,
THVector3 b)
00266     {
00267         a.x /= b.x;
00268         a.y /= b.y;
00269         a.z /= b.z;
00270
00271         return a;
00272     }
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)
00305     {
00306         return new Vector3(vec3.x, vec3.y, vec3.z);
00307     }
00308
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)
00328     {
00329         return new Quaternion(vec3.x, vec3.y, vec3.z, 0);
00330     }
00331     #endregion
00332 }
00333 }
```

7.45 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Exceptipns/Crafting↵

RecipieAdditionException.cs File Reference

Classes

- class [BeeGame.Exceptipns.CraftingRecipieAdditionException](#)

Namespaces

- namespace [BeeGame.Exceptipns](#)

7.46 CraftingRecipieAdditionException.cs

```

00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using System.Text;
00005
00006 namespace BeeGame.Exceptipns
00007 {
00008     public class CraftingRecipieAdditionException : Exception
00009     {
00010         public CraftingRecipieAdditionException() : base()
00011         {
00012         }
00013     }
00014
00015     public CraftingRecipieAdditionException(string message) : base(
00016 message)
00017     {
00018     }
00019
00020     public CraftingRecipieAdditionException(string message, Exception
00021 innerException) : base(message, innerException)
00022     {
00023     }
00024 }
00025 }
```

7.47 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Exceptipns/Input↵

Exception.cs File Reference

Classes

- class [BeeGame.Exceptipns.InputException](#)

Namespaces

- namespace [BeeGame.Exceptipns](#)

7.48 InputException.cs

```

00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using System.Text;
00005
00006 namespace BeeGame.Exceptipns
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     public InputException(string message, Exception innerException) : base(message,
00020 innerException)
00021     {
00022     }
00023 }
00024 }
00025 }

```

7.49 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/BlockInventory/↵ ApiaryInventory.cs File Reference

Classes

- class [BeeGame.Inventory.ApiaryInventory](#)
Inventory for Apiarys Apiary

Namespaces

- namespace [BeeGame.Inventory](#)

7.50 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
        bees should be combineing
00042         if(items.itemsInInventory.Length > 0 && !beesCombineing)
00043             CheckforBees();
00044
00045         /** if the currently combineing bees has finished combineing
00046         if (combinationTime < 0 && beesCombineing)
00047         {
00048             /** make the items that the bees should make and destroy the spent queen
00049             ((Apiary)myblock).MakeBees(items.itemsInInventory[0] as Items.Bee, ref items.
        itemsInInventory);
00050             beesCombineing = false;
00051             items.itemsInInventory[0] = null;
00052
00053             /** save the channages to the inventory
00054             SaveInv();
00055         }
00056     }
00057
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()
00071     {
00072         Items.Item posOneItem = items.itemsInInventory[0];
00073         Items.Item posTwoItem = items.itemsInInventory[1];
00074
00075         /** the item is checkd if it is a bee and if it is then a new variable is made for convenience
00076         /** if it is a queen then just set the combination time and go
00077         if (posOneItem is Items.Bee b && b.beeType == Core.Enums.BeeType.QUEEN)
00078         {
00079             combinationTime = ((float)b.queenBee.queen.pLifespan + 1) * 2;
00080             beesCombineing = true;
00081             SaveInv();
00082
00083             timerSlideer.maxValue = combinationTime;
00084
00085             return;
00086         }
00087
00088         /** of one bee is a princess and another is a drone in the correct slots combine them
00089         if(posOneItem is Items.Bee b1 && posTwoItem is Items.Bee b2 && b1.beeType == Core.Enums.BeeType
        .PRINCESS && b2.beeType == Core.Enums.BeeType.DRONE)
00090         {
00091             /** comvert the princess to a queen with the paired drone
00092             Items.Bee.ConvertToQueen(ref b1, b2.normalBee);
00093
00094             /** reduce number of drones in slot by 1 and check it is a valid stack number
00095             items.itemsInInventory[1].itemStackCount -= 1;
00096             slots[0].item = b1;
00097
00098             if (items.itemsInInventory[1].itemStackCount <= 0)
00099                 items.itemsInInventory[1] = null;
00100
00101             /** set the combination time
00102             combinationTime = ((float)b1.queenBee.queen.pLifespan + 1) * 2;
00103             beesCombineing = true;
00104
00105             SaveInv();
00106
00107             /** set the slider max to the combination time
00108             timerSlideer.maxValue = combinationTime;
00109         }
00110     }
00111     #endregion
00112
00113     #region Overrides
00114     public override void SetChestInventory(string invName = "Apiary")
00115     {
00116         base.SetChestInventory("Apiary" );
00117     }
00118     #endregion
00119 }
00120
00127 }

```

7.51 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/BlockInventory/↩ ChestInventory.cs File Reference

Classes

- class [BeeGame.Inventory.ChestInventory](#)
Incentory for the chests

Namespaces

- namespace [BeeGame.Inventory](#)

7.52 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 postion 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.Serialization.DeSerializeInventory(this, inventoryName);
00071         }
00072
00073         #region Player Inventory
00074         void SetPlayerItems()
00078         {
00079             for (int i = 0; i < playerinventory.items.itemsInInventory.Length; i++)
00080             {
00081                 items.itemsInInventory[i + (inventorySize - 36)] = playerinventory.
00082                 items.itemsInInventory[i];
00083             }
00084
00088         void ApplyPlayerItems()
00089         {
00090             for (int i = 0; i < playerinventory.items.itemsInInventory.Length; i++)

```

```

00091         {
00092             playerinventory.items.itemsInInventory[i] = items.itemsInInventory[i +
(inventorySize - 36)];
00093         }
00094
00095         playerinventory.SaveInv();
00096     }
00097     #endregion
00098
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 immediately 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 }

```

7.53 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/BlockInventory/↵

CraftingTableInventory.cs File Reference

Classes

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

Namespaces

- namespace [BeeGame.Inventory.BlockInventory](#)

7.54 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
00009 namespace BeeGame.Inventory.BlockInventory
00010 {
00014     public class CraftingTableInventory : ChestInventory
00015     {
00016         #region Data

```

```

00017         public delegate void ItemRemovedFromResult();
00024         public ItemRemovedFromResult result;
00025     #endregion
00026
00027     #region Unity Methods
00028     protected void Start()
00032     {
00033         SetChestInventory();
00034         result = CraftedItemRemoved;
00035     }
00036
00037
00041     protected void Update()
00042     {
00043         UpdateChestInventory();
00044
00045         if (inventory.activeInHierarchy)
00046         {
00047             CheckShapedRecipie();
00048
00049             /* checks for shapless recipies second
00050             if(items.itemsInInventory[9] == null)
00051                 CheckShapelessRecipie();
00052         }
00053     }
00054
00058     protected void OnDestroy()
00059     {
00060         /* just ensures no memory leaks occur
00061         result -= CraftedItemRemoved;
00062     }
00063     #endregion
00064
00065     #region Crafting Stuff
00066     public virtual void CheckShapedRecipie()
00070     {
00071         var items = new Item[9];
00072
00073         for (int i = 0; i < items.Length; i++)
00074         {
00075             items[i] = base.items.itemsInInventory[i];
00076         }
00077
00078         /* if it is a recipie put the result into the crafting result slot
00079         Item item = ((CraftingTable)myblock).ReturnShapedRecipieItem(items);
00080         if (item != base.items.itemsInInventory[9])
00081             base.items.itemsInInventory[9] = item;
00082     }
00083
00087     public virtual void CheckShapelessRecipie()
00088     {
00089         var items = new Item[9];
00090
00091         for (int i = 0; i < items.Length; i++)
00092         {
00093             items[i] = base.items.itemsInInventory[i];
00094         }
00095
00096         Item item = ((CraftingTable)myblock).ReturnShapelessRecipieItem(items);
00097         if (item != base.items.itemsInInventory[9])
00098             base.items.itemsInInventory[9] = item;
00099     }
00100
00104     public void CraftedItemRemoved()
00105     {
00106         if (items.itemsInInventory[9] != null)
00107         {
00108             Events.CallShapedRecipieCraftedEvent(items.
itemsInInventory[9]);
00109             for (int i = 0; i < 9; i++)
00110             {
00111                 if (items.itemsInInventory[i] != null)
00112                     items.itemsInInventory[i].itemStackCount -= 1;
00113             }
00114         }
00115     }
00116     #endregion
00117
00118     #region Inventory Stuff
00119     public virtual void DropItemsFromInventory()
00126     {
00127         /* looks at every item in the crafting grid
00128         for (int i = 0; i < 9; i++)
00129         {
00130             if (items.itemsInInventory[i] != null)
00131             {
00132                 /* spwns it and removes it from the inventory if an items exists within

```



```

00133         for (int j = 0; j < items.itemsInInventory[i].itemStackCount; j++)
00134         {
00135             items.itemsInInventory[i].SpawnItem((THVector3)this.transform.position +
new THVector3(0, 1, 0));
00136         }
00137         items.itemsInInventory[i] = null;
00138     }
00139 }
00140 }
00141 #endregion
00142
00143 #region Overrides
00144 public override void ToggleInventory(Inventory inv)
00145 {
00150     base.ToggleInventory(inv);
00151
00152     /** if the inventory was closed drop the items within
00153     if (!inventory.activeInHierarchy)
00154         DropItemsFromInventory();
00155 }
00156
00164 public override void SetChestInventory(string invName = "Workbench")
00165 {
00166     SetInventorySize(inventorySize);
00167     /** sets the UI to not be seen as inventories cannot start open
00168     inventory.SetActive(false);
00169 }
00170
00179 public override void AddItemToSlots(int slotIndex, Item item)
00180 {
00181     items.AddItem(slotIndex, item);
00182 }
00183
00187 public override void SaveInv()
00188 {
00189     /** does not need to be saved so overridden to do nothing
00190 }
00191 #endregion
00192 }
00193 }

```

7.55 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Inventory.cs

File Reference

Classes

- class [BeeGame.Inventory.Inventory](#)
Base class for all inventories in the game

Namespaces

- namespace [BeeGame.Inventory](#)

7.56 Inventory.cs

```

00001 using System;
00002 using UnityEngine;
00003 using BeeGame.Items;
00004 using BeeGame.Core.Dictionarys;
00005
00006 namespace BeeGame.Inventory
00007 {
00011     public class Inventory : MonoBehaviour
00012     {
00013         #region Data
00014         public ItemsInInventory items;
00021         public InventorySlot[] slots;
00025         internal Item floatingItem;
00029         public string inventoryName = "";
00033         protected bool thisInventoryOpen = false;
00037         private GameObject spriteAtCursor;
00044         public Blocks.Block myblock;

```

```

00045         #endregion
00046
00047         #region Init
00048         public bool InventorySet()
00053         {
00054             if (items == null)
00055                 return true;
00056
00057             return false;
00058         }
00059
00064         public void SetInventorySize(int inventorySize)
00065         {
00066             items = new ItemsInInventory(slots.Length);
00067         }
00068
00076         public void SetAllItems(ItemsInInventory items)
00077         {
00078             this.items = items;
00079         }
00080         #endregion
00081
00082         #region Update
00083         public void UpdateBase()
00087         {
00088             PutItemsInSlots();
00089             DrawItemAtCursor();
00090         }
00091
00095         private void DrawItemAtCursor()
00096         {
00097             if(floatingItem != null)
00098             {
00099                 if (spriteAtCursor == null)
00100                 {
00101                     spriteAtCursor = Instantiate(PrefabDictionary.
00102 GetPrefab("ItemIcon"));
00103                     spriteAtCursor.GetComponentInChildren<UnityEngine.UI.Image>().sprite =
00104 floatingItem.GetItemSprite();
00105                     /* will update a the sprite of in item is swapped between a slot and teh floating item if
00106 the previous item wasnt put into a slot first
00107                     else if(spriteAtCursor != null)
00108                     {
00109                         spriteAtCursor.GetComponentInChildren<UnityEngine.UI.Image>().sprite =
00110 floatingItem.GetItemSprite();
00111                     }
00112                     spriteAtCursor.transform.GetChild(0).position = Input.mousePosition;
00113                     }
00114                     else
00115                     {
00116                         Destroy(spriteAtCursor);
00117                     }
00118                 }
00119             }
00120             #endregion
00121
00122             #region Edit Inventory
00123             public virtual void ToggleInventory(Inventory inv)
00124             {
00131                 throw new NotImplementedException();
00132             }
00133
00134             public virtual void SaveInv()
00135             {
00136                 Serialization.Serialization.SerializeInventory(this, inventoryName);
00137             }
00138
00139             void PutItemsInSlots()
00140             {
00141                 /* goes through all of the items in the array setting then all to a slot
00142                 for (int i = 0; i < slots.Length; i++)
00143                 {
00144                     slots[i].slotIndex = i;
00145                     slots[i].myInventory = this;
00146                     slots[i].item = items.itemsInInventory[i];
00147                 }
00148             }
00149
00154             public ItemsInInventory GetAllItems()
00155             {
00156                 return items;
00157             }
00158
00164             public virtual void AddItemToSlots(int slotIndex, Item item)
00165             {
00166                 items.AddItem(slotIndex, item);

```

7.57

C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/InventorySlot.cs File Reference 317

```
00167         /** saves the inventory changes
00168         Serialization.Serialization.SerializeInventory(this, inventoryName);
00169     }
00170
00176     public bool AddItemToInventory(Item item)
00177     {
00178         return items.AddItem(item);
00179     }
00180     #endregion
00181 }
00182 }
```

7.57 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/InventorySlot.cs File Reference ↩

Classes

- class [BeeGame.Inventory.InventorySlot](#)

Namespaces

- namespace [BeeGame.Inventory](#)

7.58 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.Dictionarys;
00007
00008 namespace BeeGame.Inventory
00009 {
00010     public class InventorySlot : MonoBehaviour, IPointerClickHandler, IPointerEnterHandler,
00011         IPointerExitHandler
00012     {
00013         #region Data
00014         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         private void Update()
00043         {
00044             CheckItem();
00045             UpdateIcon();
00046         }
00047
00048
00052         void UpdateIcon()
00053         {
00054             if(item == null)
00055             {
00056                 GetComponent<Image>().sprite = null;
00057             }
00058             else
00059             {
00060                 if(!item.Equals(new Item()))
00061                     GetComponent<Image>().sprite = item.GetItemSprite();
00062             }
00063
00064             /** if the slot is selected in the hotbar give the player some indication by colouring it grey
00065             if (selectedSlot)
00066             {
00067                 GetComponent<Image>().color = Color.gray;
00068             }
00069             else
00070             {
```

```

00071         GetComponent<Image>().color = Color.white;
00072     }
00073 }
00074
00075 #region Interact With Slot
00076 public void OnPointerClick(PointerEventData eventData)
00084 {
00085     if (myInventory.floatingItem != null)
00086     {
00087         /* Left click moves whole stacks of items
00088         if (eventData.button == PointerEventData.InputButton.Left)
00089         {
00090             /* If the item in the slot is empty put the floating item into it then clear it and
the slot can have items inserted
00091             if (item == null && itemsCanBeInserted)
00092             {
00093                 item = myInventory.floatingItem;
00094                 myInventory.floatingItem = null;
00095                 myInventory.AddItemToSlots(slotIndex, item);
00096                 return;
00097             }
00098             /* if the items are the same
00099             if(myInventory.floatingItem == item && itemsCanBeInserted)
00100             {
00101                 /* if the item in the inventoys stack count + the floating items stack count is
less than the max stack count
00102                 if (myInventory.floatingItem.itemStackCount + item.
itemStackCount <= item.maxStackCount)
00103                 {
00104                     AddToSlot(myInventory.floatingItem.
itemStackCount);
00105                     return;
00106                 }
00107                 /* if the item stack added is larger than the max count add as many as you can and
move on
00108                 else
00109                 {
00110                     AddToSlot(item.maxStackCount - item.
itemStackCount);
00111                     return;
00112                 }
00113             }
00114             /* if the tiems are the same but items cannot be inserted into the slot add as many
items as you
00115             /* can from the slot to the floating item
00116             else if(myInventory.floatingItem == item && !itemsCanBeInserted)
00117             {
00118                 AddToFloatingItem();
00119                 {
00120                     if (myInventory is BlockInventory.CraftingTableInventory c)
00121                         c.result.Invoke();
00122                 }
00123                 return;
00124             }
00125             /* If the items were not == swap them
00126             else
00127             {
00128                 /* only if items can be inserted into the slot
00129                 if(itemsCanBeInserted)
00130                     SwapItems();
00131                 return;
00132             }
00133         }
00134     else if(eventData.button == PointerEventData.InputButton.Right)
00135     {
00136         /* if the item in slot is null add 1 from the floating item to it
00137         if(item == null && itemsCanBeInserted)
00138         {
00139             AddToSlot(1);
00140             return;
00141         }
00142         /* if the items are the same add 1 from the floating item to this item
00143         else if (item == myInventory.floatingItem && itemsCanBeInserted)
00144         {
00145             AddToSlot(1);
00146             return;
00147         }
00148     }
00149 }
00150 /* if the floating item is null
00151 else
00152 {
00153     /* add 1/2 of the stack into the floating item if right click was pressed
00154     if(eventData.button == PointerEventData.InputButton.Right)
00155     {
00156         SplitStack();
00157     }

```

```

00158             /* blocks removed some weird name conflation
00159             {
00160                 if (myInventory is BlockInventory.CraftingTableInventory c)
00161                     c.result.Invoke();
00162             }
00163
00164             return;
00165         }
00166
00167         /* otherwise add the items into the floating item slot
00168         SwapItems();
00169         /* ^ does not need to check that the slot cannot be inserted into as null be being
inserted because the floating item is null
00170
00171         {
00172             if (myInventory is BlockInventory.CraftingTableInventory c)
00173                 c.result.Invoke();
00174         }
00175
00176         return;
00177     }
00178 }
00179
00180
00184 void AddToFloatingItem()
00185 {
00186     /* if the whole stack can be added do it and move on
00187     if (myInventory.floatingItem.itemStackCount + item.
itemStackCount <= item.maxStackCount)
00188     {
00189         myInventory.floatingItem.itemStackCount += item.
itemStackCount;
00190
00191         item = null;
00192
00193         myInventory.AddItemToSlots(slotIndex, item);
00194
00195         return;
00196     }
00197
00198     /* if the whole stack cannot be added calculate how many need to be removed from the slots
item stack
00199     item.itemStackCount -= (item.maxStackCount - myInventory.
floatingItem.itemStackCount);
00200     /* set the floating item to the max stack count
00201     myInventory.floatingItem.itemStackCount = item.
maxStackCount;
00202
00203     myInventory.AddItemToSlots(slotIndex, item);
00204 }
00205
00210 void AddToSlot(int numerToAdd)
00211 {
00212     /* if the item in the slot is null create it
00213     if (item == null)
00214     {
00215         item = myInventory.floatingItem.CloneObject();
00216         item.itemStackCount = 0;
00217     }
00218
00219     /* add to number to add to the stack count
00220     item.itemStackCount += numerToAdd;
00221
00222     /* if the stack count is now larger than it should be dont let it be
00223     if (item.itemStackCount > item.maxStackCount)
00224     {
00225         item.itemStackCount = item.maxStackCount;
00226     }
00227
00228     /* remove the numebr if items form the floating item then check the floating item is not null
00229     myInventory.floatingItem.itemStackCount -= numerToAdd;
00230     CheckFloatingItem();
00231     /* save the inventory changes
00232     myInventory.AddItemToSlots(slotIndex, item);
00233 }
00234
00241 void SplitStack()
00242 {
00243     myInventory.floatingItem = item.CloneObject();
00244     int give = (item.itemStackCount + 1) / 2;
00245     myInventory.floatingItem.itemStackCount = give;
00246     item.itemStackCount -= give;
00247
00248     if (item.itemStackCount <= 0)
00249         item = null;
00250
00251     myInventory.AddItemToSlots(slotIndex, item);

```

```

00252         Destroy(itemText);
00253     }
00254
00258     void SwapItems()
00259     {
00260         /** temp copy of the item
00261         Item temp = myInventory.floatingItem;
00262         /** sets the floating item
00263         myInventory.floatingItem = item;
00264         /** sets the item that was in the floating item to the item in the the slot
00265         item = temp;
00266         /** Saves the changes to the inventory
00267         myInventory.AddItemToSlots(slotIndex, item);
00268         /** destroys the text as it is not needed anymore
00269         Destroy(itemText);
00270     }
00271
00275     void CheckFloatingItem()
00276     {
00277         if(myInventory.floatingItem.itemStackCount <= 0)
00278         {
00279             myInventory.floatingItem = null;
00280         }
00281     }
00282     #endregion
00283
00287     private void CheckItem()
00288     {
00289         if (item != null && myInventory != null)
00290         {
00291             if (item.itemStackCount == 0 || item.itemName == "TestItem")
00292             {
00293                 myInventory.items.itemsInInventory[slotIndex] = null;
00294                 Destroy(itemText);
00295             }
00296         }
00297     }
00298
00299     #region Display Item On Hover
00300     public void OnPointerEnter(PointerEventData eventData)
00301     {
00302         /** if the item is null or the floating item has something in it dont display the item text as
00303         it is not necessary
00304         if (item != null && myInventory.floatingItem == null)
00305         {
00306             itemText = Instantiate(PrefabDictionary.
GetPrefab("ItemDetails"));
00310             /** sets the text to the correct postion
00311             itemText.transform.GetChild(0).position = Input.mousePosition;
00312             /** puts the correct text in the box
00313             itemText.transform.GetChild(0).GetChild(0).GetComponent<Text>().text = $"
{item.GetItemName()}\nStack: {item.itemStackCount}";
00314         }
00315     }
00316
00321     public void OnPointerExit(PointerEventData eventData)
00322     {
00323         Destroy(itemText);
00324     }
00325
00329     void OnDisable()
00330     {
00331         Destroy(itemText);
00332     }
00333     #endregion
00334 }
00335 }

```

7.59 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](#)

7.60 ItemsInInventory.cs

```

00001 using System;
00002 using BeeGame.Items;
00003
00004 namespace BeeGame.Inventory
00005 {
00009     [Serializable]
00010     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 }

```

7.61 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](#)

7.62 PlayerInventory.cs

```

00001 using UnityEngine;
00002 using BeeGame.Items;
00003 using BeeGame.Core;
00004
00005 namespace BeeGame.Inventory.Player_Inventory
00006 {
00010     public class PlayerInventory : Inventory
00011     {
00012         #region Data
00013         public GameObject playerInventory;
00017         #endregion
00018

```

```

00019         #region Init
00020         void Awake()
00024         {
00025             SetPlayerInventory();
00026             inventoryName = "PlayerInventory";
00027             Serialization.Serialization.DeSerializeInventory(this, inventoryName);
00028         }
00029
00033         void SetPlayerInventory()
00034         {
00035             if (!InventorySet())
00036                 SetInventorySize(36);
00037         }
00038         #endregion
00039
00043         void Update()
00044         {
00045             UpdateBase();
00046
00047             /* checks if the inventory should be opened/closed
00048             if ((thisInventoryOpen || !playerInventory.activeInHierarchy) && !
THInput.chestOpen && THInput.GetButtonDown("Player Inventory"))
00049             {
00050                 if (THInput.blockInventoryJustClosed)
00051                 {
00052                     THInput.blockInventoryJustClosed = false;
00053                     return;
00054                 }
00055                 else
00056                 {
00057                     OpenPlayerInventory();
00058                 }
00059             }
00060
00061             /* dont pickup items if the inventory is open
00062             if (THInput.isAnotherInventoryOpen)
00063                 return;
00064
00065             /* checks if somethig should be picked up and put into the inventory
00066             RaycastHit[] hit = Physics.SphereCastAll(transform.position, 1f, transform.forward);
00067
00068             for (int i = hit.Length - 1; i >= 0; i--)
00069             {
00070                 if (hit[i].collider.GetComponent<ItemGameObject>())
00071                     PickupItem(hit[i].collider.GetComponent<ItemGameObject>());
00072             }
00073         }
00074
00075         #region Hotbar
00076         public void SelectedSlot(int index)
00077         {
00082             for (int i = 0; i < slots.Length; i++)
00083             {
00084                 slots[i].selectedSlot = false;
00085             }
00086
00087             slots[index].selectedSlot = true;
00088         }
00089
00090         public bool GetItemFromHotBar(int slotIndex, out Item outItem)
00091         {
00092             /* get the item
00093             outItem = GetAllItems().itemsInInventory[slotIndex];
00094
00095             if (outItem == null)
00096                 return false;
00097
00098             /* if the item is placeable and is not null remove 1 from the inventory as it is assumed it is
00099             about to be placed in the world
00100             if(outItem.placeable)
00101                 RemoveItemFromInventory(slotIndex);
00102
00103             return outItem.placeable;
00104         }
00105         #endregion
00106
00113         #region Interact With Inventory
00114         void OpenPlayerInventory()
00115         {
00116             if (floatingItem != null)
00117                 return;
00118             thisInventoryOpen = !thisInventoryOpen;
00119             playerInventory.SetActive(!playerInventory.activeInHierarchy);
00120             THInput.isAnotherInventoryOpen = !
THInput.isAnotherInventoryOpen;
00121         }
00124

```



```
00125         /** hides/shows the mouse depending on if te inventory is open or not
00126         if (playerInventory.activeInHierarchy)
00127         {
00128             Cursor.lockState = CursorLockMode.None;
00129             Cursor.visible = true;
00130         }
00131         else
00132         {
00133             Cursor.visible = false;
00134             Cursor.lockState = CursorLockMode.Locked;
00135         }
00136     }
00137
00142     public void RemoveItemFromInventory(int index)
00143     {
00144         /** if the item is already null nothign needs to be removed
00145         if (GetAllItems().itemsInInventory[index] != null)
00146         {
00147             /** remove 1 item and if that was the last in the stack remove the item from the inventory
00148             GetAllItems().itemsInInventory[index].itemStackCount -= 1;
00149
00150             if (GetAllItems().itemsInInventory[index].itemStackCount <= 0)
00151                 GetAllItems().itemsInInventory[index] = null;
00152
00153             Serialization.Serialization.SerializeInventory(this, inventoryName);
00154         }
00155     }
00156
00161     void PickupItem(ItemGameObject item)
00162     {
00163         item.item.itemStackCount = 1;
00164
00165         /** if the item can be added to the inventory do that
00166         if (AddItemToInventory(item.item))
00167         {
00168             /** if the item was added destroyits gameobject and save the inventory
00169             Destroy(item.gameObject);
00170             Serialization.Serialization.SerializeInventory(this, inventoryName);
00171         }
00172     }
00173     #endregion
00174 }
00175 }
```

7.63 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](#)

7.64 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 }
```

7.65 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](#)

7.66 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 }

```

7.67 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](#)

7.68 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.Dictionarys;
00007
00008 namespace BeeGame.Items
00009 {
00010     [Serializable]
00011     public class Bee : Item
00012     {
00013         #region Data
00014         public bool canSeeBeeData = false;
00015
00016         public BeeType beeType { get; set; }
00017         private BeeType previousBeeType { get; set; }
00018         public override int maxStackCount { get { return maxStack; } }
00019         private int maxStack = 64;
00020
00021         [NonSerialized]
00022         private Sprite itemSprite;
00023
00024         public QueenBee queenBee { get; set; }
00025         public NormalBee normalBee { get; set; }
00026
00027         public new static int ID => 11;
00028         #endregion
00029
00030         #region Constructors
00031         public Bee()
00032         {
00033             normalBee = new NormalBee();
00034         }
00035
00036         public Bee(BeeType beeType, NormalBee normalBee) : base(new CultureInfo("en-US",
00037             false).TextInfo.ToTitleCase($"{normalBee.pSpecies} {beeType}".ToLower()))
00038         {
00039             if (beeType == BeeType.PRINCESS || beeType == BeeType.QUEEN)
00040                 maxStack = 1;
00041             this.beeType = beeType;
00042             this.normalBee = normalBee;
00043         }
00044
00045         public Bee(BeeType beeType, QueenBee queenBee) : base(new CultureInfo("en-US",
00046             false).TextInfo.ToTitleCase($"{queenBee.queen.pSpecies} {beeType}".ToLower()))
00047         {
00048             if (beeType == BeeType.PRINCESS || beeType == BeeType.QUEEN)
00049                 maxStack = 1;
00050             this.beeType = beeType;
00051             this.queenBee = queenBee;
00052         }
00053         #endregion
00054
00055         #region Item Overrides
00056         public override Sprite GetItemSprite()
00057         {
00058             /* if the bee has not change in any way dont rebuild the sprite as that takes time
00059             if(previousBeeType == beeType && itemSprite != null)
00060             {
00061                 return itemSprite;
00062             }
00063
00064             previousBeeType = beeType;
00065
00066             /* set the correct sprite and colour
00067             if (beeType == BeeType.QUEEN)
00068             {
00069                 /* avoids the crown, black body, yellow body, and both colours of the wings
00070                 Color[] colorsToAvoid = { new Color(0, 0, 0), new Color(232f, 200f, 42f, 255f) / 255f, new
00071                 Color(232f, 213f, 106f, 255f) / 255f, new Color(156f, 146f, 130f, 255f) / 255f, new Color(225f, 223f, 219f,
00072                 255f) / 255f };
00073
00074                 return itemSprite = SpriteDictionary.GetSprite("Queen").
00075                 ColourSprite(BeeDictionarys.GetBeeColour((BeeSpecies) (queenBee?.queen.pSpecies)),
00076                 coloursToAvoid: colorsToAvoid);
00077             }
00078             else if (beeType == BeeType.PRINCESS)
00079             {
00080                 /* avoids the tiara, black body, yellow body, and both colours of the wings
00081                 Color[] colorsToAvoid = { new Color(0, 0, 0), new Color(191f, 195f, 45f, 255f) / 255f, new
00082                 Color(191f, 195f, 44f, 255f) / 255f, new Color(156f, 146f, 130f, 255f) / 255f, new Color(225f, 223f, 219f, 2
00083                 55f) / 255f, new Color(232f, 200, 42, 255f) / 255f };
00084
00085                 return itemSprite = SpriteDictionary.GetSprite("Princess").

```

```

        ColourSprite(BeeDictionary.GetBeeColour((BeeSpecies) normalBee?.pSpecies)),
        coloursToAvoid: coloursToAvoid);
00118     }
00119     else
00120     {
00121         /* avoids the block body, yellow body, and both wing colours */
00122         Color[] coloursToAvoid = { new Color(0, 0, 0), new Color(156f, 146f, 130f, 255f) / 255f, new
        Color(225f, 223f, 219f, 255f) / 255f, new Color(232f, 200, 42, 255f) / 255f };
00123         return itemSprite = SpriteDictionary.GetSprite("Drone").
        ColourSprite(BeeDictionary.GetBeeColour((BeeSpecies) normalBee?.pSpecies),
        coloursToAvoid: coloursToAvoid);
00124     }
00125 }
00126
00131 public override string GetItemID()
00132 {
00133     return $"{GetHashCode()}\\{(int)beeType}{queenBee?.GetHashCode() ?? normalBee?.GetHashCode()}";
00134 }
00135 #endregion
00136
00137 #region Bee Stuff
00138 public static void ConvertToQueen(Bee princess, NormalBee drone)
00139 {
00140     ConvertToQueen(ref princess, drone);
00141 }
00142
00152 public static void ConvertToQueen(ref Bee princess,
NormalBee drone)
00153 {
00154     princess.beeType = BeeType.QUEEN;
00155     princess.queenBee = new QueenBee(princess.normalBee, drone);
00156     princess.normalBee = null;
00157
00158     princess.itemName = new CultureInfo("en-US", false).TextInfo.ToTitleCase($"
{princess.queenBee.queen.pSpecies} {princess.beeType}".ToLower());
00159 }
00160
00171 public 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)
00172 {
00173     NormalBee normBee = new NormalBee()
00174     {
00175         pSpecies = species,
00176         pLifespan = lifespan,
00177         pFertility = fertility,
00178         pProdSpeed = prodSpeed,
00179         pEffect = effect,
00180         sEffect = effect,
00181         sFertility = fertility,
00182         sLifespan = lifespan,
00183         sProdSpeed = prodSpeed,
00184         sSpecies = species
00185     };
00186
00187     switch (beeType)
00188     {
00189         case BeeType.QUEEN:
00190             return new Bee(beeType, new QueenBee(normBee, normBee));
00191         default:
00192             return new Bee(beeType, normBee);
00193     }
00194 }
00195 #endregion
00196
00197 #region Overrides
00198 public override int GetHashCode()
00199 {
00200     return ID;
00201 }
00202 #endregion
00203 }
00204
00209 [Serializable]
00210 public class QueenBee
00211 {
00212     /* Properties so that they can be copied by reflection as it does not copy variables only
properties */
00216     public NormalBee queen { get; set; }
00220     public NormalBee drone { get; set; }
00221
00222     public QueenBee() { }
00223
00224     public QueenBee(NormalBee princess, NormalBee drone)
00225     {

```

```
00226         this.queen = princess;
00227         this.drone = drone;
00228     }
00229
00230     public override int GetHashCode()
00231     {
00232         unchecked
00233         {
00234             return (int)Int64.Parse($"{queen.GetHashCode()} {drone.GetHashCode()}");
00235         }
00236     }
00237 }
00238
00239 [Serializable]
00240 public class NormalBee
00241 {
00242     #region Phenotype
00243     /** Currently shown traits of the bee
00244
00245     public BeeSpecies pSpecies;
00252     public BeeLifeSpan pLifespan;
00256     public uint pFertility;
00260     public BeeEffect pEffect;
00264     public BeeProductionSpeed pProdSpeed;
00265     #endregion
00266
00267     #region Secondary
00268     /** Traits of the bee used in the bees combination
00269
00273     public BeeSpecies sSpecies;
00277     public BeeLifeSpan sLifespan;
00281     public uint sFertility;
00285     public BeeEffect sEffect;
00289     public BeeProductionSpeed sProdSpeed;
00290     #endregion Secondary
00291
00292     public override int GetHashCode()
00293     {
00294         unchecked
00295         {
00296             //int hashCode = 13;
00297
00298             var temp = $"
00299 {(int)pSpecies}{(int)sSpecies}{(int)pLifespan}{(int)sLifespan}{(int)pFertility}{(int)sFertility}{(int)pEffect}{(int)sEf
00300             var hashCode = (int)(Int64.Parse(temp) ^ (127 * 13) / 159);
00301
00302             //hashCode += ((int)pSpecies ^ (int)pLifespan ^ (int)pFertility ^ (int)pEffect ^
00303             //hashCode += ((int)sSpecies ^ (int)sLifespan ^ (int)sFertility ^ (int)sEffect ^
00304             //hashCode += ((int)pProdSpeed) * 127;
00305             //hashCode += ((int)sProdSpeed) * 307;
00306             return hashCode;
00307         }
00308     }
00309 }
```

7.69 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](#)

7.70 HoneyComb.cs

```

00001 using System;
00002 using System.Globalization;
00003 using BeeGame.Core;
00004 using BeeGame.Core.Enums;
00005 using BeeGame.Core.Dictionarys;
00006 using UnityEngine;
00007
00008 namespace BeeGame.Items
00009 {
00010     [Serializable]
00011     public class HoneyComb : Item
00012     {
00013         #region Data
00014         public HoneyCombType type { get; set; }
00015
00016         public Color CombColour
00017         {
00018             get
00019             {
00020                 return BeeDictionarys.GetCombColour(type);
00021             }
00022         }
00023     }
00024
00025     [NonSerialized]
00026     private Sprite itemSprite;
00027
00028     public new static int ID => 12;
00029 #endregion
00030
00031 #region Constructors
00032 public HoneyComb() : base(new CultureInfo("en-US", false).TextInfo.ToTitleCase($"
00033 {HoneyCombType.HONEY} Comb".ToLower()))
00034 {
00035     usesGameObject = true;
00036     type = HoneyCombType.HONEY;
00037 }
00038
00039 public HoneyComb(HoneyCombType type) : base(new CultureInfo("en-US", false).
00040 TextInfo.ToTitleCase($"{type.ToString()} Comb".ToLower()))
00041 {
00042     usesGameObject = true;
00043     this.type = type;
00044 }
00045 #endregion
00046
00047 #region Item Overrides
00048 public override Sprite GetItemSprite()
00049 {
00050     return itemSprite ?? (itemSprite = SpriteDictionary.
00051 GetSprite("HoneyComb").ColourSprite(CombColour));
00052 }
00053
00054 public override GameObject GetGameObject()
00055 {
00056     GameObject obj = PrefabDictionary.GetPrefab("HoneyComb");
00057     /* cannot access the instance material from here have to do it on the obejct
00058 obj.GetComponent<ApplyColour>().colour = CombColour;
00059 return obj;
00060 }
00061
00062 public override string GetItemID()
00063 {
00064     return $"{GetHashCode()}\\{(int)type}";
00065 }
00066 #endregion
00067
00068 #region Overrides
00069 public override int GetHashCode()
00070 {
00071     return ID;
00072 }
00073 #endregion
00074 }
00075
00076 }

```

7.71 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`

7.72 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.Dictionarys;
00008 using System.Runtime.Serialization.Formatters.Binary;
00009
00010 namespace BeeGame.Items
00011 {
00012     [Serializable]
00013     public class Item : AbstractItem, ICloneable
00014     {
00015         #region Data
00016         internal string itemName { get; set; }
00017         public virtual bool placeable => false;
00018         public bool usesGameObject { get; set; }
00019         private const float tileSize = 0.1f;
00020
00021         public int itemStackCount { set { count = value; } get { return count; } }
00022         private int count = 1;
00023
00024         public virtual int maxStackCount => 64;
00025
00026         public static int ID => 0;
00027         #endregion
00028
00029         #region Constructors
00030         public Item()
00031         {
00032             itemName = "TestItem";
00033         }
00034
00035         public Item(string name)
00036         {
00037             itemName = name;
00038         }
00039         #endregion
00040
00041         #region Player Item Interactions
00042         public virtual bool InteractWithObject()
00043         {
00044             return false;
00045         }
00046         #endregion
00047
00048         #region Item Stuff
00049         public virtual GameObject GetGameObject() { return null; }
00050
00051         public override string GetItemID()
00052         {
00053             return $"{GetHashCode()}";
00054         }
00055
00056         public virtual Sprite GetItemSprite()
00057         {
00058             return SpriteDictionary.GetSprite("TestSprite");
00059         }
00060
00061         public override string GetItemName()
00062         {
00063             return $"{itemName}";
00064         }
00065         #endregion
00066
00067         #region Item Mesh
00068         public virtual Tile TexturePosition(Direction direction)

```

```

00111         {
00112             return new Tile() { x = 1, y = 9 };
00113         }
00114
00123     public virtual MeshData ItemMesh(int x, int y, int z,
MeshData meshData)
00124     {
00125         /** adds all faces of the item to the mesh as all faces could be seen at any time
00126         meshData = FaceDataUp(x, y, z, meshData, true, 0.25f);
00127         meshData = FaceDataDown(x, y, z, meshData, true, 0.25f);
00128         meshData = FaceDataNorth(x, y, z, meshData, true, 0.25f);
00129         meshData = FaceDataEast(x, y, z, meshData, true, 0.25f);
00130         meshData = FaceDataSouth(x, y, z, meshData, true, 0.25f);
00131         meshData = FaceDataWest(x, y, z, meshData, true, 0.25f);
00132
00133         return meshData;
00134     }
00135
00141     public virtual Vector2[] FaceUVs(Direction direction)
00142     {
00143         /** only 4 uvs per face
00144         Vector2[] UVs = new Vector2[4];
00145         Tile tilePos = TexturePosition(direction);
00146
00147         /** sets the UVs for each vertex
00148         UVs[0] = new THVector2(tileSize * tilePos.x + tileSize - 0.01f, tileSize * tilePos.
y + 0.01f);
00149         UVs[1] = new THVector2(tileSize * tilePos.x + tileSize - 0.01f, tileSize * tilePos.
y + tileSize - 0.01f);
00150         UVs[2] = new THVector2(tileSize * tilePos.x + 0.01f, tileSize * tilePos.
y + tileSize - 0.01f);
00151         UVs[3] = new THVector2(tileSize * tilePos.x + 0.01f, tileSize * tilePos.
y + 0.01f);
00152
00153         return UVs;
00154     }
00155
00166     protected virtual MeshData FaceDataUp(int x, int y, int z,
MeshData meshData, bool addToRenderMesh = true, float blockSize = 0.5f)
00167     {
00168         /** Adds vertices in a anti-clockwise order
00169         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z +
blockSize), addToRenderMesh, Direction.UP);
00170         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z +
blockSize), addToRenderMesh, Direction.UP);
00171         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z -
blockSize), addToRenderMesh, Direction.UP);
00172         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z -
blockSize), addToRenderMesh, Direction.UP);
00173
00174         /** adds teh tirs for the quad
00175         meshData.AddQuadTriangles(addToRenderMesh);
00176
00177         /** if the data should be added to the render mesh also add the uvs to the mesh
00178         if (addToRenderMesh)
00179             meshData.uv.AddRange(FaceUVs(Direction.UP));
00180
00181         return meshData;
00182     }
00183
00194     protected virtual MeshData FaceDataDown(int x, int y, int z,
MeshData meshData, bool addToRenderMesh = true, float blockSize = 0.5f)
00195     {
00196         /** Adds vertices in a anti-clockwise order
00197         meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z -
blockSize), addToRenderMesh);
00198         meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z -
blockSize), addToRenderMesh);
00199         meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z +
blockSize), addToRenderMesh);
00200         meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z +
blockSize), addToRenderMesh);
00201
00202         /** adds teh tirs for the quad
00203         meshData.AddQuadTriangles(addToRenderMesh);
00204
00205         /** if the data should be added to the render mesh also add the uvs to the mesh
00206         if (addToRenderMesh)
00207             meshData.uv.AddRange(FaceUVs(Direction.DOWN));
00208
00209         return meshData;
00210     }
00211
00222     protected virtual MeshData FaceDataNorth(int x, int y, int z,
MeshData meshData, bool addToRenderMesh = true, float blockSize = 0.5f)
00223     {
00224         /** Adds vertices in a anti-clockwise order

```



```

00225         meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z +
blockSize), addToRenderMesh);
00226         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z +
blockSize), addToRenderMesh);
00227         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z +
blockSize), addToRenderMesh);
00228         meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z +
blockSize), addToRenderMesh);
00229
00230         /** adds teh tirs for the quad
00231         meshData.AddQuadTriangles(addToRenderMesh);
00232
00233         /** if the data should be added to the render mesh also add the uvs to the mesh
00234         if (addToRenderMesh)
00235             meshData.uv.AddRange(FaceUVs(Direction.NORTH));
00236
00237         return meshData;
00238     }
00239
00250     protected virtual MeshData FaceDataEast(int x, int y, int z,
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 -
blockSize), addToRenderMesh);
00254         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z -
blockSize), addToRenderMesh);
00255         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z +
blockSize), addToRenderMesh);
00256         meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z +
blockSize), addToRenderMesh);
00257
00258         /** adds teh tirs for the quad
00259         meshData.AddQuadTriangles(addToRenderMesh);
00260
00261         /** if the data should be added to the render mesh also add the uvs to the mesh
00262         if (addToRenderMesh)
00263             meshData.uv.AddRange(FaceUVs(Direction.EAST));
00264
00265         return meshData;
00266     }
00267
00278     protected virtual MeshData FaceDataSouth(int x, int y, int z,
MeshData meshData, bool addToRenderMesh = true, float blockSize = 0.5f)
00279     {
00280         /** Adds vertices in a anti-clockwise order
00281         meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z -
blockSize), addToRenderMesh);
00282         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z -
blockSize), addToRenderMesh);
00283         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z -
blockSize), addToRenderMesh);
00284         meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z -
blockSize), addToRenderMesh);
00285
00286         /** adds teh tirs for the quad
00287         meshData.AddQuadTriangles(addToRenderMesh);
00288
00289         /** if the data should be added to the render mesh also add the uvs to the mesh
00290         if (addToRenderMesh)
00291             meshData.uv.AddRange(FaceUVs(Direction.SOUTH));
00292
00293         return meshData;
00294     }
00295
00306     protected virtual MeshData FaceDataWest(int x, int y, int z,
MeshData meshData, bool addToRenderMesh = true, float blockSize = 0.5f)
00307     {
00308         /** Adds vertices in a anti-clockwise order
00309         meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z +
blockSize), addToRenderMesh);
00310         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z +
blockSize), addToRenderMesh);
00311         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z -
blockSize), addToRenderMesh);
00312         meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z -
blockSize), addToRenderMesh);
00313
00314         /** adds teh tirs for the quad
00315         meshData.AddQuadTriangles(addToRenderMesh);
00316
00317         /** if the data should be added to the render mesh also add the uvs to the mesh
00318         if (addToRenderMesh)
00319             meshData.uv.AddRange(FaceUVs(Direction.WEST));
00320
00321         return meshData;
00322     }

```

```

00323         #endregion
00324
00325         #region Interfaces
00326         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()
00349         {
00350             return $"{itemName} \nID: {GetItemID()}";
00351         }
00352
00357         public override int GetHashCode()
00358         {
00359             return ID;
00360         }
00361
00367         public override bool Equals(object obj)
00368         {
00369             if (!(obj is Item))
00370                 return false;
00371
00372             return this == (obj as Item);
00373         }
00374
00381         public static bool operator ==(Item a, Item b)
00382         {
00383             if (ReferenceEquals(a, null) && ReferenceEquals(b, null))
00384                 return true;
00385             if (ReferenceEquals(a, null) || ReferenceEquals(b, null))
00386                 return false;
00387
00388             if (a.GetItemID() == b.GetItemID())
00389                 return true;
00390
00391             return false;
00392         }
00393
00400         public static bool operator !=(Item a, Item b)
00401         {
00402             return !(a == b);
00403         }
00404         #endregion
00405     }
00406
00410     [Serializable]
00411     public struct Tile
00412     {
00416         public int x;
00420         public int y;
00421     }
00422 }

```

7.73 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/ItemGameObject.cs File Reference

Classes

- class [BeeGame.Items.ItemGameObject](#)
Interface between item and inity gameobjects

Namespaces

- namespace [BeeGame.Items](#)

7.74 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 {
00011     [RequireComponent(typeof(Rigidbody))]
00012     [RequireComponent(typeof(MeshFilter))]
00013     [RequireComponent(typeof(MeshRenderer))]
00014     [RequireComponent(typeof(BoxCollider))]
00015     public class ItemGameObject : MonoBehaviour
00016     {
00017         public Item item;
00018         public GameObject go;
00019
00020         private void Start()
00021         {
00022             if (!item.usesGameObject)
00023                 MakeMesh();
00024
00025             if (item.usesGameObject)
00026             {
00027                 Instantiate(item.GetGameObject(), transform, false);
00028                 transform.localScale = new Vector3(0.5f, 0.5f, 0.5f);
00029             }
00030         }
00031
00032         private void Update()
00033         {
00034             if (transform.position.y < -100)
00035             {
00036                 Destroy(gameObject);
00037             }
00038         }
00039
00040         void MakeMesh()
00041         {
00042             MeshData meshData = new MeshData();
00043             if (item != null)
00044                 meshData = item.ItemMesh(0, 0, 0, meshData);
00045
00046             Mesh mesh = new Mesh()
00047             {
00048                 vertices = meshData.verts.ToArray(),
00049                 triangles = meshData.tris.ToArray(),
00050                 uv = meshData.uv.ToArray()
00051             };
00052
00053             mesh.RecalculateNormals();
00054
00055             GetComponent<MeshFilter>().mesh = mesh;
00056         }
00057     }
00058 }

```

7.75 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](#)

7.76 LoadResources.cs

```

00001 using UnityEngine;
00002 using BeeGame.Core.Dictionarys;
00003
00004 namespace BeeGame
00005 {
00009     public class LoadResources : MonoBehaviour
00010     {
00014         void Awake()
00015         {
00016             Serialization.Serialization.MakeDirectorys();
00017
00018             Serialization.Serialization.LoadPlayerPosition(GameObject.Find("Player").GetComponent<Transform
00019 >());
00020
00020             SpriteDictionary.LoadSprites();
00021             PrefabDictionary.LoadPrefabs();
00022         }
00023     }
00024 }

```

7.77 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/obj/Debug/Temporary↵ GeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs File Reference

7.78 TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs

7.79 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/obj/Debug/Temporary↵ GeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs File Reference

7.80 TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs

7.81 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/obj/Debug/Temporary↵ GeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs File Reference

7.82 TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs

7.83 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](#)

7.84 PlayerLook.cs

```

00001 using UnityEngine;
00002 using BeeGame.Core;
00003
00004 namespace BeeGame.Player
00005 {
00006     public class PlayerLook : MonoBehaviour
00007     {
00008         #region Data
00009         public Transform myTransform;
00010         public Transform cameraTransform;
00011         [Range(0, 360)]
00012         public float rotationLock;
00013         public float speed = 5;
00014         float yRot = 0;
00015         float xRot = 0;
00016         #endregion
00017
00018         #region Unity Methods
00019         void Start()
00020         {
00021             Cursor.lockState = CursorLockMode.Locked;
00022             Cursor.visible = false;
00023         }
00024
00025         void Update()
00026         {
00027             /**the look will not update when a inventory GUI is open
00028             if (!THInput.isAnotherInventoryOpen)
00029             {
00030                 Look();
00031             }
00032         }
00033         #endregion
00034
00035         #region Methods
00036         void Look()
00037         {
00038             //Only X/Y rotation needed as Z rotation would be wierd
00039             yRot += Input.GetAxis("Mouse X") * speed * Time.timeScale;
00040             xRot -= Input.GetAxis("Mouse Y") * speed * Time.timeScale;
00041
00042             //clamps the X rotation so the player camera cannot do flips
00043             xRot = Mathf.Clamp(xRot, -rotationLock, rotationLock);
00044
00045             myTransform.rotation = Quaternion.Euler(0, yRot, 0);
00046             cameraTransform.localRotation = Quaternion.Euler(xRot, 0, 0);
00047         }
00048         #endregion
00049     }
00050 }

```

7.85 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](#)

7.86 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()
00050         {
00051             myRigidBody = GetComponent<Rigidbody>();
00052
00053             //i want to use myown 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()
00088         {
00089             //Calculate the speed we want to achieve
00090             Vector3 targetVelocity = new Vector3(THInput.GetAxis("Horizontal"), 0,
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, maxVelocity);
00100             velocityChange.z = Mathf.Clamp(velocityChange.z, -maxVelocity, 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, 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 }

```

7.87 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/SavePlayerPosition.cs File Reference ↩

Classes

- class [BeeGame.Player.SavePlayerPosition](#)
Saves the player postion

Namespaces

- namespace [BeeGame.Player](#)

7.88 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
00028             counter--;
00029         }
00030     }
00031 }

```

7.89 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](#)

7.90 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 {
00012     public class Selector : MonoBehaviour
00013     {
00014         #region Data
00015         public GameObject selector;
00016
00017         public PlayerInventory playerInventory;
00018
00019         public LayerMask layers;
00020         private RaycastHit hit;
00021
00022         public int selectedHotbarSlot = 27;
00023     #endregion
00024
00025     #region Unity Methods
00026     void Awake()
00027     {
00028         selector = Instantiate(selector);
00029     }
00030
00031     void FixedUpdate()
00032     {
00033         if (!IsAnotherInventoryOpen)
00034             UpdateSelector();
00035     }
00036
00037     void Update()
00038     {
00039         if (!IsAnotherInventoryOpen)
00040         {
00041             if (GetButtonDown("Break Block"))
00042                 BreakBlock();
00043             if (GetButtonDown("Place"))
00044                 PlaceBlock();
00045         }
00046     }
00047     #endregion
00048
00049     #region Update
00050     void UpdateSelector()
00051     {
00052         if (Physics.Raycast(transform.position, transform.forward, out hit, 15, layers))
00053         {
00054             selector.SetActive(true);
00055             selector.transform.position = GetBlockPos(hit);
00056             /*selector.SetActive(BlockInPosition(GetBlockPos(hit),
00057 hit.collider.GetComponent<Chunk>()));
00058         }
00059         else
00060         {
00061             selector.SetActive(false);
00062         }
00063         SelectedSlot();
00064     }
00065
00066     void SelectedSlot()
00067     {
00068         /* adds 1 to the selected slot and if that is out of range set it to the first hotbar slot
00069         if (Input.GetAxis("Mouse ScrollWheel") > 0)
00070         {
00071             selectedHotbarSlot += 1;
00072             if (selectedHotbarSlot == 36)
00073                 selectedHotbarSlot = 27;
00074         }
00075         /* removes one from the hotbar selector and if the selector would be inside the inventory set
00076         it to the last slot in the hotbar
00077         else if (Input.GetAxis("Mouse ScrollWheel") < 0)
00078         {
00079             selectedHotbarSlot -= 1;
00080             if (selectedHotbarSlot == 26)
00081                 selectedHotbarSlot = 35;
00082         }
00083
00084         transform.parent.GetComponentInChildren<PlayerInventory>().SelectedSlot (

```



```

        selectedHotbarSlot);
00116     }
00117     #endregion
00118
00119     #region Break/Place
00120     void BreakBlock()
00121     {
00122         Chunk chunk = GetChunk(selector.transform.position);
00123
00124         Block block = chunk.world.GetBlock((int)selector.transform.position.x, (int)selector.
transform.position.y, (int)selector.transform.position.z);
00128
00129         if (!block.breakable)
00130             return;
00131
00132         chunk.world.SetBlock((int)selector.transform.position.x, (int)selector.transform.position.
y, (int)selector.transform.position.z, new Air(), true);
00133         /* set to changed so when block is placed down again it will be saved
00134         block.changed = true;
00135         block.BreakBlock(selector.transform.position);
00136     }
00137
00141     void PlaceBlock()
00142     {
00143         Chunk chunk = GetChunk(selector.transform.position);
00144
00145         if (chunk == null)
00146             return;
00147
00148         if (!chunk.GetBlock((int)selector.transform.position.x - chunk.
chunkWorldPos.x, (int)selector.transform.position.y - chunk.
chunkWorldPos.y, (int)selector.transform.position.z - chunk.
chunkWorldPos.z).InteractWithBlock(playerInventory))
00149             /* gets the item in the hotbar and if the item is placeable place it
00150             if (transform.parent.GetComponentInChildren<PlayerInventory>().
GetItemFromHotBar(selectedHotbarSlot, out Item blockToPlace))
00151                 chunk.world.SetBlock((int)(selector.transform.position.x + hit.normal.x), (int)(
selector.transform.position.y + hit.normal.y), (int)(selector.transform.position.z + hit.normal.z), (
Block)blockToPlace.CloneObject(), true);
00152     }
00153     #endregion
00154 }
00155 }

```

7.91 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Properties/AssemblyInfo.cs File Reference

7.92 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

```

7.93 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Quest/QuestBook.cs File Reference

Classes

- class [BeeGame.Quest.QuestBook](#)

Namespaces

- namespace [BeeGame.Quest](#)

7.94 QuestBook.cs

```

00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using System.Text;
00005 using BeeGame.Core.Dictionarys;
00006 using BeeGame.Items;
00007 using UnityEngine;
00008
00009 namespace BeeGame.Quest
00010 {
00011     public class QuestBook : Item
00012     {
00013         public override int maxStackCount => 1;
00014
00015         public QuestBook() : base("Quest Book")
00016         {
00017
00018         }
00019
00020         public override bool InteractWithObject()
00021         {
00022             return true;
00023         }
00024
00025         public override Sprite GetItemSprite()
00026         {
00027             return SpriteDictionary.GetSprite("TestSprite");
00028         }
00029
00030         public override int GetHashCode()
00031         {
00032             return 10;
00033         }
00034     }
00035 }
00036

```

7.95 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`

7.96 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 /**-----
00010
00011 namespace BeeGame.Resources {
00012     using System;
00013     using System.Collections.Generic;
00014     using UnityEngine;
00015
00019     /** This class was auto-generated by the StronglyTypedResourceBuilder
00020     /** class via a tool like ResGen or Visual Studio.
00021     /** To add or remove a member, edit your .ResX file then rerun ResGen
00022     /** with the /str option, or rebuild your VS project.
00023     [global::System.CodeDom.Compiler.GeneratedCodeAttribute("
System.Resources.Tools.StronglyTypedResourceBuilder", "4.0.0.0")]
00024     [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
00025     [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
00026     internal class Resources {
00027
00028         private static global::System.Resources.ResourceManager
resourceMan;
00029
00030         private static global::System.Globalization.CultureInfo resourceCulture;
00031
00032         [global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance", "
CA1811:AvoidUncalledPrivateCode")]
00033         internal Resources() {
00034         }
00035
00039         [global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.
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.
ResourceManager("BeeGame.Resources.Resources", typeof(Resources).Assembly);
00044                     resourceMan = temp;
00045                 }
00046                 return resourceMan;
00047             }
00048         }
00049
00054         [global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.
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
00084         internal static Dictionary<string, Sprite> GetSprites()
00085         {
00086             string[] splitCharacters = new string[] { ",", " " };

```

```

00087         object obj = ResourceManager.GetObject("Sprites", resourceCulture);
00088
00089         string text = System.Text.Encoding.Default.GetString((byte[])obj);
00090         string lineText = "";
00091         string[] splitText;
00092         Texture2D tex;
00093         Dictionary<string, Sprite> sprites = new Dictionary<string, Sprite>();
00094
00095         for (int i = 0; i < text.Length; i++)
00096         {
00097             if (text[i] != '\n')
00098             {
00099                 lineText += text[i];
00100             }
00101             else
00102             {
00103                 splitText = lineText.Split(splitCharacters, StringSplitOptions.RemoveEmptyEntries);
00104                 lineText = "";
00105                 tex = UnityEngine.Resources.Load("Sprites/" + splitText[1].Remove(splitText[
00106 1].Length - 1, 1)) as Texture2D;
00107                 sprites.Add(splitText[0], Sprite.Create(tex, new UnityEngine.Rect(0, 0, tex.
00108 width, tex.height), Vector2.zero));
00109             }
00110
00111             splitText = lineText.Split(splitCharacters, StringSplitOptions.RemoveEmptyEntries);
00112             lineText = "";
00113             tex = UnityEngine.Resources.Load("Sprites/" + splitText[1]) as Texture2D;
00114             sprites.Add(splitText[0], Sprite.Create(tex, new UnityEngine.Rect(0, 0, tex.width,
00115 tex.height), Vector2.zero));
00116
00117             return sprites;
00118         }
00119
00120         internal static Dictionary<string, GameObject> GetPrefabs()
00121         {
00122             string[] splitCharacters = new string[] { ",", " " };
00123             object obj = ResourceManager.GetObject("Prefabs", resourceCulture);
00124
00125             string text = System.Text.Encoding.Default.GetString((byte[])obj);
00126             text = text.Remove(0, 3);
00127             string lineText = "";
00128             string[] splitText;
00129             Dictionary<string, GameObject> objects = new Dictionary<string, GameObject>();
00130
00131             for (int i = 0; i < text.Length; i++)
00132             {
00133                 if (text[i] != '\n')
00134                 {
00135                     lineText += text[i];
00136                 }
00137                 else
00138                 {
00139                     splitText = lineText.Split(splitCharacters, StringSplitOptions.RemoveEmptyEntries);
00140                     lineText = "";
00141                     objects.Add(splitText[0], UnityEngine.Resources.Load("Prefabs/" + splitText[
00142 1].Remove(splitText[1].Length - 1, 1)) as GameObject);
00143                 }
00144
00145                 splitText = lineText.Split(splitCharacters, StringSplitOptions.RemoveEmptyEntries);
00146                 lineText = "";
00147                 objects.Add(splitText[0], UnityEngine.Resources.Load("Prefabs/" + splitText[1]) as
00148 GameObject);
00149
00150                 return objects;
00151             }
00152         }

```

7.97 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](#)

7.98 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.Blocks;
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 MakeDirectorys()
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.
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.
RemoveEmptyEntries);
00060
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 }

```

```

00105         #endregion
00106
00107     #region Inventorys
00108     public static void SerializeInventory(Inventory.Inventory inventory, string inventoryName)
00109     {
00110         string inventorySavePath = $"{savePath}/Inventorys";
00111
00112         if (!Directory.Exists(inventorySavePath))
00113             Directory.CreateDirectory(inventorySavePath);
00114
00115         SaveFile(inventory.GetAllItems(), $"{inventorySavePath}/{inventoryName}.dat");
00116     }
00117
00118     public static void DeSerializeInventory(Inventory.Inventory inventory,
00119 string inventoryName)
00120     {
00121         /** make the path
00122         string inventorySavePath = $"{savePath}/Inventorys/{inventoryName}.dat";
00123
00124         /** checks that the file exists
00125         if (!File.Exists(inventorySavePath))
00126         {
00127             for (int i = 0; i < inventory.items.itemsInInventory.Length; i++)
00128             {
00129                 inventory.items.itemsInInventory[i] = null;
00130             }
00131
00132             SerializeInventory(inventory, inventoryName);
00133
00134             return;
00135         }
00136
00137         inventory.SetAllItems((ItemsInInventory) LoadFile($"{inventorySavePath}"));
00138     }
00139     #endregion
00140
00141     #region Chunk
00142     public static void SaveChunk(Chunk chunk)
00143     {
00144         /** saves the blocks
00145         SaveChunk save = new SaveChunk(chunk.blocks);
00146
00147         /** if no block was changed return early
00148         if (save.blocks.Count == 0)
00149             return;
00150
00151         /** otherwise save the file
00152         string saveFile = $"{savePath}/{FileName(chunk.chunkWorldPos)}.dat";
00153
00154         SaveFile(save, saveFile);
00155     }
00156
00157     public static bool LoadChunk(Chunk chunk)
00158     {
00159         /** gets the save file
00160         string saveFile = $"{savePath}/{FileName(chunk.chunkWorldPos)}.dat";
00161
00162         /** if the file does not exist return false
00163         if (!File.Exists(saveFile))
00164             return false;
00165
00166         /** set all of the changed blocks in the chunk
00167         SaveChunk save = (SaveChunk) LoadFile(saveFile);
00168
00169         foreach (var block in save.blocks)
00170         {
00171             chunk.blocks[block.Key.x, block.Key.y, block.Key.z] = block.Value;
00172         }
00173
00174         return true;
00175     }
00176
00177     public static string FileName(ChunkWorldPos pos)
00178     {
00179         return $"{pos.x}, {pos.y}, {pos.z}";
00180     }
00181     #endregion
00182
00183     #region Save/Load Files
00184     private static void SaveFile(object obj, string file)
00185     {
00186         BinaryFormatter bf = new BinaryFormatter();
00187         FileStream fs = new FileStream(file, FileMode.OpenOrCreate);
00188
00189         try
00190         {
00191             bf.Serialize(fs, obj);
00192         }
00193     }

```

```

00224         }
00225         catch (SerializationException e)
00226         {
00227             Debug.Log($"Serialization Exception: {e}");
00228             throw new SerializationException();
00229         }
00230         finally
00231         {
00232             fs.Close();
00233         }
00234     }
00235
00241     private static object LoadFile(string file)
00242     {
00243         BinaryFormatter bf = new BinaryFormatter();
00244         FileStream fs = new FileStream(file, FileMode.Open);
00245
00246         try
00247         {
00248             return bf.Deserialize(fs);
00249         }
00250         catch (SerializationException e)
00251         {
00252             Debug.Log($"Deserialization Exception {e}");
00253             throw new SerializationException();
00254         }
00255         finally
00256         {
00257             fs.Close();
00258         }
00259     }
00260     #endregion
00261 }
00262 }

```

7.99 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/SpawnItem.cs File Reference

Classes

- class [BeeGame.SpawnItem](#)

Namespaces

- namespace [BeeGame](#)

7.100 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
GameObject, transform.position, Quaternion.identity) as GameObject;
00017             go.GetComponent<ItemGameObject>().item = new Bee(
BeeType.DRONE, new NormalBee() { pSpecies = BeeSpecies.FOREST });
00018
00019             go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
transform.position, Quaternion.identity) as GameObject;
00020             go.GetComponent<ItemGameObject>().item = new Bee(
BeeType.PRINCESS, new NormalBee() { pSpecies = BeeSpecies.FOREST });

```

```

00021
00022         go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
transform.position, Quaternion.identity) as GameObject;
00023         go.GetComponent<ItemGameObject>().item = new Bee(
BeeType.DRONE, new NormalBee() { pSpecies = BeeSpecies.COMMON, sSpecies =
BeeSpecies.COMMON });
00024
00025         go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
transform.position, Quaternion.identity) as GameObject;
00026         go.GetComponent<ItemGameObject>().item = new Bee(
BeeType.PRINCESS, new NormalBee() { pSpecies = BeeSpecies.COMMON, sSpecies =
BeeSpecies.COMMON });
00027
00028         //go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
transform.position, Quaternion.identity) as GameObject;
00029         //go.GetComponent<ItemGameObject>().item = new Bee(BeeType.QUEEN, new QueenBee());
00030
00031         go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
transform.position, Quaternion.identity) as GameObject;
00032         go.GetComponent<ItemGameObject>().item = new HoneyComb(
HoneyCombType.ICEY);
00033
00034         go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
transform.position, Quaternion.identity) as GameObject;
00035         go.GetComponent<ItemGameObject>().item = new HoneyComb(
HoneyCombType.HONEY);
00036
00037         go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
transform.position, Quaternion.identity) as GameObject;
00038         go.GetComponent<ItemGameObject>().item = new Chest();
00039         go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
transform.position, Quaternion.identity) as GameObject;
00040         go.GetComponent<ItemGameObject>().item = new Chest();
00041
00042         go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
transform.position, Quaternion.identity) as GameObject;
00043         go.GetComponent<ItemGameObject>().item = new Apiary();
00044
00045         go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
transform.position, Quaternion.identity) as GameObject;
00046         go.GetComponent<ItemGameObject>().item = new
CraftingTable();
00047     }
00048
00049     private void OnDrawGizmos()
00050     {
00051         //Gizmos.color = Color.green;
00052         //Gizmos.DrawSphere(transform.position, 0.5f);
00053     }
00054 }
00055 }

```

7.101 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](#)

7.102 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 checkNeighbouringChunks = 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(checkNeighbouringChunks)
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 checkNeighbouringChunks =
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             }

```

```

00152         if (checkNeighbouringChunks)
00153             /** if the block is not in the chunk find its chunk and set it their
00154             world.SetBlock(chunkWorldPos.x + x, chunkWorldPos.y + y, chunkWorldPos.
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
secondary thread...thanks unity
00216
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         /** redoes the normals incase they got messed up
00230         filter.mesh.RecalculateNormals();
00231         /** is this necessary as it causes alsot of lag?
00232     }
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
their is no point
00240         if (this.mesh.verts.Count == 0)
00241             return;
00242
00243         /** if the render and collision meshes should be shared set the render mesh to the collision
mesh otherwise make a collision mesh
00244         if (this.mesh.shareMeshes)
00245         {
00246             world.chunkHasMadeCollisionMesh = true;
00247             applyCollisionMesh = false;
00248             meshCollider.sharedMesh = filter.mesh;
00249             return;
00250         }
00251
00252         world.chunkHasMadeCollisionMesh = true;
00253         /** Applying the mesh takes the longest but nothing can be done with the mesh class in a
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     }
00270     #endregion
00271 }
00272 }

```

7.103 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/Load↵

Chunks.cs File Reference

Classes

- class [BeeGame.Terrain.Chunks.LoadChunks](#)

Loads the [Chunks](#) around the player

Namespaces

- namespace [BeeGame.Terrain.Chunks](#)

7.104 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), new
00042         ChunkWorldPos( 2, 0, -2), new ChunkWorldPos( 2, 0, 2),
00043         new 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), new ChunkWorldPos(-2, 0, -3),
00052         new ChunkWorldPos(-2, 0, 3), new
00053         ChunkWorldPos( 2, 0, -3), new ChunkWorldPos( 2, 0, 3),
00054         new ChunkWorldPos(-3, 0, -3), new
00055         ChunkWorldPos(-3, 0, 3), new ChunkWorldPos( 3, 0, -3),
00056         new ChunkWorldPos( 3, 0, 3)
00057     };
00058     }
00059 }

```

```

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), new
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( 5, 0, -3), new
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( 1, 0, -6), new
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), new
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
ChunkWorldPos( 7, 0, -3), new ChunkWorldPos( 7, 0, 3),

```

```

00064         new ChunkWorldPos(-6, 0, -5), new
ChunkWorldPos(-6, 0, 5), new ChunkWorldPos(-5, 0, -6), new
ChunkWorldPos(-5, 0, 6), new ChunkWorldPos(5, 0, -6),
00065         new ChunkWorldPos(5, 0, 6), new
ChunkWorldPos(6, 0, -5), new ChunkWorldPos(6, 0, 5) };
00066
00070     private static ChunkWorldPos[] nearbyChunks = 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),
00071         new
ChunkWorldPos(1, 0, 1), new ChunkWorldPos(1, 0, -1), new
ChunkWorldPos(-1, 0, 1), new ChunkWorldPos(-1, 0, -1)};
00072
00076     private static int timer = 0;
00077     #endregion
00078
00082     private void Start()
00083     {
00084         LandGeneration.Terrain.world = world;
00085     }
00086
00090     void Update()
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     }
00103
00111     void ApplyCollisionMeshToNearbyChunks()
00112     {
00113         /* gets the player position in chunk coordinates
00114         ChunkWorldPos playerPos = new ChunkWorldPos(Mathf.FloorToInt(
transform.position.x / Chunk.chunkSize) * Chunk.chunkSize, Mathf.FloorToInt(transform.
position.y / Chunk.chunkSize) * Chunk.chunkSize, Mathf.FloorToInt(transform.
position.z / Chunk.chunkSize) * Chunk.chunkSize);
00115
00116         for (int i = 0; i < nearbyChunks.Length; i++)
00117         {
00118             ChunkWorldPos chunkPos = new ChunkWorldPos(nearbyChunks[i].x *
Chunk.chunkSize + playerPos.x, 0, nearbyChunks[i].z * Chunk.
chunkSize + playerPos.z);
00119
00120             for (int j = -1; j < 2; j++)
00121             {
00122                 Chunk nearbyChunk = world.GetChunk(chunkPos.x, j *
Chunk.chunkSize, chunkPos.z);
00123
00124                 if (nearbyChunk != null)
00125                     nearbyChunk.applyCollisionMesh = true;
00126             }
00127         }
00128     }
00129
00133     void LoadAndRenderChunks()
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
chunk is missed because chunks are removed from the list as they are made
00139             for (int i = buildList.Count - 1, j = 0; i >= 0 && j < 8; i--, j++)
00140             {
00141                 BuildChunk(buildList[0]);
00142                 buildList.RemoveAt(0);
00143             }
00144         }
00145     }
00146
00150     void FindChunksToLoad()
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.chunkSize) * Chunk.chunkSize, Mathf.FloorToInt(
transform.position.y / Chunk.chunkSize) * Chunk.chunkSize, Mathf.FloorToInt(transform.
position.z / Chunk.chunkSize) * Chunk.chunkSize);
00156
00157             /* check all of the chunk positions and if that position does not have a chunk in it make

```

```

    it
00158         for (int i = 0; i < chunkPositions.Length; i++)
00159         {
00160             ChunkWorldPos newChunkPos = new ChunkWorldPos(chunkPositions[
i].x * Chunk.chunkSize + playerPos.x, 0, chunkPositions[i].z *
Chunk.chunkSize + playerPos.z);
00161
00162             Chunk newChunk = world.GetChunk(newChunkPos.x, newChunkPos.
y, newChunkPos.z);
00163
00164             if (newChunk != null && (newChunk.rendered || buildList.Contains(newChunkPos)))
00165                 continue;
00166
00167             for (int y = -1; y < 2; y++)
00168             {
00169                 for (int x = newChunkPos.x - Chunk.chunkSize; x < newChunkPos.
x + Chunk.chunkSize; x += Chunk.chunkSize)
00170                 {
00171                     for (int z = newChunkPos.z - Chunk.chunkSize; z < newChunkPos.
z + Chunk.chunkSize; z += Chunk.chunkSize)
00172                     {
00173                         buildList.Add(new ChunkWorldPos(x, y *
Chunk.chunkSize, z));
00174                     }
00175                 }
00176             }
00177             return;
00178         }
00179     }
00180 }
00181
00186 void BuildChunk(ChunkWorldPos pos)
00187 {
00188     if (world.GetChunk(pos.x, pos.y, pos.z) == null)
00189         world.CreateChunk(pos.x, pos.y, pos.z);
00190 }
00191
00196 bool DeleteChunks()
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 }
00225 }
00226 }

```

7.105 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/Mesh↵ Data.cs File Reference

Classes

- class [BeeGame.Terrain.Chunks.MeshData](#)

The data for a [Chunks's](#) Mesh

Namespaces

- namespace [BeeGame.Terrain.Chunks](#)

7.106 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,
Direction direction = Direction.DOWN)
00075         {
00076             if (addToRenderMesh)
00077                 verts.Add(pos);
00078
00079             /* if the vertice is on the top face make its position slightly smaller
00080             if(direction == Direction.UP)
00081                 colVerts.Add(pos - new THVector3(0.01f, 0, 0.01f));
00082         }
00083
00091         public void AddTriangle(int tri)
00092         {
00093             tris.Add(tri);
00094
00095             colTris.Add(tri - (verts.Count - colVerts.Count));
00096         }
00097     }
00098 }

```

7.107 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/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](#)

7.108 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 }

```

7.109 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/ChunkWorldPos.cs File Reference

Classes

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

Namespaces

- namespace [BeeGame.Terrain](#)

7.110 ChunkWorldPos.cs

```

00001 using System;
00002 using BeeGame.Core;
00003
00004 namespace BeeGame.Terrain
00005 {
00009     [Serializable]
00010     public struct ChunkWorldPos
00011     {
00015         public int x, y, z;
00016
00023         public ChunkWorldPos(int x, int y, int z)
00024         {
00025             this.x = x;
00026             this.y = y;
00027             this.z = z;
00028         }

```



```

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", "
CA2231:OverloadOperatorEqualsOnOverridingValueTypeEquals")]
00041     public override bool Equals(object obj)
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     }
00055
00063     public override int GetHashCode()
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     }
00076
00081     public static implicit operator THVector3(ChunkWorldPos pos)
00082     {
00083         return new THVector3(pos.x, pos.y, pos.z);
00084     }
00085
00093     public static explicit operator ChunkWorldPos(THVector3 pos)
00094     {
00095         return new ChunkWorldPos((int)pos.x, (int)pos.y, (int)pos.
z);
00096     }
00097 }
00098 }

```

7.111 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](#)

7.112 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.
00010
00011  /**In jurisdictions that recognize copyright laws, the author or authors
00012  /**of this software dedicate any and all copyright interest in the
00013  /**software to the public domain. We make this dedication for the benefit
00014  /**of the public at large and to the detriment of our heirs and
00015  /**successors. We intend this dedication to be an overt act of
00016  /**relinquishment in perpetuity of all present and future rights to this
00017  /**software under copyright law.
00018
00019  /**THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
00020  /**EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
00021  /**MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
00022  /**IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY CLAIM, DAMAGES OR
00023  /**OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
00024  /**ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
00025  /**OTHER DEALINGS IN THE SOFTWARE.
00026
00027  /**For more information, please refer to <http://unlicense.org/>
00028
00029
00030 namespace BeeGame.Terrain.LandGeneration.Noise
00031 {
00032     public class SimplexNoise
00033     {
00034         public static float Generate(float x)
00035         {
00036             int i0 = FastFloor(x);
00037             int i1 = i0 + 1;
00038             float x0 = x - i0;
00039             float x1 = x0 - 1.0f;
00040
00041             float n0, n1;
00042
00043             float t0 = 1.0f - x0 * x0;
00044             t0 *= t0;
00045             n0 = t0 * t0 * grad(perm[i0 & 0xff], x0);
00046
00047             float t1 = 1.0f - x1 * x1;
00048             t1 *= t1;
00049             n1 = t1 * t1 * grad(perm[i1 & 0xff], x1);
00050             /** The maximum value of this noise is  $8 \cdot (3/4)^4 = 2.53125$ 
00051             /** A factor of 0.395 scales to fit exactly within [-1,1]
00052             return 0.395f * (n0 + n1);
00053         }
00054
00055         public static float Generate(float x, float y)
00056         {
00057             const float F2 = 0.366025403f; /** F2 =  $0.5 \cdot (\sqrt{3} - 1)$ 
00058             const float G2 = 0.211324865f; /** G2 =  $(3 - \text{Math.sqrt}(3)) / 6$ 
00059
00060             float n0, n1, n2; /** Noise contributions from the three corners
00061
00062             /** Skew the input space to determine which simplex cell we're in
00063             float s = (x + y) * F2; /** Hairy factor for 2D
00064             float xs = x + s;
00065             float ys = y + s;
00066             int i = FastFloor(xs);
00067             int j = FastFloor(ys);
00068
00069             float t = (float)(i + j) * G2;
00070             float X0 = i - t; /** Unskew the cell origin back to (x,y) space
00071             float Y0 = j - t;
00072             float x0 = x - X0; /** The x,y distances from the cell origin
00073             float y0 = y - Y0;
00074
00075             /** For the 2D case, the simplex shape is an equilateral triangle.
00076             /** Determine which simplex we are in.
00077             int i1, j1; /** Offsets for second (middle) corner of simplex in (i,j) coords
00078             if (x0 > y0) { i1 = 1; j1 = 0; } /** lower triangle, XY order: (0,0)->(1,0)->(1,1)
00079             else { i1 = 0; j1 = 1; } /** upper triangle, YX order: (0,0)->(0,1)->(1,1)
00080
00081             /** A step of (1,0) in (i,j) means a step of (1-c,-c) in (x,y), and
00082             /** a step of (0,1) in (i,j) means a step of (-c,1-c) in (x,y), where
00083             /** c =  $(3 - \sqrt{3}) / 6$ 
00084
00085             float x1 = x0 - i1 + G2; /** Offsets for middle corner in (x,y) unskewed coords
00086             float y1 = y0 - j1 + G2;
00087             float x2 = x0 - 1.0f + 2.0f * G2; /** Offsets for last corner in (x,y) unskewed coords
00088             float y2 = y0 - 1.0f + 2.0f * G2;
00089
00090             /** Wrap the integer indices at 256, to avoid indexing perm[] out of bounds
00091             int ii = i % 256;
00092             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 + i1 + 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 = 0.3333333333f;
00145     const float G3 = 0.1666666667f;
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 + 1 + perm[jj + 1 + 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 }

```

7.113 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](#)

7.114 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
00054     public static ChunkWorldPos GetBlockPosFromRayCast(RaycastHit
hit)
00055     {
00056         return new ChunkWorldPos((int)RoundXZ(hit.point.x, hit.normal.x), (int)RoundY(hit.
point.y, hit.normal.y), (int)RoundXZ(hit.point.z, hit.normal.z));
00057     }
00058
00068     static float RoundXZ(float pos, float normal)
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
00102             /* if negative nomrmal
00103             if (normal < 0)
00104             {
00105                 pos = (int)pos;
00106                 return pos;
00107             }
00108             /* if their is no normal
00109
00110             /* if pos is greater than 0.5 we are in the next block so go to it
00111             if ((-pos - (int)-pos) > 0.5)
00112             {
00113                 return (int)pos - 1;
00114             }
00115
00116             return (int)pos;
00117         }
00118     }
00119
00129     static float RoundY(float pos, float normal)
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
00139                 return Mathf.RoundToInt((float)Math.Round(pos, 1)) - normal;
00140             }
00141
00142             if((int)pos % 2 == 0)
00143                 return Mathf.RoundToInt((float)Math.Round(pos, 1)) - normal;
00144
00145             return Mathf.RoundToInt((float)Math.Round(pos, 1));

```

```

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
00179 public static float Round(float pos, float norm, bool adjacent = false)
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 }
00195 #endregion
00196
00197 #region Get Block
00198 public static ChunkWorldPos GetBlockPos(RaycastHit hit, bool adjacent = false)
00200 {
00201     return GetBlockPos(new THVector3()
00202     {
00203         /* rounds the hit to the correct position
00204         x = Round(hit.point.x, hit.normal.x, adjacent),
00205         y = Round(hit.point.y, hit.normal.y, adjacent),
00206         z = Round(hit.point.z, hit.normal.z, adjacent)
00207     });
00208 }
00209
00210
00221 public static Block GetBlock(RaycastHit hit, bool adjacent = false)
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 }
00234
00235 public static Block GetBlock(THVector3 pos)
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 }
00246
00247 public static bool BlockInPosition(THVector3 pos,
Chunk chunk)
00248 {
00249     if (chunk == null)
00250         return false;
00251
00252     if (chunk.GetBlock((int)pos.x, (int)pos.y, (int)pos.z) != new

```

```

    Air()
00253         return true;
00254
00255         return false;
00256     }
00257     #endregion
00258
00259     public static Chunk GetChunk(THVector3 vec3)
00260     {
00261         return world.GetChunk((int)vec3.x, (int)vec3.y, (int)vec3.
z);
00262     }
00263
00264     #region Set Block
00265     public static bool SetBlock(RaycastHit hit, Block block, bool adjacent = false)
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 tryign 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     }
00293     #endregion
00294 }
00295 }

```

7.115 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](#)

7.116 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
Chunk Thread @ {chunk.chunkWorldPos}"};
00085
00086             thread.Start();
00087             return outChunk;
00088         }
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.
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     }
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         /* 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 botton 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     }
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 *
scale) + 1f) * (max / 2f));
00176     }
00177
00187     public static void SetBlock(int x, int y, int z, Blocks.Block block,
Chunk chunk, bool replacesBlocks = false)
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) &&
Chunk.InRange(z))
00196             if (replacesBlocks || chunk.blocks[x, y, z] == null)
00197                 chunk.SetBlock(x, y, z, block, false);
00198     }
00199
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                 {
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     }
00230 }
00231 }

```

7.117 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](#)

7.118 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)
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.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 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         }
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
Chunk chunk))
00101             {
00102                 /** saves the chunk before destroying it in case 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         }
00108         #endregion
00109
00110         #region Block
00111         public void SetBlock(int x, int y, int z, Block block, bool saveChunk = false)
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.
x, y - chunk.chunkWorldPos.y, z - chunk.chunkWorldPos.
z].breakable)
00125         {

```

```

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 neighbouring chunks as when a block is broken it may be in the edge 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 #endregion
00196
00197 void UpdateIfEqual(int value1, int value2, ChunkWorldPos pos)
00198 {
00199     if(value1 == value2)
00200     {
00201         Chunk chunk = GetChunk(pos.x, pos.y, pos.z);
00202
00203         if (chunk != null)
00204             chunk.update = true;
00205     }
00206 }
00207 }
00208 }
00209 }

```

7.119 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/test.cs File Reference

Classes

- class [BeeGame.Test](#)

Namespaces

- namespace [BeeGame](#)

7.120 test.cs

```
00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using System.Text;
00005 using UnityEngine;
00006 using UnityEngine.UI;
00007 using BeeGame.Core.Dictionarys;
00008 using BeeGame.Items;
00009 using BeeGame.Blocks;
00010 using BeeGame.Core;
00011
00012 namespace BeeGame
00013 {
00014     public class Test : MonoBehaviour
00015     {
00016         private void Start()
00017         {
00018             CraftingRecipies.AddShapedRecipie(new object[] { " ", " X ",
00019 " ", "X", Dirt.ID }, new Grass());
00019             CraftingRecipies.AddShaplessRecipie(new object[] { new
Grass(), 1 }, new Dirt());
00020
00021             Events.shapedRecipieCrafted += Print;
00022         }
00023         public void Print(Item item) => print(item.GetItemID());
00024     }
00025 }
```


Index

- AddItem
 - BeeGame::Inventory::ItemsInInventory, [158](#), [159](#)
- AddItemToInventory
 - BeeGame::Inventory::Inventory, [125](#)
- AddItemToSlots
 - BeeGame::Inventory::BlockInventory::Crafting↔
TableInventory, [102](#)
 - BeeGame::Inventory::Inventory, [126](#)
- AddQuadTriangles
 - BeeGame::Terrain::Chunks::MeshData, [170](#)
- AddShapedRecipe
 - BeeGame::Core::Dictionarys::CraftingRecipes, [90](#)
- AddShaplessRecipe
 - BeeGame::Core::Dictionarys::CraftingRecipes, [91](#)
- AddToFloatingItem
 - BeeGame::Inventory::InventorySlot, [132](#)
- AddToSlot
 - BeeGame::Inventory::InventorySlot, [133](#)
- AddTriangle
 - BeeGame::Terrain::Chunks::MeshData, [171](#)
- AddVertices
 - BeeGame::Terrain::Chunks::MeshData, [171](#)
- Air
 - BeeGame::Blocks::Air, [22](#)
- Apiary
 - BeeGame::Blocks::Apiary, [27](#)
- applyCollisionMesh
 - BeeGame::Terrain::Chunks::Chunk, [82](#)
- ApplyCollsionMeshToNearbyChunks
 - BeeGame::Terrain::Chunks::LoadChunks, [164](#)
- ApplyPlayerItems
 - BeeGame::Inventory::ChestInventory, [73](#)
- Awake
 - BeeGame::Inventory::Player_Inventory::Player↔
Inventory, [177](#)
 - BeeGame::LoadResources, [169](#)
 - BeeGame::Player::PlayerMove, [185](#)
 - BeeGame::Player::Selector, [201](#)
- Bedrock
 - BeeGame::Blocks::Bedrock, [43](#)
- Bee
 - BeeGame::Items::Bee, [47](#), [48](#)
- beeColour
 - BeeGame::Core::Dictionarys::BeeDictionarys, [58](#)
- beeCombinationWeights
 - BeeGame::Core::Dictionarys::BeeDictionarys, [58](#)
- beeCombinations
 - BeeGame::Core::Dictionarys::BeeDictionarys, [58](#)
- beeCraftedEvent
 - BeeGame::Core::Events, [111](#)
- BeeEffect
 - BeeGame::Core::Enums, [12](#)
- BeeGame, [10](#)
- BeeGame.Blocks, [11](#)
- BeeGame.Blocks.Air, [22](#)
- BeeGame.Blocks.Apiary, [25](#)
- BeeGame.Blocks.Bedrock, [43](#)
- BeeGame.Blocks.Block, [60](#)
- BeeGame.Blocks.Chest, [67](#)
- BeeGame.Blocks.CraftingTable, [95](#)
- BeeGame.Blocks.Dirt, [107](#)
- BeeGame.Blocks.Grass, [115](#)
- BeeGame.Blocks.Leaves, [160](#)
- BeeGame.Blocks.Wood, [268](#)
- BeeGame.Core, [11](#)
- BeeGame.Core.Dictionarys, [11](#)
- BeeGame.Core.Dictionarys.BeeCombinationDictionary↔
EqualityComparer, [53](#)
- BeeGame.Core.Dictionarys.BeeDictionarys, [55](#)
- BeeGame.Core.Dictionarys.CraftingRecipes, [89](#)
- BeeGame.Core.Dictionarys.PrefabDictionary, [188](#)
- BeeGame.Core.Dictionarys.SpriteDictionary, [220](#)
- BeeGame.Core.Enums, [12](#)
- BeeGame.Core.Events, [110](#)
- BeeGame.Core.Extensions, [112](#)
- BeeGame.Core.THInput, [240](#)
- BeeGame.Core.THVector2, [246](#)
- BeeGame.Core.THVector3, [254](#)
- BeeGame.Core.UnityTypeReplacements, [17](#)
- BeeGame.Core.UnityTypeReplacements.THQuaternion, [245](#)
- BeeGame.Exceptipns, [17](#)
- BeeGame.Exceptipns.CraftingRecipeAdditionException, [88](#)
- BeeGame.Exceptipns.InputException, [123](#)
- BeeGame.Inventory, [17](#)
- BeeGame.Inventory.ApiaryInventory, [37](#)
- BeeGame.Inventory.BlockInventory, [18](#)
- BeeGame.Inventory.BlockInventory.CraftingTable↔
Inventory, [101](#)
- BeeGame.Inventory.ChestInventory, [72](#)
- BeeGame.Inventory.Inventory, [124](#)
- BeeGame.Inventory.InventorySlot, [131](#)
- BeeGame.Inventory.ItemsInInventory, [157](#)
- BeeGame.Inventory.Player_Inventory, [18](#)
- BeeGame.Inventory.Player_Inventory.PlayerInventory, [176](#)
- BeeGame.Items, [18](#)
- BeeGame.Items.AbstractItem, [20](#)
- BeeGame.Items.ApplyColour, [41](#)
- BeeGame.Items.Bee, [46](#)
- BeeGame.Items.HoneyComb, [118](#)
- BeeGame.Items.Item, [140](#)
- BeeGame.Items.ItemGameObject, [155](#)
- BeeGame.Items.NormalBee, [173](#)
- BeeGame.Items.QueenBee, [190](#)
- BeeGame.Items.Tile, [267](#)
- BeeGame.LoadResources, [168](#)
- BeeGame.Player, [18](#)

- BeeGame.Player.PlayerLook, 181
- BeeGame.Player.PlayerMove, 184
- BeeGame.Player.SavePlayerPosition, 199
- BeeGame.Player.Selector, 200
- BeeGame.Quest, 19
- BeeGame.Quest.QuestBook, 192
- BeeGame.Resources, 19
- BeeGame.Resources.Resources, 194
- BeeGame.Serialization, 19
- BeeGame.Serialization.Serialization, 205
- BeeGame.SpawnItem, 219
- BeeGame.Terrain, 19
- BeeGame.Terrain.ChunkWorldPos, 84
- BeeGame.Terrain.Chunks, 19
- BeeGame.Terrain.Chunks.Chunk, 76
- BeeGame.Terrain.Chunks.LoadChunks, 163
- BeeGame.Terrain.Chunks.MeshData, 169
- BeeGame.Terrain.Chunks.SaveChunk, 197
- BeeGame.Terrain.LandGeneration, 20
- BeeGame.Terrain.LandGeneration.Noise, 20
- BeeGame.Terrain.LandGeneration.Noise.SimplexNoise, 212
- BeeGame.Terrain.LandGeneration.Terrain, 222
- BeeGame.Terrain.LandGeneration.TerrainGeneration, 230
- BeeGame.Terrain.LandGeneration.World, 271
- BeeGame.Test, 239
- BeeGame::Blocks::Air
 - Air, 22
 - BlockData, 23
 - BreakBlock, 23
 - GetHashCode, 24
 - ID, 25
 - IsSolid, 24
 - ToString, 24
- BeeGame::Blocks::Apiary
 - Apiary, 27
 - BlockData, 27
 - BreakBlock, 28
 - CombineEffect, 28
 - CombineFertility, 29
 - CombineLifespan, 29
 - CombineProductionSpeed, 30
 - CombineSpecies, 30
 - GetGameObject, 31
 - GetHashCode, 31
 - ID, 37
 - InteractWithBlock, 32
 - MakeBee, 32
 - MakeBees, 33
 - mutationMultiplier, 37
 - myGameobject, 37
 - Rand, 34
 - ReturnChange, 35
 - TexturePosition, 36
 - ToString, 36
- BeeGame::Blocks::Bedrock
 - Bedrock, 43
- BreakBlock, 44
- GetHashCode, 44
- ID, 45
- TexturePosition, 44
- ToString, 45
- BeeGame::Blocks::Block
 - Block, 61
 - BlockData, 62
 - BreakBlock, 63
 - breakable, 66
 - changed, 66
 - GetHashCode, 63
 - GetItemSprite, 64
 - ID, 66
 - InteractWithBlock, 64
 - IsSolid, 64
 - placeable, 66
 - ToString, 65
 - UpdateBlock, 65
- BeeGame::Blocks::Chest
 - BlockData, 68
 - BreakBlock, 69
 - Chest, 68
 - GetGameObject, 69
 - GetHashCode, 69
 - ID, 71
 - InteractWithBlock, 70
 - myGameobject, 71
 - TexturePosition, 70
 - ToString, 71
- BeeGame::Blocks::CraftingTable
 - BlockData, 96
 - BreakBlock, 97
 - CraftingTable, 96
 - GetGameObject, 97
 - GetHashCode, 97
 - GetItemSprite, 98
 - ID, 101
 - InteractWithBlock, 98
 - myGameobject, 101
 - ReturnShapedRecipeItem, 99
 - ReturnShapelessRecipeItem, 100
 - TexturePosition, 100
- BeeGame::Blocks::Dirt
 - Dirt, 108
 - GetHashCode, 108
 - GetItemSprite, 108
 - ID, 110
 - TexturePosition, 109
 - ToString, 109
- BeeGame::Blocks::Grass
 - GetHashCode, 116
 - GetItemSprite, 116
 - Grass, 115
 - ID, 118
 - TexturePosition, 116
 - ToString, 117
 - UpdateBlock, 117

- BeeGame::Blocks::Leaves
 - GetHashCode, 161
 - GetItemSprite, 161
 - ID, 163
 - IsSolid, 161
 - Leaves, 160
 - TexturePosition, 162
 - ToString, 162
- BeeGame::Blocks::Wood
 - GetHashCode, 269
 - GetItemSprite, 270
 - ID, 271
 - TexturePosition, 270
 - ToString, 271
 - Wood, 269
- BeeGame::Core::Dictionarys::BeeCombination↔
 - DictionaryEqualityComparer
 - Equals, 54
 - GetHashCode, 54
- BeeGame::Core::Dictionarys::BeeDictionarys
 - beeColour, 58
 - beeCombinationWeights, 58
 - beeCombinations, 58
 - beeProduce, 59
 - CombColour, 55
 - GetBeeColour, 56
 - GetBeeProduce, 56
 - GetCombColour, 56
 - GetCombinations, 57
 - GetWeights, 57
 - honeyCoubColour, 59
- BeeGame::Core::Dictionarys::CraftingRecipies
 - AddShapedRecipie, 90
 - AddShaplessRecipie, 91
 - GetShapedRecipeltem, 91
 - GetShaplessRecipieResult, 92, 93
 - GetShaplessRecipieString, 93
 - shapedCraftingRecipies, 94
 - shaplessRecipies, 94
- BeeGame::Core::Dictionarys::PrefabDictionary
 - GetPrefab, 189
 - LoadPrefabs, 189
 - prefabDictionary, 190
- BeeGame::Core::Dictionarys::SpriteDictionary
 - GetSprite, 221
 - itemSpriteDictionary, 222
 - LoadSprites, 221
- BeeGame::Core::Enums
 - BeeEffect, 12
 - BeeHumidityPreference, 13
 - BeeLifeSpan, 13
 - BeeProductionSpeed, 14
 - BeeSpecies, 14
 - BeeTempPreference, 15
 - BeeType, 16
 - Direction, 16
 - HoneyCombType, 16
- BeeGame::Core::Events
 - beeCraftedEvent, 111
 - CallBeeCraftedEvent, 110
 - CallShapedRecipieCraftedEvent, 110
 - CallShaplessRecipirCraftedEvent, 111
 - ItemCraftedEvent, 111
 - shapedRecipieCrafted, 111
 - shaplessRecipieCrafted, 111
- BeeGame::Core::Extensions
 - CloneObject< T >, 112
 - ColourSprite, 113
 - SpawnItem, 114
- BeeGame::Core::THInput
 - blockInventoryJustClosed, 244
 - chestOpen, 244
 - GetAxis, 241
 - GetButton, 242
 - GetButtonDown, 243
 - GetButtonUp, 243
 - inputButtons, 244
 - isAnotherInventoryOpen, 245
- BeeGame::Core::THVector2
 - Equals, 248
 - GetHashCode, 249
 - operator THVector2, 249
 - operator Vector2, 249
 - operator!=, 249
 - operator*, 250
 - operator+, 250, 251
 - operator-, 251, 252
 - operator/, 252
 - operator==, 253
 - THVector2, 247, 248
 - ToString, 253
 - x, 253
 - y, 253
- BeeGame::Core::THVector3
 - Distance, 257
 - Equals, 257
 - GetHashCode, 258
 - operator Quaternion, 258
 - operator THVector3, 259
 - operator Vector3, 259
 - operator!=, 259
 - operator*, 260, 261
 - operator+, 261, 262
 - operator-, 263, 264
 - operator/, 264, 265
 - operator==, 266
 - THVector3, 255, 256
 - ToString, 266
 - x, 267
 - y, 267
 - z, 267
- BeeGame::Core::UnityTypeReplacements::THQuaternion
 - w, 246
 - x, 246
 - y, 246
 - z, 246

- BeeGame::Exceptionps::CraftingRecipieAddition←
Exception
 - CraftingRecipieAdditionException, [88](#), [89](#)
- BeeGame::Exceptionps::InputException
 - InputException, [123](#)
- BeeGame::Inventory::ApiaryInventory
 - beesCombineing, [40](#)
 - CheckforBees, [38](#)
 - combinationTime, [40](#)
 - FixedUpdate, [39](#)
 - SetChestInventory, [39](#)
 - timerSlideer, [41](#)
 - Update, [40](#)
- BeeGame::Inventory::BlockInventory::CraftingTable←
Inventory
 - AddItemToSlots, [102](#)
 - CheckShapedRecipie, [103](#)
 - CheckShapelessRecipie, [103](#)
 - CraftedItemRemoved, [103](#)
 - DropltemsFromInventory, [104](#)
 - ItemRemovedFromResult, [104](#)
 - OnDestroy, [104](#)
 - result, [107](#)
 - SaveInv, [105](#)
 - SetChestInventory, [105](#)
 - Start, [105](#)
 - ToggleInventory, [106](#)
 - Update, [106](#)
- BeeGame::Inventory::ChestInventory
 - ApplyPlayerItems, [73](#)
 - inventory, [75](#)
 - inventoryPosition, [75](#)
 - inventorySize, [75](#)
 - playerinventory, [76](#)
 - SetChestInventory, [73](#)
 - SetPlayerItems, [73](#)
 - ToggleInventory, [74](#)
 - Update, [74](#)
 - UpdateChestInventory, [75](#)
- BeeGame::Inventory::Inventory
 - AddItemToInventory, [125](#)
 - AddItemToSlots, [126](#)
 - DrawItemAtCursor, [126](#)
 - floatingItem, [129](#)
 - GetAllItems, [127](#)
 - inventoryName, [130](#)
 - InventorySet, [127](#)
 - items, [130](#)
 - myblock, [130](#)
 - PutItemsInSlots, [127](#)
 - SaveInv, [128](#)
 - SetAllItems, [128](#)
 - SetInventorySize, [129](#)
 - slots, [130](#)
 - spriteAtCursor, [130](#)
 - thisInventoryOpen, [131](#)
 - ToggleInventory, [129](#)
 - UpdateBase, [129](#)
- BeeGame::Inventory::InventorySlot
 - AddToFloatingItem, [132](#)
 - AddToSlot, [133](#)
 - CheckFloatingItem, [133](#)
 - CheckItem, [134](#)
 - item, [138](#)
 - itemText, [139](#)
 - itemsCanBeInserted, [138](#)
 - myInventory, [139](#)
 - OnDisable, [134](#)
 - OnPointerClick, [134](#)
 - OnPointerEnter, [136](#)
 - OnPointerExit, [136](#)
 - selectedSlot, [139](#)
 - slotIndex, [139](#)
 - SplitStack, [137](#)
 - SwapItems, [137](#)
 - Update, [137](#)
 - UpdateIcon, [138](#)
- BeeGame::Inventory::ItemsInInventory
 - AddItem, [158](#), [159](#)
 - ItemsInInventory, [158](#)
 - itemsInInventory, [159](#)
- BeeGame::Inventory::Player_Inventory::PlayerInventory
 - Awake, [177](#)
 - GetItemFromHotBar, [177](#)
 - OpenPlayerInventory, [178](#)
 - PickupItem, [178](#)
 - playerInventory, [181](#)
 - RemoveItemFromInventory, [179](#)
 - SelectedSlot, [179](#)
 - SetPlayerInventory, [180](#)
 - Update, [180](#)
- BeeGame::Items::AbstractItem
 - GetHashCode, [21](#)
 - GetItemID, [21](#)
 - GetItemName, [21](#)
- BeeGame::Items::ApplyColour
 - colour, [42](#)
 - objects, [42](#)
 - Start, [42](#)
- BeeGame::Items::Bee
 - Bee, [47](#), [48](#)
 - beeType, [52](#)
 - canSeeBeeData, [51](#)
 - ConvertToQueen, [48](#), [49](#)
 - GetHashCode, [49](#)
 - GetItemID, [49](#)
 - GetItemSprite, [50](#)
 - ID, [51](#)
 - itemSprite, [52](#)
 - MakeBeeWithStats, [50](#)
 - maxStack, [52](#)
 - maxStackCount, [52](#)
 - normalBee, [52](#)
 - previousBeeType, [53](#)
 - queenBee, [53](#)
- BeeGame::Items::HoneyComb

- CombColour, [122](#)
- GetGameObject, [120](#)
- GetHashCode, [120](#)
- GetItemID, [121](#)
- GetItemSprite, [121](#)
- HoneyComb, [119](#), [120](#)
- ID, [122](#)
- itemSprite, [122](#)
- type, [122](#)
- BeeGame::Items::Item
 - Clone, [142](#)
 - count, [153](#)
 - Equals, [143](#)
 - FaceDataDown, [143](#)
 - FaceDataEast, [144](#)
 - FaceDataNorth, [145](#)
 - FaceDataSouth, [146](#)
 - FaceDataUp, [147](#)
 - FaceDataWest, [147](#)
 - FaceUVs, [148](#)
 - GetGameObject, [149](#)
 - GetHashCode, [149](#)
 - GetItemID, [149](#)
 - GetItemName, [150](#)
 - GetItemSprite, [150](#)
 - ID, [153](#)
 - InteractWithObject, [150](#)
 - Item, [142](#)
 - ItemMesh, [151](#)
 - itemName, [154](#)
 - itemStackCount, [154](#)
 - maxStackCount, [154](#)
 - operator!=, [151](#)
 - operator==, [152](#)
 - placeable, [154](#)
 - TexturePosition, [152](#)
 - tileSize, [154](#)
 - ToString, [153](#)
 - usesGameObject, [155](#)
- BeeGame::Items::ItemGameObject
 - go, [157](#)
 - item, [157](#)
 - MakeMesh, [156](#)
 - Start, [156](#)
 - Update, [156](#)
- BeeGame::Items::NormalBee
 - GetHashCode, [174](#)
 - pEffect, [174](#)
 - pFertility, [174](#)
 - pLifespan, [174](#)
 - pProdSpeed, [175](#)
 - pSpecies, [175](#)
 - sEffect, [175](#)
 - sFertility, [175](#)
 - sLifespan, [175](#)
 - sProdSpeed, [176](#)
 - sSpecies, [176](#)
- BeeGame::Items::QueenBee
 - drone, [191](#)
 - GetHashCode, [191](#)
 - queen, [191](#)
 - QueenBee, [190](#)
- BeeGame::Items::Tile
 - x, [268](#)
 - y, [268](#)
- BeeGame::LoadResources
 - Awake, [169](#)
- BeeGame::Player::PlayerLook
 - cameraTransform, [183](#)
 - Look, [182](#)
 - myTransform, [183](#)
 - rotationLock, [183](#)
 - speed, [183](#)
 - Start, [182](#)
 - Update, [182](#)
 - xRot, [183](#)
 - yRot, [184](#)
- BeeGame::Player::PlayerMove
 - Awake, [185](#)
 - canJump, [187](#)
 - FixedUpdate, [185](#)
 - gravity, [187](#)
 - jumpHeight, [187](#)
 - maxVelocity, [188](#)
 - MovePlayer, [186](#)
 - myRigidBody, [188](#)
 - OnCollisionStay, [186](#)
 - speed, [188](#)
 - VerticalJumpSpeed, [187](#)
- BeeGame::Player::SavePlayerPosition
 - counter, [200](#)
 - Update, [199](#)
- BeeGame::Player::Selector
 - Awake, [201](#)
 - BreakBlock, [201](#)
 - FixedUpdate, [202](#)
 - hit, [204](#)
 - layers, [204](#)
 - PlaceBlock, [202](#)
 - playerInventory, [204](#)
 - selectedHotbarSlot, [204](#)
 - SelectedSlot, [202](#)
 - selector, [204](#)
 - Update, [203](#)
 - UpdateSelector, [203](#)
- BeeGame::Quest::QuestBook
 - GetHashCode, [192](#)
 - GetItemSprite, [193](#)
 - InteractWithObject, [193](#)
 - maxStackCount, [194](#)
 - QuestBook, [192](#)
- BeeGame::Resources::Resources
 - Culture, [196](#)
 - GetPrefabs, [195](#)
 - GetSprites, [195](#)
 - Prefabs, [197](#)

- resourceCulture, 196
- resourceMan, 196
- ResourceManager, 197
- Resources, 195
- Sprites, 197
- BeeGame::Serialization::Serialization
 - DeSerializeInventory, 206
 - DeleteFile, 206
 - FileName, 207
 - LoadChunk, 207
 - LoadFile, 208
 - LoadPlayerPosition, 208
 - MakeDirectorys, 209
 - SaveChunk, 209
 - SaveFile, 210
 - saveFolderName, 211
 - savePath, 211
 - SavePlayerPosition, 210
 - SerializeInventory, 211
 - worldName, 212
- BeeGame::SpawnItem
 - OnDrawGizmos, 219
 - Start, 219
- BeeGame::Terrain::ChunkWorldPos
 - ChunkWorldPos, 85
 - Equals, 85
 - GetHashCode, 86
 - operator ChunkWorldPos, 86
 - operator THVector3, 87
 - ToString, 87
 - x, 87
 - y, 87
 - z, 88
- BeeGame::Terrain::Chunks::Chunk
 - applyCollisionMesh, 82
 - blocks, 82
 - chunkSize, 82
 - chunkWorldPos, 82
 - ColliderMesh, 78
 - filter, 83
 - GetBlock, 78
 - InRange, 79
 - mesh, 83
 - meshCollider, 83
 - RenderMesh, 79
 - rendered, 83
 - SetBlock, 80
 - SetBlocksUnmodified, 80
 - Start, 81
 - Update, 81
 - update, 83
 - UpdateChunk, 81
 - updateCollisionMesh, 84
 - world, 84
- BeeGame::Terrain::Chunks::LoadChunks
 - ApplyCollisionMeshToNearbyChunks, 164
 - BuildChunk, 164
 - buildList, 167
 - chunkPositions, 167
 - DeleteChunks, 165
 - FindChunksToLoad, 165
 - LoadAndRenderChunks, 166
 - nearbyChunks, 167
 - Start, 166
 - timer, 168
 - Update, 167
 - world, 168
- BeeGame::Terrain::Chunks::MeshData
 - AddQuadTriangles, 170
 - AddTriangle, 171
 - AddVertices, 171
 - colTris, 172
 - colVerts, 172
 - done, 172
 - shareMeshes, 172
 - tris, 172
 - uv, 172
 - verts, 173
- BeeGame::Terrain::Chunks::SaveChunk
 - blocks, 198
 - SaveChunk, 198
- BeeGame::Terrain::LandGeneration::Noise::Simplex↔
 - Noise
 - FastFloor, 213
 - Generate, 213–215
 - grad, 216, 217
 - Mod, 217
 - perm, 218
- BeeGame::Terrain::LandGeneration::Terrain
 - BlockInPosition, 223
 - GetBlock, 223, 224
 - GetBlockPos, 224, 225
 - GetBlockPosFromRayCast, 226
 - GetChunk, 226
 - Round, 226
 - RoundXZ, 227
 - RoundY, 228
 - SetBlock, 229
 - world, 230
- BeeGame::Terrain::LandGeneration::TerrainGeneration
 - caveFrequency, 237
 - caveSize, 237
 - ChunkGen, 232
 - ChunkGenThread, 233
 - CreateTree, 233
 - dirtBaseHeight, 237
 - dirtNoise, 237
 - dirtNoiseHeight, 237
 - GenChunkColumn, 234
 - GetNoise, 235
 - SetBlock, 236
 - stoneBaseHeight, 237
 - stoneBaseNoise, 238
 - stoneBaseNoiseHeight, 238
 - stoneMinHeight, 238
 - stoneMountainFrequency, 238

- stoneMountainHeight, [238](#)
- treeDensity, [239](#)
- treeFrequency, [239](#)
- BeeGame::Terrain::LandGeneration::World
 - chunkHasMadeCollisionMesh, [277](#)
 - chunkPrefab, [277](#)
 - chunks, [277](#)
 - CreateChunk, [272](#)
 - DestroyChunk, [273](#)
 - GetBlock, [274](#)
 - GetChunk, [274](#)
 - SetBlock, [275](#)
 - UpdateIfEqual, [276](#)
- BeeGame::Test
 - Print, [240](#)
 - Start, [240](#)
- BeeHumidityPreference
 - BeeGame::Core::Enums, [13](#)
- BeeLifeSpan
 - BeeGame::Core::Enums, [13](#)
- beeProduce
 - BeeGame::Core::Dictionarys::BeeDictionarys, [59](#)
- BeeProductionSpeed
 - BeeGame::Core::Enums, [14](#)
- BeeSpecies
 - BeeGame::Core::Enums, [14](#)
- BeeTempPreference
 - BeeGame::Core::Enums, [15](#)
- BeeType
 - BeeGame::Core::Enums, [16](#)
- beeType
 - BeeGame::Items::Bee, [52](#)
- beesCombineing
 - BeeGame::Inventory::ApiaryInventory, [40](#)
- Block
 - BeeGame::Blocks::Block, [61](#)
- BlockData
 - BeeGame::Blocks::Air, [23](#)
 - BeeGame::Blocks::Apiary, [27](#)
 - BeeGame::Blocks::Block, [62](#)
 - BeeGame::Blocks::Chest, [68](#)
 - BeeGame::Blocks::CraftingTable, [96](#)
- BlockInPosition
 - BeeGame::Terrain::LandGeneration::Terrain, [223](#)
- blockInventoryJustClosed
 - BeeGame::Core::THInput, [244](#)
- blocks
 - BeeGame::Terrain::Chunks::Chunk, [82](#)
 - BeeGame::Terrain::Chunks::SaveChunk, [198](#)
- BreakBlock
 - BeeGame::Blocks::Air, [23](#)
 - BeeGame::Blocks::Apiary, [28](#)
 - BeeGame::Blocks::Bedrock, [44](#)
 - BeeGame::Blocks::Block, [63](#)
 - BeeGame::Blocks::Chest, [69](#)
 - BeeGame::Blocks::CraftingTable, [97](#)
 - BeeGame::Player::Selector, [201](#)
- breakable
 - BeeGame::Blocks::Block, [66](#)
- BuildChunk
 - BeeGame::Terrain::Chunks::LoadChunks, [164](#)
- buildList
 - BeeGame::Terrain::Chunks::LoadChunks, [167](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 - Code/BeeGame/BeeGame/Blocks/Air.cs, [277](#), [278](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 - Code/BeeGame/BeeGame/Blocks/Apiary.cs, [278](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 - Code/BeeGame/BeeGame/Blocks/Bedrock.↵cs, [282](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 - Code/BeeGame/BeeGame/Blocks/Block.cs, [283](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 - Code/BeeGame/BeeGame/Blocks/Chest.cs, [284](#), [285](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 - Code/BeeGame/BeeGame/Blocks/Crafting↵Table.cs, [286](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 - Code/BeeGame/BeeGame/Blocks/Dirt.cs, [287](#), [288](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 - Code/BeeGame/BeeGame/Blocks/Grass.cs, [288](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 - Code/BeeGame/BeeGame/Blocks/Leaves.cs, [289](#), [290](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 - Code/BeeGame/BeeGame/Blocks/Wood.cs, [290](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 - Code/BeeGame/BeeGame/Core/Dictionarys/↵BeeDictionarys.cs, [291](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 - Code/BeeGame/BeeGame/Core/Dictionarys/↵CraftingRecipies.cs, [293](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 - Code/BeeGame/BeeGame/Core/Dictionarys/↵EqualityComperors.cs, [295](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 - Code/BeeGame/BeeGame/Core/Dictionarys/↵PrefabDictionary.cs, [296](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 - Code/BeeGame/BeeGame/Core/Dictionarys/↵SpriteDictionary.cs, [296](#), [297](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 - Code/BeeGame/BeeGame/Core/Enums/↵Enums.cs, [297](#), [298](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 - Code/BeeGame/BeeGame/Core/Events.cs, [299](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 - Code/BeeGame/BeeGame/Core/Extensions.↵

- cs, [300](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Core/UnityType↵
Replacements/THInput.cs, [302](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Core/UnityType↵
Replacements/THQuaternion.cs, [304](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Core/UnityType↵
Replacements/THVector2.cs, [304](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Core/UnityType↵
Replacements/THVector3.cs, [306](#), [307](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Exceptipns/↵
CraftingRecipieAdditionException.cs, [309](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Exceptipns/↵
InputException.cs, [309](#), [310](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Inventory/Block↵
Inventory/ApiaryInventory.cs, [310](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Inventory/Block↵
Inventory/ChestInventory.cs, [312](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Inventory/Block↵
Inventory/CraftingTableInventory.cs, [313](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Inventory/↵
Inventory.cs, [315](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Inventory/↵
InventorySlot.cs, [317](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Inventory/Items↵
InInventory.cs, [320](#), [321](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Inventory/Player
Inventory/PlayerInventory.cs, [321](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Items/Abstract↵
Item.cs, [323](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Items/Apply↵
Colour.cs, [324](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Items/Bee.cs,
[324](#), [325](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Items/Honey↵
Comb.cs, [327](#), [328](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Items/Item.cs,
[328](#), [329](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Items/Item↵
GameObject.cs, [332](#), [333](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/LoadResources.↵
cs, [333](#), [334](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Player/Player↵
Look.cs, [334](#), [335](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Player/Player↵
Move.cs, [335](#), [336](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Player/Save↵
PlayerPosition.cs, [337](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Player/Selector.↵
cs, [337](#), [338](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Properties/↵
AssemblyInfo.cs, [339](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Quest/Quest↵
Book.cs, [340](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Resources/↵
Resources.Designer.cs, [340](#), [341](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Serialization/↵
Serialization.cs, [342](#), [343](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/SpawnITem.cs,
[345](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Terrain/Chunk↵
WorldPos.cs, [354](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Terrain/Chunks/↵
Chunk.cs, [346](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Terrain/Chunks/↵
LoadChunks.cs, [349](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Terrain/Chunks/↵
MeshData.cs, [352](#), [353](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Terrain/Chunks/↵
SaveChunk.cs, [353](#), [354](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Terrain/Land↵
Generation/Noise/SimplexNoise.cs, [355](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Terrain/Land↵
Generation/Terrain.cs, [359](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Terrain/Land↵
Generation/TerrainGeneration.cs, [362](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/Terrain/Land↵
Generation/World.cs, [364](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵

- Code/BeeGame/BeeGame/obj/Debug/↵
TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs, [334](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/obj/Debug/↵
TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs, [334](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/obj/Debug/↵
TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs, [334](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↵
Code/BeeGame/BeeGame/test.cs, [367](#)
- CallBeeCraftedEvent
 - BeeGame::Core::Events, [110](#)
- CallShapedRecipieCraftedEvent
 - BeeGame::Core::Events, [110](#)
- CallShaplessRecipirCraftedEvent
 - BeeGame::Core::Events, [111](#)
- cameraTransform
 - BeeGame::Player::PlayerLook, [183](#)
- canJump
 - BeeGame::Player::PlayerMove, [187](#)
- canSeeBeeData
 - BeeGame::Items::Bee, [51](#)
- caveFrequency
 - BeeGame::Terrain::LandGeneration::Terrain↵
Generation, [237](#)
- caveSize
 - BeeGame::Terrain::LandGeneration::Terrain↵
Generation, [237](#)
- changed
 - BeeGame::Blocks::Block, [66](#)
- CheckFloatingItem
 - BeeGame::Inventory::InventorySlot, [133](#)
- CheckItem
 - BeeGame::Inventory::InventorySlot, [134](#)
- CheckShapedRecipie
 - BeeGame::Inventory::BlockInventory::Crafting↵
TableInventory, [103](#)
- CheckShapelessRecipie
 - BeeGame::Inventory::BlockInventory::Crafting↵
TableInventory, [103](#)
- CheckforBees
 - BeeGame::Inventory::ApiaryInventory, [38](#)
- Chest
 - BeeGame::Blocks::Chest, [68](#)
- chestOpen
 - BeeGame::Core::THInput, [244](#)
- ChunkGen
 - BeeGame::Terrain::LandGeneration::Terrain↵
Generation, [232](#)
- ChunkGenThread
 - BeeGame::Terrain::LandGeneration::Terrain↵
Generation, [233](#)
- chunkHasMadeCollisionMesh
 - BeeGame::Terrain::LandGeneration::World, [277](#)
- chunkPositions
 - BeeGame::Terrain::Chunks::LoadChunks, [167](#)
- chunkPrefab
 - BeeGame::Terrain::LandGeneration::World, [277](#)
- chunkSize
 - BeeGame::Terrain::Chunks::Chunk, [82](#)
- ChunkWorldPos
 - BeeGame::Terrain::ChunkWorldPos, [85](#)
- chunkWorldPos
 - BeeGame::Terrain::Chunks::Chunk, [82](#)
- chunks
 - BeeGame::Terrain::LandGeneration::World, [277](#)
- Clone
 - BeeGame::Items::Item, [142](#)
- CloneObject< T >
 - BeeGame::Core::Extensions, [112](#)
- colTris
 - BeeGame::Terrain::Chunks::MeshData, [172](#)
- colVerts
 - BeeGame::Terrain::Chunks::MeshData, [172](#)
- ColliderMesh
 - BeeGame::Terrain::Chunks::Chunk, [78](#)
- colour
 - BeeGame::Items::ApplyColour, [42](#)
- ColourSprite
 - BeeGame::Core::Extensions, [113](#)
- CombColour
 - BeeGame::Core::Dictionarys::BeeDictionarys, [55](#)
 - BeeGame::Items::HoneyComb, [122](#)
- combinationTime
 - BeeGame::Inventory::ApiaryInventory, [40](#)
- CombineEffect
 - BeeGame::Blocks::Apiary, [28](#)
- CombineFertility
 - BeeGame::Blocks::Apiary, [29](#)
- CombineLifespan
 - BeeGame::Blocks::Apiary, [29](#)
- CombineProductionSpeed
 - BeeGame::Blocks::Apiary, [30](#)
- CombineSpecies
 - BeeGame::Blocks::Apiary, [30](#)
- ConvertToQueen
 - BeeGame::Items::Bee, [48](#), [49](#)
- count
 - BeeGame::Items::Item, [153](#)
- counter
 - BeeGame::Player::SavePlayerPosition, [200](#)
- CraftedItemRemoved
 - BeeGame::Inventory::BlockInventory::Crafting↵
TableInventory, [103](#)
- CraftingRecipieAdditionException
 - BeeGame::Exceptipns::CraftingRecipieAddition↵
Exception, [88](#), [89](#)
- CraftingTable
 - BeeGame::Blocks::CraftingTable, [96](#)
- CreateChunk
 - BeeGame::Terrain::LandGeneration::World, [272](#)
- CreateTree

- BeeGame::Terrain::LandGeneration::Terrain↔
Generation, [233](#)
- Culture
 - BeeGame::Resources::Resources, [196](#)
- DeSerializeInventory
 - BeeGame::Serialization::Serialization, [206](#)
- DeleteChunks
 - BeeGame::Terrain::Chunks::LoadChunks, [165](#)
- DeleteFile
 - BeeGame::Serialization::Serialization, [206](#)
- DestroyChunk
 - BeeGame::Terrain::LandGeneration::World, [273](#)
- Direction
 - BeeGame::Core::Enums, [16](#)
- Dirt
 - BeeGame::Blocks::Dirt, [108](#)
- dirtBaseHeight
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, [237](#)
- dirtNoise
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, [237](#)
- dirtNoiseHeight
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, [237](#)
- Distance
 - BeeGame::Core::THVector3, [257](#)
- done
 - BeeGame::Terrain::Chunks::MeshData, [172](#)
- DrawItemAtCursor
 - BeeGame::Inventory::Inventory, [126](#)
- drone
 - BeeGame::Items::QueenBee, [191](#)
- DropItemsFromInventory
 - BeeGame::Inventory::BlockInventory::Crafting↔
TableInventory, [104](#)
- Equals
 - BeeGame::Core::Dictionaries::BeeCombination↔
DictionaryEqualityComparer, [54](#)
 - BeeGame::Core::THVector2, [248](#)
 - BeeGame::Core::THVector3, [257](#)
 - BeeGame::Items::Item, [143](#)
 - BeeGame::Terrain::ChunkWorldPos, [85](#)
- FaceDataDown
 - BeeGame::Items::Item, [143](#)
- FaceDataEast
 - BeeGame::Items::Item, [144](#)
- FaceDataNorth
 - BeeGame::Items::Item, [145](#)
- FaceDataSouth
 - BeeGame::Items::Item, [146](#)
- FaceDataUp
 - BeeGame::Items::Item, [147](#)
- FaceDataWest
 - BeeGame::Items::Item, [147](#)
- FaceUVs
 - BeeGame::Items::Item, [148](#)
- FastFloor
 - BeeGame::Terrain::LandGeneration::Noise::↔
SimplexNoise, [213](#)
- FileName
 - BeeGame::Serialization::Serialization, [207](#)
- filter
 - BeeGame::Terrain::Chunks::Chunk, [83](#)
- FindChunksToLoad
 - BeeGame::Terrain::Chunks::LoadChunks, [165](#)
- FixedUpdate
 - BeeGame::Inventory::ApiaryInventory, [39](#)
 - BeeGame::Player::PlayerMove, [185](#)
 - BeeGame::Player::Selector, [202](#)
- floatingItem
 - BeeGame::Inventory::Inventory, [129](#)
- GenChunkColumn
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, [234](#)
- Generate
 - BeeGame::Terrain::LandGeneration::Noise::↔
SimplexNoise, [213–215](#)
- GetAllItems
 - BeeGame::Inventory::Inventory, [127](#)
- GetAxis
 - BeeGame::Core::THInput, [241](#)
- GetBeeColour
 - BeeGame::Core::Dictionaries::BeeDictionaries, [56](#)
- GetBeeProduce
 - BeeGame::Core::Dictionaries::BeeDictionaries, [56](#)
- GetBlock
 - BeeGame::Terrain::Chunks::Chunk, [78](#)
 - BeeGame::Terrain::LandGeneration::Terrain, [223](#),
[224](#)
 - BeeGame::Terrain::LandGeneration::World, [274](#)
- GetBlockPos
 - BeeGame::Terrain::LandGeneration::Terrain, [224](#),
[225](#)
- GetBlockPosFromRayCast
 - BeeGame::Terrain::LandGeneration::Terrain, [226](#)
- GetButton
 - BeeGame::Core::THInput, [242](#)
- GetButtonDown
 - BeeGame::Core::THInput, [243](#)
- GetButtonUp
 - BeeGame::Core::THInput, [243](#)
- GetChunk
 - BeeGame::Terrain::LandGeneration::Terrain, [226](#)
 - BeeGame::Terrain::LandGeneration::World, [274](#)
- GetCombColour
 - BeeGame::Core::Dictionaries::BeeDictionaries, [56](#)
- GetCombinations
 - BeeGame::Core::Dictionaries::BeeDictionaries, [57](#)
- GetGameObject
 - BeeGame::Blocks::Apiary, [31](#)
 - BeeGame::Blocks::Chest, [69](#)
 - BeeGame::Blocks::CraftingTable, [97](#)
 - BeeGame::Items::HoneyComb, [120](#)

- BeeGame::Items::Item, 149
- GetHashCode
 - BeeGame::Blocks::Air, 24
 - BeeGame::Blocks::Apiary, 31
 - BeeGame::Blocks::Bedrock, 44
 - BeeGame::Blocks::Block, 63
 - BeeGame::Blocks::Chest, 69
 - BeeGame::Blocks::CraftingTable, 97
 - BeeGame::Blocks::Dirt, 108
 - BeeGame::Blocks::Grass, 116
 - BeeGame::Blocks::Leaves, 161
 - BeeGame::Blocks::Wood, 269
 - BeeGame::Core::Dictionarys::BeeCombination↔
DictionaryEqualityComparer, 54
 - BeeGame::Core::THVector2, 249
 - BeeGame::Core::THVector3, 258
 - BeeGame::Items::AbstractItem, 21
 - BeeGame::Items::Bee, 49
 - BeeGame::Items::HoneyComb, 120
 - BeeGame::Items::Item, 149
 - BeeGame::Items::NormalBee, 174
 - BeeGame::Items::QueenBee, 191
 - BeeGame::Quest::QuestBook, 192
 - BeeGame::Terrain::ChunkWorldPos, 86
- GetItemFromHotBar
 - BeeGame::Inventory::Player_Inventory::Player↔
Inventory, 177
- GetItemID
 - BeeGame::Items::AbstractItem, 21
 - BeeGame::Items::Bee, 49
 - BeeGame::Items::HoneyComb, 121
 - BeeGame::Items::Item, 149
- GetItemName
 - BeeGame::Items::AbstractItem, 21
 - BeeGame::Items::Item, 150
- GetItemSprite
 - BeeGame::Blocks::Block, 64
 - BeeGame::Blocks::CraftingTable, 98
 - BeeGame::Blocks::Dirt, 108
 - BeeGame::Blocks::Grass, 116
 - BeeGame::Blocks::Leaves, 161
 - BeeGame::Blocks::Wood, 270
 - BeeGame::Items::Bee, 50
 - BeeGame::Items::HoneyComb, 121
 - BeeGame::Items::Item, 150
 - BeeGame::Quest::QuestBook, 193
- GetNoise
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, 235
- GetPrefab
 - BeeGame::Core::Dictionarys::PrefabDictionary, 189
- GetPrefabs
 - BeeGame::Resources::Resources, 195
- GetShapedRecipeItem
 - BeeGame::Core::Dictionarys::CraftingRecipes, 91
- GetShaplessRecipeResult
 - BeeGame::Core::Dictionarys::CraftingRecipes, 92, 93
- GetShaplessRecipeString
 - BeeGame::Core::Dictionarys::CraftingRecipes, 93
- GetSprite
 - BeeGame::Core::Dictionarys::SpriteDictionary, 221
- GetSprites
 - BeeGame::Resources::Resources, 195
- GetWeights
 - BeeGame::Core::Dictionarys::BeeDictionarys, 57
- go
 - BeeGame::Items::ItemGameObject, 157
- grad
 - BeeGame::Terrain::LandGeneration::Noise::↔
SimplexNoise, 216, 217
- Grass
 - BeeGame::Blocks::Grass, 115
- gravity
 - BeeGame::Player::PlayerMove, 187
- hit
 - BeeGame::Player::Selector, 204
- HoneyComb
 - BeeGame::Items::HoneyComb, 119, 120
- HoneyCombType
 - BeeGame::Core::Enums, 16
- honeyCoubColour
 - BeeGame::Core::Dictionarys::BeeDictionarys, 59
- ID
 - BeeGame::Blocks::Air, 25
 - BeeGame::Blocks::Apiary, 37
 - BeeGame::Blocks::Bedrock, 45
 - BeeGame::Blocks::Block, 66
 - BeeGame::Blocks::Chest, 71
 - BeeGame::Blocks::CraftingTable, 101
 - BeeGame::Blocks::Dirt, 110
 - BeeGame::Blocks::Grass, 118
 - BeeGame::Blocks::Leaves, 163
 - BeeGame::Blocks::Wood, 271
 - BeeGame::Items::Bee, 51
 - BeeGame::Items::HoneyComb, 122
 - BeeGame::Items::Item, 153
- InRange
 - BeeGame::Terrain::Chunks::Chunk, 79
- inputButtons
 - BeeGame::Core::THInput, 244
- InputException
 - BeeGame::Exceptipns::InputException, 123
- InteractWithBlock
 - BeeGame::Blocks::Apiary, 32
 - BeeGame::Blocks::Block, 64
 - BeeGame::Blocks::Chest, 70
 - BeeGame::Blocks::CraftingTable, 98
- InteractWithObject
 - BeeGame::Items::Item, 150
 - BeeGame::Quest::QuestBook, 193
- inventory

- BeeGame::Inventory::ChestInventory, 75
- inventoryName
 - BeeGame::Inventory::Inventory, 130
- inventoryPosition
 - BeeGame::Inventory::ChestInventory, 75
- InventorySet
 - BeeGame::Inventory::Inventory, 127
- inventorySize
 - BeeGame::Inventory::ChestInventory, 75
- isAnotherInventoryOpen
 - BeeGame::Core::THInput, 245
- IsSolid
 - BeeGame::Blocks::Air, 24
 - BeeGame::Blocks::Block, 64
 - BeeGame::Blocks::Leaves, 161
- Item
 - BeeGame::Items::Item, 142
- item
 - BeeGame::Inventory::InventorySlot, 138
 - BeeGame::Items::ItemGameObject, 157
- ItemCraftedEvent
 - BeeGame::Core::Events, 111
- ItemMesh
 - BeeGame::Items::Item, 151
- itemName
 - BeeGame::Items::Item, 154
- ItemRemovedFromResult
 - BeeGame::Inventory::BlockInventory::Crafting←
TableInventory, 104
- itemSprite
 - BeeGame::Items::Bee, 52
 - BeeGame::Items::HoneyComb, 122
- itemSpriteDictionary
 - BeeGame::Core::Dictionaries::SpriteDictionary, 222
- itemStackCount
 - BeeGame::Items::Item, 154
- itemText
 - BeeGame::Inventory::InventorySlot, 139
- items
 - BeeGame::Inventory::Inventory, 130
- itemsCanBeInserted
 - BeeGame::Inventory::InventorySlot, 138
- ItemsInInventory
 - BeeGame::Inventory::ItemsInInventory, 158
- itemsInInventory
 - BeeGame::Inventory::ItemsInInventory, 159
- jumpHeight
 - BeeGame::Player::PlayerMove, 187
- layers
 - BeeGame::Player::Selector, 204
- Leaves
 - BeeGame::Blocks::Leaves, 160
- LoadAndRenderChunks
 - BeeGame::Terrain::Chunks::LoadChunks, 166
- LoadChunk
 - BeeGame::Serialization::Serialization, 207
- LoadFile
 - BeeGame::Serialization::Serialization, 208
- LoadPlayerPosition
 - BeeGame::Serialization::Serialization, 208
- LoadPrefabs
 - BeeGame::Core::Dictionaries::PrefabDictionary, 189
- LoadSprites
 - BeeGame::Core::Dictionaries::SpriteDictionary, 221
- Look
 - BeeGame::Player::PlayerLook, 182
- MakeBee
 - BeeGame::Blocks::Apiary, 32
- MakeBeeWithStats
 - BeeGame::Items::Bee, 50
- MakeBees
 - BeeGame::Blocks::Apiary, 33
- MakeDirectories
 - BeeGame::Serialization::Serialization, 209
- MakeMesh
 - BeeGame::Items::ItemGameObject, 156
- maxStack
 - BeeGame::Items::Bee, 52
- maxStackCount
 - BeeGame::Items::Bee, 52
 - BeeGame::Items::Item, 154
 - BeeGame::Quest::QuestBook, 194
- maxVelocity
 - BeeGame::Player::PlayerMove, 188
- mesh
 - BeeGame::Terrain::Chunks::Chunk, 83
- meshCollider
 - BeeGame::Terrain::Chunks::Chunk, 83
- Mod
 - BeeGame::Terrain::LandGeneration::Noise::←
SimplexNoise, 217
- MovePlayer
 - BeeGame::Player::PlayerMove, 186
- mutationMultiplier
 - BeeGame::Blocks::Apiary, 37
- myGameObject
 - BeeGame::Blocks::Apiary, 37
 - BeeGame::Blocks::Chest, 71
 - BeeGame::Blocks::CraftingTable, 101
- myInventory
 - BeeGame::Inventory::InventorySlot, 139
- myRigidBody
 - BeeGame::Player::PlayerMove, 188
- myTransform
 - BeeGame::Player::PlayerLook, 183
- myblock
 - BeeGame::Inventory::Inventory, 130
- nearbyChunks
 - BeeGame::Terrain::Chunks::LoadChunks, 167
- normalBee
 - BeeGame::Items::Bee, 52

- objects
 - BeeGame::Items::ApplyColour, [42](#)
- OnCollisionStay
 - BeeGame::Player::PlayerMove, [186](#)
- OnDestroy
 - BeeGame::Inventory::BlockInventory::Crafting↔
TableInventory, [104](#)
- OnDisable
 - BeeGame::Inventory::InventorySlot, [134](#)
- OnDrawGizmos
 - BeeGame::SpawnItem, [219](#)
- OnPointerClick
 - BeeGame::Inventory::InventorySlot, [134](#)
- OnPointerEnter
 - BeeGame::Inventory::InventorySlot, [136](#)
- OnPointerExit
 - BeeGame::Inventory::InventorySlot, [136](#)
- OpenPlayerInventory
 - BeeGame::Inventory::Player_Inventory::Player↔
Inventory, [178](#)
- operator ChunkWorldPos
 - BeeGame::Terrain::ChunkWorldPos, [86](#)
- operator Quaternion
 - BeeGame::Core::THVector3, [258](#)
- operator THVector2
 - BeeGame::Core::THVector2, [249](#)
- operator THVector3
 - BeeGame::Core::THVector3, [259](#)
 - BeeGame::Terrain::ChunkWorldPos, [87](#)
- operator Vector2
 - BeeGame::Core::THVector2, [249](#)
- operator Vector3
 - BeeGame::Core::THVector3, [259](#)
- operator!=
 - BeeGame::Core::THVector2, [249](#)
 - BeeGame::Core::THVector3, [259](#)
 - BeeGame::Items::Item, [151](#)
- operator*
 - BeeGame::Core::THVector2, [250](#)
 - BeeGame::Core::THVector3, [260](#), [261](#)
- operator+
 - BeeGame::Core::THVector2, [250](#), [251](#)
 - BeeGame::Core::THVector3, [261](#), [262](#)
- operator-
 - BeeGame::Core::THVector2, [251](#), [252](#)
 - BeeGame::Core::THVector3, [263](#), [264](#)
- operator/
 - BeeGame::Core::THVector2, [252](#)
 - BeeGame::Core::THVector3, [264](#), [265](#)
- operator==
 - BeeGame::Core::THVector2, [253](#)
 - BeeGame::Core::THVector3, [266](#)
 - BeeGame::Items::Item, [152](#)
- pEffect
 - BeeGame::Items::NormalBee, [174](#)
- pFertility
 - BeeGame::Items::NormalBee, [174](#)
- pLifespan
 - BeeGame::Items::NormalBee, [174](#)
- pProdSpeed
 - BeeGame::Items::NormalBee, [175](#)
- pSpecies
 - BeeGame::Items::NormalBee, [175](#)
- perm
 - BeeGame::Terrain::LandGeneration::Noise::↔
SimplexNoise, [218](#)
- PickupItem
 - BeeGame::Inventory::Player_Inventory::Player↔
Inventory, [178](#)
- PlaceBlock
 - BeeGame::Player::Selector, [202](#)
- placeable
 - BeeGame::Blocks::Block, [66](#)
 - BeeGame::Items::Item, [154](#)
- playerInventory
 - BeeGame::Inventory::Player_Inventory::Player↔
Inventory, [181](#)
 - BeeGame::Player::Selector, [204](#)
- playerinventory
 - BeeGame::Inventory::ChestInventory, [76](#)
- prefabDictionary
 - BeeGame::Core::Dictionaries::PrefabDictionary,
[190](#)
- Prefabs
 - BeeGame::Resources::Resources, [197](#)
- previousBeeType
 - BeeGame::Items::Bee, [53](#)
- Print
 - BeeGame::Test, [240](#)
- PutItemsInSlots
 - BeeGame::Inventory::Inventory, [127](#)
- queen
 - BeeGame::Items::QueenBee, [191](#)
- QueenBee
 - BeeGame::Items::QueenBee, [190](#)
- queenBee
 - BeeGame::Items::Bee, [53](#)
- QuestBook
 - BeeGame::Quest::QuestBook, [192](#)
- Rand
 - BeeGame::Blocks::Apiary, [34](#)
- RemoveItemFromInventory
 - BeeGame::Inventory::Player_Inventory::Player↔
Inventory, [179](#)
- RenderMesh
 - BeeGame::Terrain::Chunks::Chunk, [79](#)
- rendered
 - BeeGame::Terrain::Chunks::Chunk, [83](#)
- resourceCulture
 - BeeGame::Resources::Resources, [196](#)
- resourceMan
 - BeeGame::Resources::Resources, [196](#)
- ResourceManager
 - BeeGame::Resources::Resources, [197](#)
- Resources

- BeeGame::Resources::Resources, 195
- result
 - BeeGame::Inventory::BlockInventory::Crafting↔
TableInventory, 107
- ReturnChange
 - BeeGame::Blocks::Apiary, 35
- ReturnShapedRecipeItem
 - BeeGame::Blocks::CraftingTable, 99
- ReturnShapelessRecipeItem
 - BeeGame::Blocks::CraftingTable, 100
- rotationLock
 - BeeGame::Player::PlayerLook, 183
- Round
 - BeeGame::Terrain::LandGeneration::Terrain, 226
- RoundXZ
 - BeeGame::Terrain::LandGeneration::Terrain, 227
- RoundY
 - BeeGame::Terrain::LandGeneration::Terrain, 228
- sEffect
 - BeeGame::Items::NormalBee, 175
- sFertility
 - BeeGame::Items::NormalBee, 175
- sLifespan
 - BeeGame::Items::NormalBee, 175
- sProdSpeed
 - BeeGame::Items::NormalBee, 176
- sSpecies
 - BeeGame::Items::NormalBee, 176
- SaveChunk
 - BeeGame::Serialization::Serialization, 209
 - BeeGame::Terrain::Chunks::SaveChunk, 198
- SaveFile
 - BeeGame::Serialization::Serialization, 210
- saveFolderName
 - BeeGame::Serialization::Serialization, 211
- SaveInv
 - BeeGame::Inventory::BlockInventory::Crafting↔
TableInventory, 105
 - BeeGame::Inventory::Inventory, 128
- savePath
 - BeeGame::Serialization::Serialization, 211
- SavePlayerPosition
 - BeeGame::Serialization::Serialization, 210
- selectedHotbarSlot
 - BeeGame::Player::Selector, 204
- SelectedSlot
 - BeeGame::Inventory::Player_Inventory::Player↔
Inventory, 179
 - BeeGame::Player::Selector, 202
- selectedSlot
 - BeeGame::Inventory::InventorySlot, 139
- selector
 - BeeGame::Player::Selector, 204
- SerializeInventory
 - BeeGame::Serialization::Serialization, 211
- SetAllItems
 - BeeGame::Inventory::Inventory, 128
- SetBlock
 - BeeGame::Terrain::Chunks::Chunk, 80
- BeeGame::Terrain::LandGeneration::Terrain, 229
- BeeGame::Terrain::LandGeneration::Terrain↔
Generation, 236
- BeeGame::Terrain::LandGeneration::World, 275
- SetBlocksUnmodified
 - BeeGame::Terrain::Chunks::Chunk, 80
- SetChestInventory
 - BeeGame::Inventory::ApiaryInventory, 39
 - BeeGame::Inventory::BlockInventory::Crafting↔
TableInventory, 105
 - BeeGame::Inventory::ChestInventory, 73
- SetInventorySize
 - BeeGame::Inventory::Inventory, 129
- SetPlayerInventory
 - BeeGame::Inventory::Player_Inventory::Player↔
Inventory, 180
- SetPlayerItems
 - BeeGame::Inventory::ChestInventory, 73
- shapedCraftingRecipes
 - BeeGame::Core::Dictionaries::CraftingRecipes, 94
- shapedRecipeCrafted
 - BeeGame::Core::Events, 111
- shaplessRecipeCrafted
 - BeeGame::Core::Events, 111
- shaplessRecipes
 - BeeGame::Core::Dictionaries::CraftingRecipes, 94
- shareMeshes
 - BeeGame::Terrain::Chunks::MeshData, 172
- slotIndex
 - BeeGame::Inventory::InventorySlot, 139
- slots
 - BeeGame::Inventory::Inventory, 130
- SpawnItem
 - BeeGame::Core::Extensions, 114
- speed
 - BeeGame::Player::PlayerLook, 183
 - BeeGame::Player::PlayerMove, 188
- SplitStack
 - BeeGame::Inventory::InventorySlot, 137
- spriteAtCursor
 - BeeGame::Inventory::Inventory, 130
- Sprites
 - BeeGame::Resources::Resources, 197
- Start
 - BeeGame::Inventory::BlockInventory::Crafting↔
TableInventory, 105
 - BeeGame::Items::ApplyColour, 42
 - BeeGame::Items::ItemGameObject, 156
 - BeeGame::Player::PlayerLook, 182
 - BeeGame::SpawnItem, 219
 - BeeGame::Terrain::Chunks::Chunk, 81
 - BeeGame::Terrain::Chunks::LoadChunks, 166
 - BeeGame::Test, 240
- stoneBaseHeight
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, 237
- stoneBaseNoise

- BeeGame::Terrain::LandGeneration::Terrain↔
Generation, [238](#)
- stoneBaseNoiseHeight
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, [238](#)
- stoneMinHeight
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, [238](#)
- stoneMountainFrequency
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, [238](#)
- stoneMountainHeight
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, [238](#)
- SwapItems
 - BeeGame::Inventory::InventorySlot, [137](#)
- THVector2
 - BeeGame::Core::THVector2, [247](#), [248](#)
- THVector3
 - BeeGame::Core::THVector3, [255](#), [256](#)
- TexturePosition
 - BeeGame::Blocks::Apiary, [36](#)
 - BeeGame::Blocks::Bedrock, [44](#)
 - BeeGame::Blocks::Chest, [70](#)
 - BeeGame::Blocks::CraftingTable, [100](#)
 - BeeGame::Blocks::Dirt, [109](#)
 - BeeGame::Blocks::Grass, [116](#)
 - BeeGame::Blocks::Leaves, [162](#)
 - BeeGame::Blocks::Wood, [270](#)
 - BeeGame::Items::Item, [152](#)
- thisInventoryOpen
 - BeeGame::Inventory::Inventory, [131](#)
- tileSize
 - BeeGame::Items::Item, [154](#)
- timer
 - BeeGame::Terrain::Chunks::LoadChunks, [168](#)
- timerSlideer
 - BeeGame::Inventory::ApiaryInventory, [41](#)
- ToString
 - BeeGame::Blocks::Air, [24](#)
 - BeeGame::Blocks::Apiary, [36](#)
 - BeeGame::Blocks::Bedrock, [45](#)
 - BeeGame::Blocks::Block, [65](#)
 - BeeGame::Blocks::Chest, [71](#)
 - BeeGame::Blocks::Dirt, [109](#)
 - BeeGame::Blocks::Grass, [117](#)
 - BeeGame::Blocks::Leaves, [162](#)
 - BeeGame::Blocks::Wood, [271](#)
 - BeeGame::Core::THVector2, [253](#)
 - BeeGame::Core::THVector3, [266](#)
 - BeeGame::Items::Item, [153](#)
 - BeeGame::Terrain::ChunkWorldPos, [87](#)
- ToggleInventory
 - BeeGame::Inventory::BlockInventory::Crafting↔
TableInventory, [106](#)
 - BeeGame::Inventory::ChestInventory, [74](#)
 - BeeGame::Inventory::Inventory, [129](#)
- treeDensity
- BeeGame::Terrain::LandGeneration::Terrain↔
Generation, [239](#)
- treeFrequency
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, [239](#)
- tris
 - BeeGame::Terrain::Chunks::MeshData, [172](#)
- type
 - BeeGame::Items::HoneyComb, [122](#)
- Update
 - BeeGame::Inventory::ApiaryInventory, [40](#)
 - BeeGame::Inventory::BlockInventory::Crafting↔
TableInventory, [106](#)
 - BeeGame::Inventory::ChestInventory, [74](#)
 - BeeGame::Inventory::InventorySlot, [137](#)
 - BeeGame::Inventory::Player_Inventory::Player↔
Inventory, [180](#)
 - BeeGame::Items::ItemGameObject, [156](#)
 - BeeGame::Player::PlayerLook, [182](#)
 - BeeGame::Player::SavePlayerPosition, [199](#)
 - BeeGame::Player::Selector, [203](#)
 - BeeGame::Terrain::Chunks::Chunk, [81](#)
 - BeeGame::Terrain::Chunks::LoadChunks, [167](#)
- update
 - BeeGame::Terrain::Chunks::Chunk, [83](#)
- UpdateBase
 - BeeGame::Inventory::Inventory, [129](#)
- UpdateBlock
 - BeeGame::Blocks::Block, [65](#)
 - BeeGame::Blocks::Grass, [117](#)
- UpdateChestInventory
 - BeeGame::Inventory::ChestInventory, [75](#)
- UpdateChunk
 - BeeGame::Terrain::Chunks::Chunk, [81](#)
- updateCollisionMesh
 - BeeGame::Terrain::Chunks::Chunk, [84](#)
- UpdateIcon
 - BeeGame::Inventory::InventorySlot, [138](#)
- UpdateIfEqual
 - BeeGame::Terrain::LandGeneration::World, [276](#)
- UpdateSelector
 - BeeGame::Player::Selector, [203](#)
- usesGameObject
 - BeeGame::Items::Item, [155](#)
- uv
 - BeeGame::Terrain::Chunks::MeshData, [172](#)
- VerticalJumpSpeed
 - BeeGame::Player::PlayerMove, [187](#)
- verts
 - BeeGame::Terrain::Chunks::MeshData, [173](#)
- w
 - BeeGame::Core::UnityTypeReplacements::TH↔
Quaternion, [246](#)
- Wood
 - BeeGame::Blocks::Wood, [269](#)
- world

- BeeGame::Terrain::Chunks::Chunk, [84](#)
- BeeGame::Terrain::Chunks::LoadChunks, [168](#)
- BeeGame::Terrain::LandGeneration::Terrain, [230](#)
- worldName
 - BeeGame::Serialization::Serialization, [212](#)
- x
 - BeeGame::Core::THVector2, [253](#)
 - BeeGame::Core::THVector3, [267](#)
 - BeeGame::Core::UnityTypeReplacements::TH↔
Quaternion, [246](#)
 - BeeGame::Items::Tile, [268](#)
 - BeeGame::Terrain::ChunkWorldPos, [87](#)
- xRot
 - BeeGame::Player::PlayerLook, [183](#)
- y
 - BeeGame::Core::THVector2, [253](#)
 - BeeGame::Core::THVector3, [267](#)
 - BeeGame::Core::UnityTypeReplacements::TH↔
Quaternion, [246](#)
 - BeeGame::Items::Tile, [268](#)
 - BeeGame::Terrain::ChunkWorldPos, [87](#)
- yRot
 - BeeGame::Player::PlayerLook, [184](#)
- z
 - BeeGame::Core::THVector3, [267](#)
 - BeeGame::Core::UnityTypeReplacements::TH↔
Quaternion, [246](#)
 - BeeGame::Terrain::ChunkWorldPos, [88](#)