

Block Game, Namespace, Class, and File Documentation

Block Game Version 0.1.2

Max Rose

Generated by Doxygen 1.8.13

Contents

I	Namespace Documentation	1
0.1	BeeGame Namespace Reference	1
0.2	BeeGame.Blocks Namespace Reference	2
0.3	BeeGame.Core Namespace Reference	2
0.4	BeeGame.Core.Enums Namespace Reference	3
0.4.1	Enumeration Type Documentation	3
0.5	BeeGame.Inventory Namespace Reference	8
0.6	BeeGame.Inventory.Player_Inventory Namespace Reference	9
0.7	BeeGame.Items Namespace Reference	9
0.8	BeeGame.Player Namespace Reference	9
0.9	BeeGame.Resources Namespace Reference	9
0.10	BeeGame.Serialization Namespace Reference	9
0.11	BeeGame.Terrain Namespace Reference	10
0.12	BeeGame.Terrain.Chunks Namespace Reference	10
0.13	BeeGame.Terrain.LandGeneration Namespace Reference	10
0.14	BeeGame.Terrain.LandGeneration.Noise Namespace Reference	11
II	Class Documentation	11
1	Items	11
1.1	BeeGame.Items.Item Class Reference	11
1.1.1	Detailed Description	13
1.1.2	Constructor & Destructor Documentation	13
1.1.3	Member Function Documentation	14
1.1.4	Member Data Documentation	24
1.2	BeeGame.Items.HoneyComb Class Reference	26
1.2.1	Detailed Description	26
1.2.2	Constructor & Destructor Documentation	27
1.2.3	Member Function Documentation	27
1.2.4	Member Data Documentation	29
1.2.5	Property Documentation	29
1.3	BeeGame.Items.ItemGameObject Class Reference	30
1.3.1	Detailed Description	30
1.3.2	Member Function Documentation	30
1.3.3	Member Data Documentation	31
1.4	BeeGame.Items.ApplyColour Class Reference	32
1.4.1	Detailed Description	32
1.4.2	Member Function Documentation	32
1.4.3	Member Data Documentation	33

2	Blocks	33
2.1	BeeGame.Blocks.Block Class Reference	33
2.1.1	Detailed Description	34
2.1.2	Constructor & Destructor Documentation	34
2.1.3	Member Function Documentation	35
2.1.4	Member Data Documentation	40
2.2	BeeGame.Items.Tile Struct Reference	40
2.2.1	Detailed Description	40
2.2.2	Member Data Documentation	41
2.3	BeeGame.Blocks.Air Class Reference	41
2.3.1	Detailed Description	42
2.3.2	Constructor & Destructor Documentation	42
2.3.3	Member Function Documentation	42
2.4	BeeGame.Blocks.Dirt Class Reference	44
2.4.1	Detailed Description	45
2.4.2	Constructor & Destructor Documentation	45
2.4.3	Member Function Documentation	45
2.5	BeeGame.Blocks.Grass Class Reference	47
2.5.1	Detailed Description	48
2.5.2	Constructor & Destructor Documentation	48
2.5.3	Member Function Documentation	48
2.6	BeeGame.Blocks.Chest Class Reference	51
2.6.1	Detailed Description	52
2.6.2	Constructor & Destructor Documentation	52
2.6.3	Member Function Documentation	52
2.6.4	Member Data Documentation	55
2.7	BeeGame.Blocks.Apiary Class Reference	55
2.7.1	Detailed Description	56
2.7.2	Constructor & Destructor Documentation	56
2.7.3	Member Function Documentation	56
2.8	BeeGame.Blocks.Wood Class Reference	57
2.8.1	Detailed Description	57
2.8.2	Constructor & Destructor Documentation	58
2.8.3	Member Function Documentation	58
2.9	BeeGame.Blocks.Leaves Class Reference	59
2.9.1	Detailed Description	60
2.9.2	Constructor & Destructor Documentation	60
2.9.3	Member Function Documentation	60
2.10	BeeGame.Blocks.Bedrock Class Reference	62
2.10.1	Detailed Description	63
2.10.2	Constructor & Destructor Documentation	63
2.10.3	Member Function Documentation	63

3	Inventory	65
3.1	BeeGame.Inventory.Inventory Class Reference	65
3.1.1	Detailed Description	66
3.1.2	Member Function Documentation	67
3.1.3	Member Data Documentation	70
3.2	BeeGame.Inventory.ItemsInInventory Class Reference	71
3.2.1	Detailed Description	71
3.2.2	Constructor & Destructor Documentation	71
3.2.3	Member Function Documentation	72
3.2.4	Member Data Documentation	73
3.3	BeeGame.Inventory.InventorySlot Class Reference	73
3.3.1	Detailed Description	74
3.3.2	Member Function Documentation	74
3.3.3	Member Data Documentation	79
3.4	BeeGame.Inventory.Player_Inventory.PlayerInventory Class Reference	80
3.4.1	Detailed Description	81
3.4.2	Member Function Documentation	81
3.4.3	Member Data Documentation	85
3.5	BeeGame.Inventory.ChestInventory Class Reference	85
3.5.1	Detailed Description	86
3.5.2	Member Function Documentation	86
3.5.3	Member Data Documentation	88

4	Chunk	89
4.1	BeeGame.Terrain.Chunks.Chunk Class Reference	89
4.1.1	Detailed Description	90
4.1.2	Member Function Documentation	91
4.1.3	Member Data Documentation	95
4.2	BeeGame.Terrain.Chunks.MeshData Class Reference	97
4.2.1	Detailed Description	98
4.2.2	Member Function Documentation	98
4.2.3	Member Data Documentation	100
4.3	BeeGame.Terrain.ChunkWorldPos Struct Reference	101
4.3.1	Detailed Description	102
4.3.2	Constructor & Destructor Documentation	102
4.3.3	Member Function Documentation	102
4.3.4	Member Data Documentation	104
4.4	BeeGame.Terrain.Chunks.LoadChunks Class Reference	105
4.4.1	Detailed Description	106
4.4.2	Member Function Documentation	106
4.4.3	Member Data Documentation	109
4.5	BeeGame.Terrain.Chunks.SaveChunk Class Reference	110
4.5.1	Detailed Description	111
4.5.2	Constructor & Destructor Documentation	111
4.5.3	Member Data Documentation	111
4.6	BeeGame.Terrain.LandGeneration.World Class Reference	112
4.6.1	Detailed Description	112
4.6.2	Member Function Documentation	112
4.6.3	Member Data Documentation	117
4.7	BeeGame.Terrain.LandGeneration.Terrain Class Reference	117
4.7.1	Detailed Description	118
4.7.2	Member Function Documentation	118
4.7.3	Member Data Documentation	126
4.8	BeeGame.Terrain.LandGeneration.TerrainGeneration Class Reference	126
4.8.1	Detailed Description	127
4.8.2	Member Function Documentation	127
4.8.3	Member Data Documentation	131
4.9	BeeGame.Terrain.LandGeneration.Noise.SimplexNoise Class Reference	134
4.9.1	Detailed Description	135
4.9.2	Member Function Documentation	135
4.9.3	Member Data Documentation	140

5	Player	141
5.1	BeeGame.Player.PlayerLook Class Reference	141
5.1.1	Detailed Description	142
5.1.2	Member Function Documentation	142
5.1.3	Member Data Documentation	143
5.2	BeeGame.Player.Selector Class Reference	144
5.2.1	Detailed Description	145
5.2.2	Member Function Documentation	145
5.2.3	Member Data Documentation	147
5.3	BeeGame.Player.PlayerMove Class Reference	148
5.3.1	Detailed Description	149
5.3.2	Member Function Documentation	149
5.3.3	Member Data Documentation	151
5.4	BeeGame.Player.SavePlayerPosition Class Reference	153
5.4.1	Detailed Description	153
5.4.2	Member Function Documentation	153
5.4.3	Member Data Documentation	154
6	Resources	154
6.1	BeeGame.LoadResources Class Reference	154
6.1.1	Detailed Description	154
6.1.2	Member Function Documentation	154
6.2	BeeGame.Core.PrefabDictionary Class Reference	155
6.2.1	Detailed Description	155
6.2.2	Member Function Documentation	155
6.2.3	Member Data Documentation	156
6.3	BeeGame.Core.SpriteDictionary Class Reference	156
6.3.1	Detailed Description	157
6.3.2	Member Function Documentation	157
6.3.3	Member Data Documentation	158
6.4	BeeGame.Core.BeeDictionarys Class Reference	158
6.4.1	Detailed Description	159
6.4.2	Member Function Documentation	159
6.4.3	Member Data Documentation	160
6.5	BeeGame.Resources.Resources Class Reference	160
6.5.1	Detailed Description	161
6.5.2	Constructor & Destructor Documentation	161
6.5.3	Member Function Documentation	161
6.5.4	Member Data Documentation	163
6.5.5	Property Documentation	163

7	Unity Type & Method Replacements	164
7.1	BeeGame.Core.THInput Class Reference	164
7.1.1	Detailed Description	165
7.1.2	Member Function Documentation	165
7.1.3	Member Data Documentation	168
7.2	BeeGame.Core.THVector2 Struct Reference	169
7.2.1	Detailed Description	169
7.2.2	Constructor & Destructor Documentation	170
7.2.3	Member Function Documentation	171
7.2.4	Member Data Documentation	176
7.3	BeeGame.Core.THVector3 Struct Reference	176
7.3.1	Detailed Description	178
7.3.2	Constructor & Destructor Documentation	178
7.3.3	Member Function Documentation	179
7.3.4	Member Data Documentation	189
8	Misc	190
8.1	BeeGame.Core.Extensions Class Reference	190
8.1.1	Detailed Description	190
8.1.2	Member Function Documentation	190
8.2	BeeGame.Serialization.Serialization Class Reference	191
8.2.1	Detailed Description	192
8.2.2	Member Function Documentation	192
8.2.3	Member Data Documentation	198
8.3	BeeGame.SpawnItem Class Reference	199
8.3.1	Detailed Description	199
8.3.2	Member Function Documentation	199
8.4	BeeGame.Test Class Reference	200
8.4.1	Detailed Description	200
8.4.2	Member Function Documentation	200

III	File Documentation	200
1	Items	200
1.1	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/Item.cs File Reference	200
1.2	Item.cs	201
1.3	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/HoneyComb.cs File Reference	204
1.4	HoneyComb.cs	204
1.5	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/ItemGameObject.cs File Reference	205
1.6	ItemGameObject.cs	205
1.7	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/ApplyColour.cs File Reference	206
1.8	ApplyColour.cs	206
2	Blocks	207
2.1	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Block.cs File Reference	207
2.2	Block.cs	207
2.3	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Air.cs File Reference	209
2.4	Air.cs	209
2.5	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Dirt.cs File Reference	209
2.6	Dirt.cs	210
2.7	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Grass.cs File Reference	210
2.8	Grass.cs	211
2.9	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Chest.cs File Reference	211
2.10	Chest.cs	212
2.11	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Apiary.cs File Reference	213
2.12	Apiary.cs	213
2.13	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Wood.cs File Reference	213
2.14	Wood.cs	214
2.15	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Leaves.cs File Reference	214
2.16	Leaves.cs	214
2.17	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Bedrock.cs File Reference	215
2.18	Bedrock.cs	215

3	Inventorys	216
3.1	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Inventory.cs File Reference	216
3.2	Inventory.cs	216
3.3	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Inventory↔ Slot.cs File Reference	217
3.4	InventorySlot.cs	217
3.5	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Player Inventory/PlayerInventory.cs File Reference	220
3.6	PlayerInventory.cs	220
3.7	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Chest↔ Inventory.cs File Reference	222
3.8	ChestInventory.cs	222
3.9	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/ItemsIn↔ Inventory.cs File Reference	223
3.10	ItemsInInventory.cs	224
4	Chunks	224
4.1	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/↔ Chunk.cs File Reference	224
4.2	Chunk.cs	225
4.3	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/↔ MeshData.cs File Reference	227
4.4	MeshData.cs	227
4.5	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunk↔ WorldPos.cs File Reference	228
4.6	ChunkWorldPos.cs	228
4.7	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/↔ LoadChunks.cs File Reference	229
4.8	LoadChunks.cs	229
4.9	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/↔ SaveChunk.cs File Reference	233
4.10	SaveChunk.cs	233
4.11	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Land↔ Generation/World.cs File Reference	233
4.12	World.cs	234
4.13	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Land↔ Generation/Terrain.cs File Reference	236
4.14	Terrain.cs	236
4.15	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Land↔ Generation/TerrainGeneration.cs File Reference	239
4.16	TerrainGeneration.cs	239
4.17	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Land↔ Generation/Noise/SimplexNoise.cs File Reference	241
4.18	SimplexNoise.cs	241

5	Player	245
5.1	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/PlayerLook.cs File Reference	245
5.2	PlayerLook.cs	245
5.3	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/Selector.cs File Reference	246
5.4	Selector.cs	246
5.5	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/Player↔ Move.cs File Reference	248
5.6	PlayerMove.cs	248
5.7	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/SavePlayer↔ Position.cs File Reference	249
5.8	SavePlayerPosition.cs	249
6	Resources	250
6.1	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/↔ PrefabDictionary.cs File Reference	250
6.2	PrefabDictionary.cs	250
6.3	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/↔ SpriteDictionary.cs File Reference	250
6.4	SpriteDictionary.cs	251
6.5	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/↔ BeeDictionarys.cs File Reference	251
6.6	BeeDictionarys.cs	251
6.7	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Resources/Resources.Designer.↔ cs File Reference	252
6.8	Resources.Designer.cs	252
6.9	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/LoadResources.cs File Reference	254
6.10	LoadResources.cs	254
7	Unity Type & Method Replacements	254
7.1	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityType↔ Replacements/THInput.cs File Reference	254
7.2	THInput.cs	255
7.3	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityType↔ Replacements/THVector2.cs File Reference	256
7.4	THVector2.cs	257
7.5	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityType↔ Replacements/THVector3.cs File Reference	258
7.6	THVector3.cs	259

8 Misc	261
8.1 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Serialization/↔ Serialization.cs File Reference	261
8.2 Serialization.cs	261
8.3 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Extensions.cs File Reference	263
8.4 Extensions.cs	264
8.5 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Enums/↔ Enums.cs File Reference	264
8.6 Enums.cs	266
8.7 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/SpawnItem.cs File Reference	267
8.8 SpawnItem.cs	267
8.9 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/test.cs File Reference	267
8.10 test.cs	268
Index	269

Part I

Namespace Documentation

0.1 BeeGame Namespace Reference

Namespaces

- namespace [Blocks](#)
- namespace [Core](#)
- namespace [Inventory](#)
- namespace [Items](#)
- namespace [Player](#)
- namespace [Resources](#)
- namespace [Serialization](#)
- namespace [Terrain](#)

Classes

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

0.2 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](#)
- class [Dirt](#)
Dirt Block
- class [Grass](#)
Grass Block
- class [Leaves](#)
- class [Wood](#)

0.3 BeeGame.Core Namespace Reference

Namespaces

- namespace [Enums](#)

Classes

- class [BeeDictionary](#)
- class [Extensions](#)
- class [PrefabDictionary](#)
The prefabs available to the game
- class [SpriteDictionary](#)
All of the sprites available to the game
- 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

0.4 BeeGame.Core.Enums Namespace Reference

Enumerations

- enum [ItemType](#) { [ItemType.ITEM](#), [ItemType.BEE](#) }
The item types
- 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.PRINCESS](#), [BeeType.DRONE](#) }
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

0.4.1 Enumeration Type Documentation

0.4.1.1 BeeEffect

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

Any effects of the bee

Enumerator

NONE	
POSITION	

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

```
00064    {  
00065        NONE, POSITION  
00066    }
```

0.4.1.2 BeeHumidityPreference

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

Humidity preferences of the bee

Enumerator

ARID	
DRY	
TEMPERATE	
MOIST	
HUMID	

Definition at line 71 of file [Enums.cs](#).

```
00072    {  
00073        ARID, DRY, TEMPERATE, MOIST, HUMID  
00074    };
```

0.4.1.3 BeeLifeSpan

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

The lifespan of the bee

Enumerator

HUMMINGBIRD	
SHORTEST	
SHORT	
NORMAL	
LONG	
LONGEST	
SEATURTLE	

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

```
00048     {  
00049         HUMMINGBIRD, SHORTEST, SHORT, NORMAL, LONG,  
00050         LONGEST, SEATURTLE  
    };
```

0.4.1.4 BeeProductionSpeed

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

How fast the bee produces items

Enumerator

SLOW	
NORMAL	
FAST	

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

```
00056     {  
00057         SLOW, NORMAL, FAST  
00058     };
```

0.4.1.5 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 23 of file [Enums.cs](#).

```
00024     {  
00025         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  
00026     };
```

0.4.1.6 BeeTempPreference

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

The different bee temp preferences

Enumerator

FROZEN	
COLD	
TEMPERATE	
HOT	
Generated by Pexen	

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

```
00040    {  
00041        FROZEN, COLD, TEMPERATE, HOT, HELL  
00042    };
```

0.4.1.7 BeeType

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

The different bee types

Enumerator

QUEEN	
PRINCESS	
DRONE	

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

```
00032    {  
00033        QUEEN, PRINCESS, DRONE  
00034    };
```

0.4.1.8 Direction

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

Direction in the game

Enumerator

NORTH	
EAST	
SOUTH	
WEST	
UP	
DOWN	

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

```
00081    {  
00082        NORTH, EAST, SOUTH, WEST, UP, DOWN  
00083    };
```

0.4.1.9 HoneyCombType

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

Honey Comb Types

Enumerator

HONEY	
ICEY	

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

```
00015     {
00016         HONEY, ICEY
00017     };
```

0.4.1.10 ItemType

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

The item types

Enumerator

ITEM	
BEE	

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

```
00007     {
00008         ITEM, BEE
00009     };
```

0.5 BeeGame.Inventory Namespace Reference

Namespaces

- namespace [Player_Inventory](#)

Classes

- class [ChestInventory](#)
Incentory for the chests
- class [Inventory](#)
Base class for all inventories in the game
- class [InventorySlot](#)
- class [ItemsInInventory](#)
Class that holds all of the items in the inventory. Can be serialized so inventory may be saved

0.6 BeeGame.Inventory.Player_Inventory Namespace Reference

Classes

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

0.7 BeeGame.Items Namespace Reference

Classes

- class [ApplyColour](#)
Applies a given colour to a gameobject
- 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
- struct [Tile](#)
Position of the items texture

0.8 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

0.9 BeeGame.Resources Namespace Reference

Classes

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

0.10 BeeGame.Serialization Namespace Reference

Classes

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

0.11 BeeGame.Terrain Namespace Reference

Namespaces

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

Classes

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

0.12 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

0.13 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

0.14 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/>

Part II

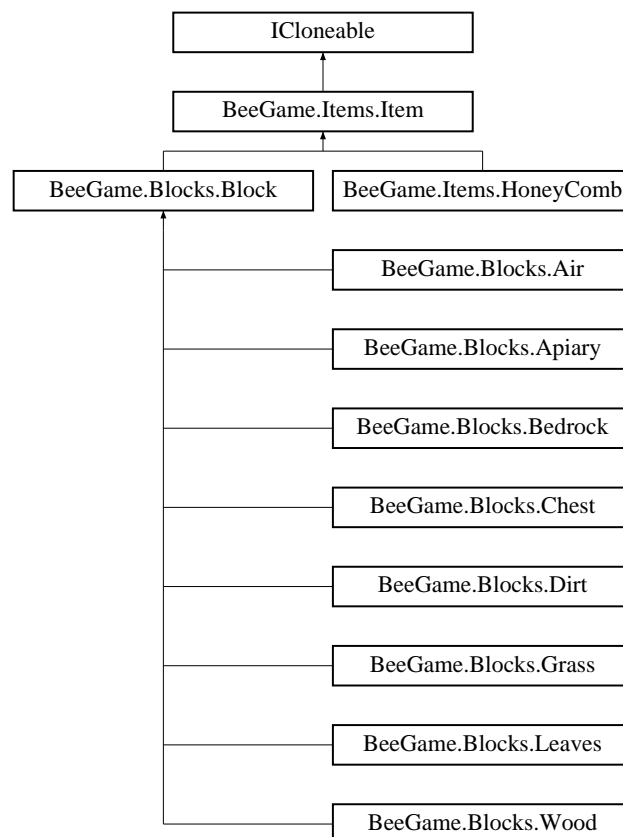
Class Documentation

1 Items

1.1 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 GameObject [GetGameObject](#) ()
Returns the GameObject for the item of it has one
- virtual string [GetItemID](#) ()
Returns the id for the item as a string
- virtual Sprite [GetItemSprite](#) ()
Returns the sprite for the item
- virtual string [GetItemName](#) ()
- 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

- bool [placeable](#) = false
Is this item placeable. Saves checking if the item is a block type
- bool [usesGameObject](#) = false
Does the item use a gameobject
- int [itemStackCount](#) = 1
Number of items in the stack
- int [maxStackCount](#) = 64
Max number of items in a stack

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

Package Attributes

- string [itemName](#) = "Test Item"
Name of the item

Private Attributes

- const float [tileSize](#) = 0.1f
How big are the texture tiles in the texture map (1/tile number x)

1.1.1 Detailed Description

Base class for all [Items](#) and [Blocks](#) in the game

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

1.1.2 Constructor & Destructor Documentation

1.1.2.1 Item() [1/2]

BeeGame.Items.Item.Item ()

Definition at line 46 of file [Item.cs](#).

```
00047     {
00048         itemName = "TestItem";
00049     }
```

1.1.2.2 Item() [2/2]

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

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

```
00052     {
00053         itemName = name;
00054     }
```

1.1.3 Member Function Documentation

1.1.3.1 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 314 of file [Item.cs](#).

```
00315     {
00316         /** Saves this to a file then reads it back so that a copy and not a reference is passed
00317         BinaryFormatter bf = new BinaryFormatter();
00318         MemoryStream ms = new MemoryStream();
00319
00320         bf.Serialize(ms, this);
00321         ms.Seek(0, SeekOrigin.Begin);
00322
00323         return bf.Deserialize(ms);
00324     }
```

1.1.3.2 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 351 of file [Item.cs](#).

```
00352     {
00353         if (!(obj is Item))
00354             return false;
00355
00356         return this == (obj as Item);
00357     }
```

1.1.3.3 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 178 of file [Item.cs](#).

```
00179     {
00180         /* Adds vertices in a anti-clockwise order
00181         meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z -
blockSize), addToRenderMesh);
00182         meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z -
blockSize), addToRenderMesh);
00183         meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z +
blockSize), addToRenderMesh);
00184         meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z +
blockSize), addToRenderMesh);
00185
00186         /* adds teh tirs for the quad
00187         meshData.AddQuadTriangles(addToRenderMesh);
00188
00189         /* if the data should be added to the render mesh also add the uvs to the mesh
00190         if (addToRenderMesh)
00191             meshData.uv.AddRange(FaceUVs(Direction.DOWNN));
00192
00193         return meshData;
00194     }
```

1.1.3.4 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 234 of file [Item.cs](#).

```
00235     {
00236         /* Adds vertices in a anti-clockwise order
00237         meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z -
00238         blockSize), addToRenderMesh);
00239         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z -
00240         blockSize), addToRenderMesh);
00241         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z +
00242         blockSize), addToRenderMesh);
00243         meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z +
00244         blockSize), addToRenderMesh);
00245         /* adds teh tirs for the quad
00246         meshData.AddQuadTriangles(addToRenderMesh);
00247         /* if the data should be added to the render mesh also add the uvs to the mesh
00248         if (addToRenderMesh)
00249             meshData.uv.AddRange(FaceUVs(Direction.EAST));
00250         return meshData;
00251     }
```

1.1.3.5 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 206 of file [Item.cs](#).

```

00207     {
00208         /* Adds vertices in a anti-clockwise order
00209         meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z +
00210         blockSize), addToRenderMesh);
00210         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z +
00211         blockSize), addToRenderMesh);
00211         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z +
00212         blockSize), addToRenderMesh);
00212         meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z +
00213         blockSize), addToRenderMesh);
00213
00214         /* adds teh tirs for the quad
00215         meshData.AddQuadTriangles (addToRenderMesh);
00216
00217         /* if the data should be added to the render mesh also add the uvs to the mesh
00218         if (addToRenderMesh)
00219             meshData.uv.AddRange (FaceUVs (Direction.NORTH));
00220
00221         return meshData;
00222     }

```

1.1.3.6 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 262 of file [Item.cs](#).

```

00263     {
00264         /* Adds vertices in a anti-clockwise order
00265         meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z -
blockSize), addToRenderMesh);
00266         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z -
blockSize), addToRenderMesh);
00267         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z -
blockSize), addToRenderMesh);
00268         meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z -
blockSize), addToRenderMesh);
00269
00270         /* adds teh tirs for the quad
00271         meshData.AddQuadTriangles (addToRenderMesh);
00272
00273         /* if the data should be added to the render mesh also add the uvs to the mesh
00274         if (addToRenderMesh)
00275             meshData.uv.AddRange(FaceUVs(Direction.SOUTH));
00276
00277         return meshData;
00278     }

```

1.1.3.7 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 150 of file [Item.cs](#).

```

00151     {
00152         /* Adds vertices in a anti-clockwise order
00153         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z +
blockSize), addToRenderMesh, Direction.UP);

```

```

00154         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z +
blockSize), addToRenderMesh, Direction.UP);
00155         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z -
blockSize), addToRenderMesh, Direction.UP);
00156         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z -
blockSize), addToRenderMesh, Direction.UP);
00157
00158         /* adds teh tirs for the quad
00159         meshData.AddQuadTriangles(addToRenderMesh);
00160
00161         /* if the data should be added to the render mesh also add the uvs to the mesh
00162         if (addToRenderMesh)
00163             meshData.uv.AddRange(FaceUVs(Direction.UP));
00164
00165         return meshData;
00166     }

```

1.1.3.8 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 290 of file [Item.cs](#).

```

00291     {
00292         /* Adds vertices in a anti-clockwise order
00293         meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z +
blockSize), addToRenderMesh);
00294         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z +
blockSize), addToRenderMesh);
00295         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z -
blockSize), addToRenderMesh);
00296         meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z -
blockSize), addToRenderMesh);
00297
00298         /* adds teh tirs for the quad
00299         meshData.AddQuadTriangles(addToRenderMesh);
00300
00301         /* if the data should be added to the render mesh also add the uvs to the mesh
00302         if (addToRenderMesh)
00303             meshData.uv.AddRange(FaceUVs(Direction.WEST));
00304
00305         return meshData;
00306     }

```

1.1.3.9 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 125 of file [Item.cs](#).

```
00126     {
00127         /* only 4 uvs per face
00128         Vector2[] UVs = new Vector2[4];
00129         Tile tilePos = TexturePosition(direction);
00130
00131         /* sets the UVs for each vertex
00132         UVs[0] = new THVector2(tileSize * tilePos.x +
00133         tileSize - 0.01f, tileSize * tilePos.y + 0.01f);
00133         UVs[1] = new THVector2(tileSize * tilePos.x +
00134         tileSize - 0.01f, tileSize * tilePos.y + tileSize - 0.01f);
00134         UVs[2] = new THVector2(tileSize * tilePos.x + 0.01f,
00135         tileSize * tilePos.y + tileSize - 0.01f);
00135         UVs[3] = new THVector2(tileSize * tilePos.x + 0.01f,
00136         tileSize * tilePos.y + 0.01f);
00136
00137         return UVs;
00138     }
```

1.1.3.10 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.Items.HoneyComb](#), and [BeeGame.Blocks.Chest](#).

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

```
00062 { return null; }
```

1.1.3.11 GetHashCode()

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

Returns the hashcode for the item

Returns

1

Definition at line 341 of file [Item.cs](#).

```
00342     {  
00343         return 1;  
00344     }
```

1.1.3.12 GetItemID()

```
virtual string BeeGame.Items.Item.GetItemID ( ) [virtual]
```

Returns the id for the item as a string

Returns

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

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

```
00069     {  
00070         return $"{GetHashCode()}";  
00071     }
```

1.1.3.13 GetItemName()

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

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

Definition at line 82 of file [Item.cs](#).

```
00083     {  
00084         return $"{itemName}";  
00085     }
```

1.1.3.14 GetItemSprite()

```
virtual Sprite BeeGame.Items.Item.GetItemSprite ( ) [virtual]
```

Returns the sprite for the item

Returns

Sprite for this item

Reimplemented in [BeeGame.Items.HoneyComb](#), [BeeGame.Blocks.Block](#), [BeeGame.Blocks.Grass](#), [BeeGame.Blocks.Dirt](#), [BeeGame.Blocks.Wood](#), and [BeeGame.Blocks.Leaves](#).

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

```
00078     {
00079         return SpriteDictionary.GetSprite("TestSprite");
00080     }
```

1.1.3.15 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 107 of file [Item.cs](#).

```
00108     {
00109         /* adds all faces of the item to the mesh as all faces could be seen at any time
00110         meshData = FaceDataUp(x, y, z, meshData, true, 0.25f);
00111         meshData = FaceDataDown(x, y, z, meshData, true, 0.25f);
00112         meshData = FaceDataNorth(x, y, z, meshData, true, 0.25f);
00113         meshData = FaceDataEast(x, y, z, meshData, true, 0.25f);
00114         meshData = FaceDataSouth(x, y, z, meshData, true, 0.25f);
00115         meshData = FaceDataWest(x, y, z, meshData, true, 0.25f);
00116
00117         return meshData;
00118     }
```


1.1.3.16 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 384 of file [Item.cs](#).

```
00385     {
00386         return !(a == b);
00387     }
```

1.1.3.17 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 365 of file [Item.cs](#).

```
00366     {
00367         if (ReferenceEquals(a, null) && ReferenceEquals(b, null))
00368             return true;
00369         if (ReferenceEquals(a, null) || ReferenceEquals(b, null))
00370             return false;
00371
00372         if (a.GetItemID() == b.GetItemID())
00373             return true;
00374
00375         return false;
00376     }
```

1.1.3.18 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.Grass](#), [BeeGame.Blocks.Bedrock](#), [BeeGame.Blocks.Dirt](#), [BeeGame.Blocks.Chest](#), [BeeGame.Blocks.Wood](#), and [BeeGame.Blocks.Leaves](#).

Definition at line 94 of file [Item.cs](#).

```
00095         {
00096             return new Tile() { x = 1, y = 9 };
00097         }
```

1.1.3.19 ToString()

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

Returns the item name an id formatted nicely

Returns

Definition at line 332 of file [Item.cs](#).

```
00333         {
00334             return $"{itemName} \nID: {GetItemID()}";
00335         }
```

1.1.4 Member Data Documentation

1.1.4.1 itemName

```
string BeeGame.Items.Item.itemName = "Test Item" [package]
```

Name of the item

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

1.1.4.2 itemStackCount

```
int BeeGame.Items.Item.itemStackCount = 1
```

Number of items in the stack

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

1.1.4.3 maxStackCount

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

Max number of items in a stack

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

1.1.4.4 placeable

```
bool BeeGame.Items.Item.placeable = false
```

Is this item placeable. Saves checking if the item is a block type

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

1.1.4.5 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 33 of file [Item.cs](#).

1.1.4.6 usesGameObject

```
bool BeeGame.Items.Item.usesGameObject = false
```

Does the item use a gameobject

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

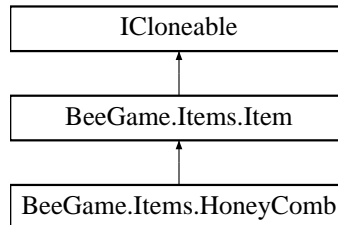
The documentation for this class was generated from the following file:

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

1.2 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
- 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](#)

Public Attributes

- [HoneyCombType type](#)
The type of comb this is, [HoneyCombType](#)

Properties

- Color [CombColour](#) [get]
The colour if this coumb, [BeeDictionary.GetCombColour\(HoneyCombType\)](#)

Additional Inherited Members

1.2.1 Detailed Description

Honey comb item produced by bees

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

1.2.2 Constructor & Destructor Documentation

1.2.2.1 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 37 of file [HoneyComb.cs](#).

```
00037             : base(new CultureInfo("en-US", false).TextInfo.ToTitleCase($"
{HoneyCombType.HONEY} Comb".ToLower()))
00038         {
00039             usesGameObject = true;
00040             type = HoneyCombType.HONEY;
00041         }
```

1.2.2.2 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 47 of file [HoneyComb.cs](#).

```
00047             : base(new CultureInfo("en-US", false).TextInfo.ToTitleCase($"
{type.ToString()} Comb".ToLower()))
00048         {
00049             usesGameObject = true;
00050             this.type = type;
00051         }
```

1.2.3 Member Function Documentation

1.2.3.1 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 68 of file [HoneyComb.cs](#).

```
00069         {
00070             GameObject obj = PrefabDictionary.GetPrefab("HoneyComb");
00071             /* cannot access the instance material from here have to do it on the object
00072             obj.GetComponent<ApplyColour>().colour = CombColour;
00073             return obj;
00074         }
```

1.2.3.2 GetHashCode()

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

Returns the hashcode for this [Item](#)

Returns

8

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

```
00092         {
00093             return 8;
00094         }
```

1.2.3.3 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

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

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

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

1.2.3.4 GetItemSprite()

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

Retuens the sprite for the this

Returns

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

Definition at line 59 of file [HoneyComb.cs](#).

```
00060      {  
00061          return SpriteDictionary.GetSprite("HoneyComb");  
00062      }
```

1.2.4 Member Data Documentation

1.2.4.1 type

[HoneyCombType](#) BeeGame.Items.HoneyComb.type

The type of comb this is, HoneyCombType

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

1.2.5 Property Documentation

1.2.5.1 CombColour

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

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

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

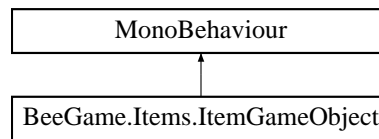
The documentation for this class was generated from the following file:

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

1.3 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

1.3.1 Detailed Description

Interface between item and inity gameobjects

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

1.3.2 Member Function Documentation

1.3.2.1 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     }
  
```


1.3.2.2 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     }
```

1.3.2.3 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     }
```

1.3.3 Member Data Documentation

1.3.3.1 go

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

GameObject to make

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

1.3.3.2 item

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

[Item](#) that this gameobject represents

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

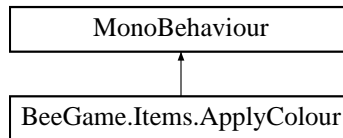
The documentation for this class was generated from the following file:

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

1.4 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

1.4.1 Detailed Description

Applies a given colour to a gameobject

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

1.4.2 Member Function Documentation

1.4.2.1 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         }
```

1.4.3 Member Data Documentation

1.4.3.1 colour

`Color BeeGame.Items.ApplyColour.colour`

Colour to apply

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

1.4.3.2 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:

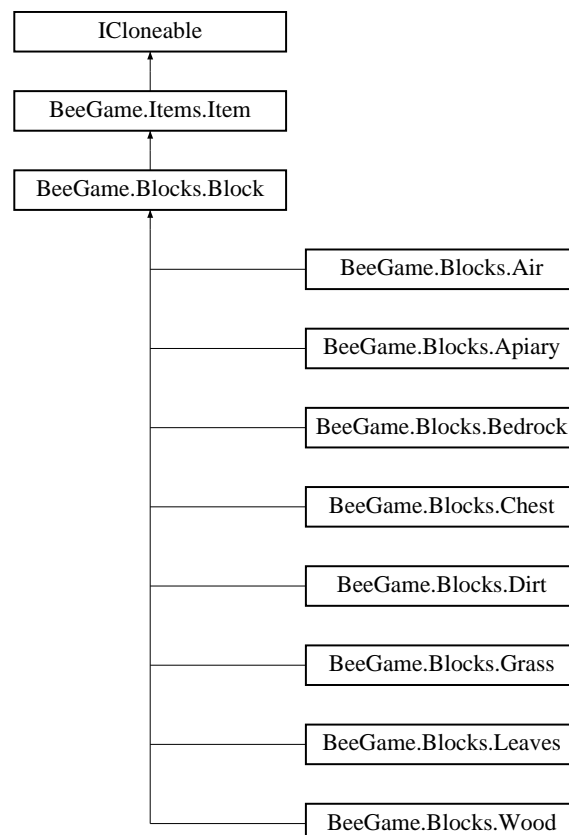
- [C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/ApplyColour.cs](#)

2 Blocks

2.1 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

Additional Inherited Members

2.1.1 Detailed Description

Base class for blocks

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

2.1.2 Constructor & Destructor Documentation

2.1.2.1 Block() [1/2]

```
BeeGame.Blocks.Block.Block ( )
```

Constructor sets the Item.placeable to true

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

```

00030             : base()
00031     {
00032         itemName = "Stone";
00033         placeable = true;
00034     }

```

2.1.2.2 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 40 of file [Block.cs](#).

```

00040             : base(name)
00041     {
00042         placeable = true;
00043     }

```

2.1.3 Member Function Documentation

2.1.3.1 BlockData()

```

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

```

The data that this block adds to the mesh

Parameters

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

Returns

Given *meshData* with this blocks data added to it

If no data of either collider or render should be added override to return the givn mesh.
If only collsion data should be added override to say render mesh false.

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

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

```

00099     {
00100         /* Adds the Top face of the block
00101         if (!chunk.GetBlock(x, y + 1, z, false).IsSolid(Direction.DOWN))
00102         {
00103             meshData = FaceDataUp(x, y, z, meshData, addToRenderMesh);
00104         }
00105
00106         /* Adds the Bottom face of the block
00107         if (!chunk.GetBlock(x, y - 1, z, false).IsSolid(Direction.UP))
00108         {
00109             meshData = FaceDataDown(x, y, z, meshData, addToRenderMesh);
00110         }
00111
00112         /* Adds the North face of the block
00113         if (!chunk.GetBlock(x, y, z + 1, false).IsSolid(Direction.SOUTH))
00114         {
00115             meshData = FaceDataNorth(x, y, z, meshData, addToRenderMesh);
00116         }
00117
00118         /* Adds the South face of the block
00119         if (!chunk.GetBlock(x, y, z - 1, false).IsSolid(Direction.NORTH))
00120         {
00121             meshData = FaceDataSouth(x, y, z, meshData, addToRenderMesh);
00122         }
00123
00124         /* Adds the East face of the block
00125         if (!chunk.GetBlock(x + 1, y, z, false).IsSolid(Direction.WEST))
00126         {
00127             meshData = FaceDataEast(x, y, z, meshData, addToRenderMesh);
00128         }
00129
00130         /* Adds the West face of the block
00131         if (!chunk.GetBlock(x - 1, y, z, false).IsSolid(Direction.EAST))
00132         {
00133             meshData = FaceDataWest(x, y, z, meshData, addToRenderMesh);
00134         }
00135         return meshData;
00136     }
00137 
```

2.1.3.2 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.Chest](#), [BeeGame.Blocks.Bedrock](#), and [BeeGame.Blocks.Air](#).

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

```
00059         {
00060             GameObject go = Object.Instantiate(UnityEngine.Resources.Load("
Prefabs/ItemGameObject") as GameObject, pos, Quaternion.identity) as GameObject;
00061             go.GetComponent<ItemGameObject>().item = this;
00062         }
```

2.1.3.3 GetHashCode()

```
override int BeeGame.Blocks.Block.GetHashCode ( )
```

Hascode for the [Block](#)

Returns

1

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

```
00156         {
00157             return 1;
00158         }
```

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

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

```
00048         {
00049             return SpriteDictionary.GetSprite("Stone");
00050         }
```

2.1.3.5 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 77 of file [Block.cs](#).

```
00078         {
00079             return false;
00080         }
```

2.1.3.6 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 144 of file [Block.cs](#).

```
00145         {
00146             return true;
00147         }
```

2.1.3.7 ToString()

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

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

Returns

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

```
00165         {
00166             return $"{itemName} \nID: {GetHashCode()}";
00167         }
```

2.1.3.8 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 71 of file [Block.cs](#).

```
00071 { }
```

2.1.4 Member Data Documentation

2.1.4.1 breakable

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

Can this [Block](#) be broken

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

2.1.4.2 changed

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

Has this block been placed by the player

Definition at line 23 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](#)

2.2 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

2.2.1 Detailed Description

Position of the items texture

Definition at line 395 of file [Item.cs](#).

2.2.2 Member Data Documentation

2.2.2.1 x

```
int BeeGame.Items.Tile.x
```

X pos of the texture

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

2.2.2.2 y

```
int BeeGame.Items.Tile.y
```

Y pos of the texture

Definition at line 404 of file [Item.cs](#).

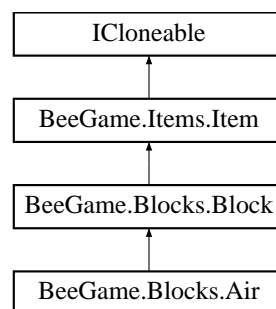
The documentation for this struct was generated from the following file:

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

2.3 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

Additional Inherited Members

2.3.1 Detailed Description

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

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

2.3.2 Constructor & Destructor Documentation

2.3.2.1 Air()

BeeGame.Blocks.Air.Air ()

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

```
00014         : base("Air")
00015     {
00016     }
```

2.3.3 Member Function Documentation

2.3.3.1 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 31 of file [Air.cs](#).

```
00032     {
00033         return meshData;
00034     }
```

2.3.3.2 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 22 of file [Air.cs](#).

```
00023     {  
00024         return;  
00025     }
```

2.3.3.3 GetHashCode()

```
override int BeeGame.Blocks.Air.GetHashCode ( )
```

Hashcode acts as the base ID for an item

Returns

2

Definition at line 50 of file [Air.cs](#).

```
00051     {  
00052         return 2;  
00053     }
```

2.3.3.4 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 41 of file [Air.cs](#).

```
00042     {  
00043         return false;  
00044     }
```

2.3.3.5 ToString()

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

Gets the item name and ID in a nice format

Returns

Definition at line 59 of file [Air.cs](#).

```
00060      {
00061          return $"{itemName} \nID: {GetItemID()}";
00062      }
```

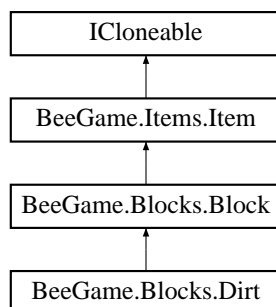
The documentation for this class was generated from the following file:

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

2.4 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

Additional Inherited Members

2.4.1 Detailed Description

Dirt Block

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

2.4.2 Constructor & Destructor Documentation

2.4.2.1 Dirt()

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

Constructor

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

```
00019 : base("Dirt"){}
```

2.4.3 Member Function Documentation

2.4.3.1 GetHashCode()

```
override int BeeGame.Blocks.Dirt.GetHashCode ( )
```

Base ID of the block

Returns

5

Definition at line 46 of file [Dirt.cs](#).

```
00047         {  
00048             return 5;  
00049         }
```

2.4.3.2 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 23 of file [Dirt.cs](#).

```
00024     {  
00025         return SpriteDictionary.GetSprite("Dirt");  
00026     }
```

2.4.3.3 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 35 of file [Dirt.cs](#).

```
00036         {  
00037             return new Tile { x = 2, y = 9 };  
00038         }
```

2.4.3.4 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 55 of file [Dirt.cs](#).

```
00056         {  
00057             return $"{itemName} \nID: {GetItemID()}";  
00058         }
```

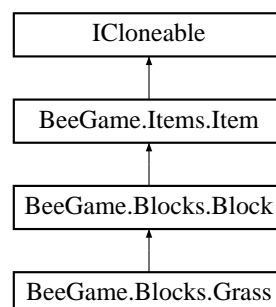
The documentation for this class was generated from the following file:

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

2.5 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 string [GetItemName](#) ()
- override int [GetHashCode](#) ()
The Base id for the block
- override string [ToString](#) ()
REturns the name and value for the block as a string

Additional Inherited Members

2.5.1 Detailed Description

[Grass Block](#)

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

2.5.2 Constructor & Destructor Documentation

2.5.2.1 [Grass\(\)](#)

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

Constructor also sets teh items name

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

```
00020 : base("Grass") {}
```

2.5.3 Member Function Documentation

2.5.3.1 GetHashCode()

```
override int BeeGame.Blocks.Grass.GetHashCode ( )
```

The Base id for the block

Returns

4

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

```
00086         {  
00087             return 4;  
00088         }
```

2.5.3.2 GetItemName()

```
override string BeeGame.Blocks.Grass.GetItemName ( ) [virtual]
```

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

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

```
00077         {  
00078             return "Grass";  
00079         }
```

2.5.3.3 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 24 of file [Grass.cs](#).

```
00025         {  
00026             return SpriteDictionary.GetSprite("Grass");  
00027         }
```

2.5.3.4 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 49 of file [Grass.cs](#).

```

00050     {
00051         //All textures are on the same Y value for the texture atlas so Y can be set
00052         Tile tile = new Tile()
00053     {
00054         y = 9
00055     };
00056
00057     switch (direction)
00058     {
00059         //if we want the top face return the full grass texture
00060         case Direction.UP:
00061             tile.x = 3;
00062             return tile;
00063         //if we want the bottom face return the dirt texture
00064         case Direction.DOWN:
00065             tile.x = 2;
00066             return tile;
00067         //return the 1/2 grass texture if a side face is wanted
00068         default:
00069             tile.x = 4;
00070             return tile;
00071     }
00072 }
```

2.5.3.5 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 94 of file [Grass.cs](#).

```

00095     {
00096         return $"{itemName} \nID: {GetItemID()}";
00097     }
```

2.5.3.6 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 38 of file [Grass.cs](#).

```

00039         {
00040             if (chunk.GetBlock(x, y + 1, z, false).IsSolid(Direction.DOWN))
00041                 chunk.blocks[x, y, z] = new Dirt() { changed =
changed };
00042         }

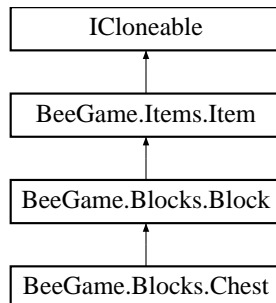
```

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

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

2.6 BeeGame.Blocks.Chest Class Reference

Inheritance diagram for BeeGame.Blocks.Chest:



Public Member Functions

- [Chest](#) ()
- override GameObject [GetGameObject](#) ()
Returns the GameObject for the item of it has one
- override [Tile TexturePosition](#) ([Direction](#) direction)
Texture position of the items texture
- override [MeshData BlockData](#) ([Chunk](#) chunk, int x, int y, int z, [MeshData](#) meshData, bool addToRender↔ Mesh=true)
The data that this block adds to the mesh
- override bool [InteractWithBlock](#) ([BeeGame.Inventory.Inventory](#) inv)
- override void [BreakBlock](#) ([THVector3](#) pos)
Spawns an item with the same texture as the broken block
- override int [GetHashCode](#) ()
- override string [ToString](#) ()

Private Attributes

- GameObject [myGameObject](#)

Additional Inherited Members

2.6.1 Detailed Description

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

2.6.2 Constructor & Destructor Documentation

2.6.2.1 Chest()

`BeeGame.Blocks.Chest.Chest ()`

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

```
00017                                     : base("Chest")
00018     {
00019         usesGameObject = true;
00020     }
```

2.6.3 Member Function Documentation

2.6.3.1 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

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

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

```

00034         {
00035             if (myGameObject == null)
00036             {
00037                 myGameObject = UnityEngine.Object.Instantiate(
00038                     PrefabDictionary.GetPrefab("Chest"), new THVector3(x, y, z) + chunk.
00039                     chunkWorldPos, Quaternion.identity, chunk.transform);
00038                 myGameObject.GetComponent<ChestInventory>().inventoryPosition =
00039                 new THVector3(x, y, z) + chunk.chunkWorldPos;
00039                 myGameObject.GetComponent<ChestInventory>().SetChestInventory();
00040             }
00041             return base.BlockData(chunk, x, y, z, meshData, true);
00042         }

```

2.6.3.2 BreakBlock()

```

override void BeeGame.Blocks.Chest.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 from [BeeGame.Blocks.Block](#).

Definition at line 50 of file [Chest.cs](#).

```

00051         {
00052             Serialization.Serialization.DeleteFile(myGameObject.GetComponent<
00053             ChestInventory>().inventoryName);
00053             UnityEngine.Object.Destroy(myGameObject);
00054             base.BreakBlock(pos);
00055         }

```

2.6.3.3 GetGameObject()

```

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

```

Returns the GameObject for the item of it has one

Returns

GameObject for the item

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

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

```
00023     {
00024
00025         return PrefabDictionary.GetPrefab("Chest");
00026     }
```

2.6.3.4 GetHashCode()

```
override int BeeGame.Blocks.Chest.GetHashCode ( )
```

Definition at line 57 of file [Chest.cs](#).

```
00058     {
00059         return 8;
00060     }
```

2.6.3.5 InteractWithBlock()

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

Definition at line 44 of file [Chest.cs](#).

```
00045     {
00046         myGameObject.GetComponent<ChestInventory>().ToggleInventory(inv);
00047         return true;
00048     }
```

2.6.3.6 TexturePosition()

```
override Tile BeeGame.Blocks.Chest.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 28 of file [Chest.cs](#).

```
00029         {  
00030             return new Tile() { x = 0, y = 9 };  
00031         }
```

2.6.3.7 ToString()

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

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

```
00063         {  
00064             return $"{itemName}\nID{GetItemID()}";  
00065         }
```

2.6.4 Member Data Documentation

2.6.4.1 myGameobject

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

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

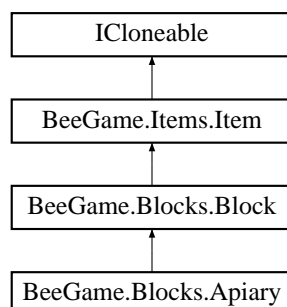
The documentation for this class was generated from the following file:

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

2.7 BeeGame.Blocks.Apiary Class Reference

[Apiary Block](#)

Inheritance diagram for BeeGame.Blocks.Apiary:



Public Member Functions

- [Apiary \(\)](#)
Constructor
- override int [GetHashCode \(\)](#)
ID of the item
- override string [ToString \(\)](#)
The item name and ID as a string

Additional Inherited Members

2.7.1 Detailed Description

[Apiary Block](#)

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

2.7.2 Constructor & Destructor Documentation

2.7.2.1 Apiary()

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

Constructor

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

```
00014                                     : base("Apiary")
00015     {
00016     }
```

2.7.3 Member Function Documentation

2.7.3.1 GetHashCode()

```
override int BeeGame.Blocks.Apiary.GetHashCode ( )
```

ID of the item

Returns

3

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

```
00025     {
00026         return 3;
00027     }
```

2.7.3.2 ToString()

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

The item name and ID as a string

Returns

A nicely formatted string

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

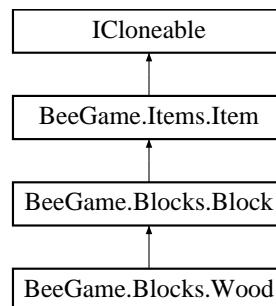
```
00034     {
00035         return $"{itemName} \nID: {GetItemID()}";
00036     }
```

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

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

2.8 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

Additional Inherited Members

2.8.1 Detailed Description

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

2.8.2 Constructor & Destructor Documentation

2.8.2.1 Wood()

BeeGame.Blocks.Wood.Wood ()

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

```
00015             : base("Wood")
00016         {
00017
00018         }
```

2.8.3 Member Function Documentation

2.8.3.1 GetHashCode()

override int BeeGame.Blocks.Wood.GetHashCode ()

Base ID of the block

Returns

5

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

```
00038         {
00039             return 6;
00040         }
```

2.8.3.2 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 21 of file [Wood.cs](#).

```
00022         {
00023             return SpriteDictionary.GetSprite("Wood");
00024         }
```

2.8.3.3 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 27 of file [Wood.cs](#).

```
00028         {  
00029             return new Tile() { x = 7, y = 9 };  
00030         }
```

2.8.3.4 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 46 of file [Wood.cs](#).

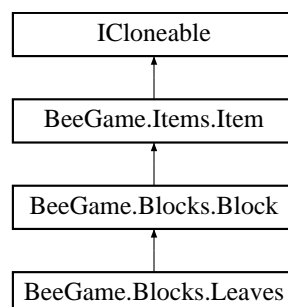
```
00047         {  
00048             return $"{itemName} \nID: {GetItemID()}";  
00049         }
```

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

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

2.9 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

Additional Inherited Members

2.9.1 Detailed Description

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

2.9.2 Constructor & Destructor Documentation

2.9.2.1 Leaves()

`BeeGame.Blocks.Leaves.Leaves ()`

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

```

00013             : base("Leaves")
00014         {
00015
00016         }
```

2.9.3 Member Function Documentation

2.9.3.1 GetHashCode()

`override int BeeGame.Blocks.Leaves.GetHashCode ()`

Base ID of the block

Returns

5

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

```

00041         {
00042             return 7;
00043         }
```

2.9.3.2 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 19 of file [Leaves.cs](#).

```
00020      {  
00021          return SpriteDictionary.GetSprite("Leaves");  
00022      }
```

2.9.3.3 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 30 of file [Leaves.cs](#).

```
00031      {  
00032          return false;  
00033      }
```

2.9.3.4 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 25 of file [Leaves.cs](#).

```
00026      {
00027          return new Tile() { x = 5, y = 9 };
00028      }
```

2.9.3.5 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 49 of file [Leaves.cs](#).

```
00050      {
00051          return $"{itemName} \nID: {GetItemID()}";
00052      }
```

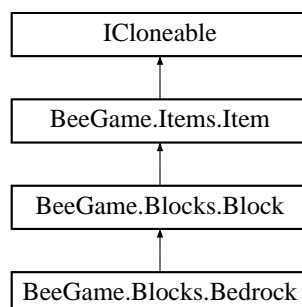
The documentation for this class was generated from the following file:

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

2.10 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

Additional Inherited Members

2.10.1 Detailed Description

[Bedrock Block](#)

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

2.10.2 Constructor & Destructor Documentation

2.10.2.1 [Bedrock\(\)](#)

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

Constructor

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

```
00018                                     : base("Bedrock")
00019     {
00020         breakable = false;
00021     }
```

2.10.3 Member Function Documentation

2.10.3.1 [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 29 of file [Bedrock.cs](#).

```
00030         {
00031             return;
00032         }
```

2.10.3.2 GetHashCode()

```
override int BeeGame.Blocks.Bedrock.GetHashCode ( )
```

Returns the ID of the item

Returns

-1

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

```
00053         {
00054             return -1;
00055         }
```

2.10.3.3 TexturePosition()

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

Position if te bedrock texture in the atlas

Parameters

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

Returns

Position in the texture atlas

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

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

```
00042      {  
00043          return new Tile() { x = 0, y = 0};  
00044      }
```

2.10.3.4 ToString()

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

The item name and ID as a string

Returns

A nicely formatted string

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

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

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

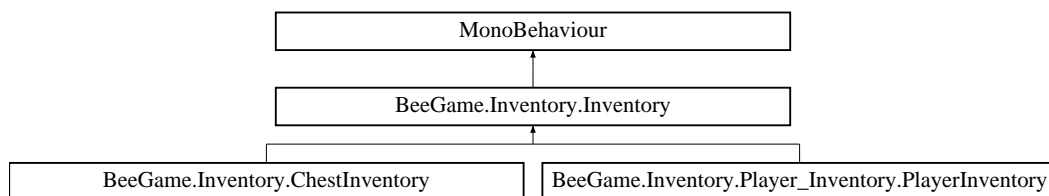
- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/[Bedrock.cs](#)

3 Inventory

3.1 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
- void [SaveInv](#) ()
Saves the inventory
- [ItemsInInventory](#) [GetAllItems](#) ()
Gets all of the items in the invntory
- 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

Protected Attributes

- bool [thisInventoryOpen](#) = false
is this inventory open?

Package Attributes

- [Item](#) floatingItem
Item that is currenty being moved

Private Member Functions

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

3.1.1 Detailed Description

Base class for all inventorys in the game

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

3.1.2 Member Function Documentation

3.1.2.1 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 131 of file [Inventory.cs](#).

```
00132     {
00133         return items.AddItem(item);
00134     }
```

3.1.2.2 AddItemToSlots()

```
void BeeGame.Inventory.Inventory.AddItemToSlots (
    int slotIndex,
    Item item )
```

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

Definition at line 119 of file [Inventory.cs](#).

```
00120     {
00121         items.AddItem(slotIndex, item);
00122         /* saves the inventory changes
00123         Serialization.Serialization.SerializeInventory(this, inventoryName);
00124     }
```

3.1.2.3 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 109 of file [Inventory.cs](#).

```
00110      {
00111          return items;
00112      }
```

3.1.2.4 InventorySet()

bool BeeGame.Inventory.Inventory.InventorySet ()

Is the inventory set?

Returns

true if [items](#) == null

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

```
00040      {
00041          if (items == null)
00042              return true;
00043          return false;
00044      }
```

3.1.2.5 PutItemsInSlots()

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

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

Definition at line 94 of file [Inventory.cs](#).

```
00095      {
00096          /** goes through all of the items in the array setting then all to a slot
00097          for (int i = 0; i < slots.Length; i++)
00098          {
00099              slots[i].slotIndex = i;
00100              slots[i].myInventory = this;
00101              slots[i].item = items.itemsInInventory[i];
00102          }
00103      }
```

3.1.2.6 SaveInv()

```
void BeeGame.Inventory.Inventory.SaveInv ( )
```

Saves the inventory

Used when closing a chest so the changes to the player inventory are saved

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

```
00087         {
00088             Serialization.Serialization.SerializeInventory(this, inventoryName);
00089         }
```

3.1.2.7 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 63 of file [Inventory.cs](#).

```
00064         {
00065             this.items = items;
00066         }
```

3.1.2.8 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 51 of file [Inventory.cs](#).

```
00052         {
00053             items = new ItemsInInventory(slots.Length);
00054         }
```

3.1.2.9 UpdateBase()

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

Things in the inventory that should be updated

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

```
00074         {  
00075             PutItemsInSlots ();  
00076         }
```

3.1.3 Member Data Documentation

3.1.3.1 floatingItem

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

Item that is currenty being moved

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

3.1.3.2 inventoryName

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

Name of this inventory

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

3.1.3.3 items

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

Items in the inventory

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

3.1.3.4 slots

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

Slots in the inventory

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

3.1.3.5 thisInventoryOpen

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

is this inventory open?

Definition at line 31 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](#)

3.2 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

3.2.1 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](#).

3.2.2 Constructor & Destructor Documentation

3.2.2.1 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         }
```

3.2.3 Member Function Documentation

3.2.3.1 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>	Where 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         }
```

3.2.3.2 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     }
00057     return false;
00058 }

```

3.2.4 Member Data Documentation

3.2.4.1 itemsInInventory

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

All of the items in the inventory

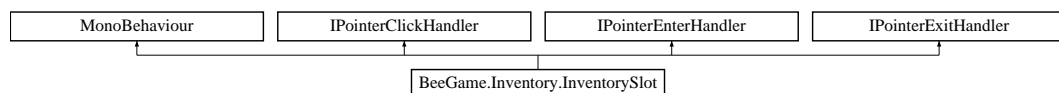
Definition at line 15 of file [ItemsInInventory.cs](#).

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

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

3.3 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?

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 AddToSlot \(int numerToAdd\)](#)
Adds a number to items into the slot
- [void SplitStack \(\)](#)
Halfs a [Item.itemStackCount](#) between the slot and the [Inventory.floatingItem](#)
- [void SwapItems \(\)](#)
Swaps the [Item](#) in the [Inventory.floatingItem](#) with the slots [item](#)
- [void CheckFloatingItem \(\)](#)
Checks if the [Inventory.floatingItem](#) should be null
- [void OnDisable \(\)](#)
Destroys the item text when the inventory is closed

3.3.1 Detailed Description

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

3.3.2 Member Function Documentation

3.3.2.1 AddToSlot()

```
void BeeGame.Inventory.InventorySlot.AddToSlot (  
    int numerToAdd ) [private]
```

Adds a number to items into the slot

Parameters

<i>numeToAdd</i>	Numebr or items to add to the slot
------------------	------------------------------------

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

```

00163     {
00164         /** if the item in the slot is null create it
00165         if (item == null)
00166         {
00167             item = myInventory.floatingItem.CloneObject();
00168             item.itemStackCount = 0;
00169         }
00170
00171         /** add to number to add to the stack count
00172         item.itemStackCount += numeToAdd;
00173
00174         /** if the stack count is now larger than it should be dont let it be
00175         if (item.itemStackCount > item.maxStackCount)
00176         {
00177             item.itemStackCount = item.maxStackCount;
00178         }
00179
00180         /** remove the numebr if items form the floating item then check the floating item is not null
00181         myInventory.floatingItem.itemStackCount -= numeToAdd;
00182         CheckFloatingItem();
00183         /** save the inventory changes
00184         myInventory.AddItemToSlots(slotIndex,
00185         item);
00185     }

```

3.3.2.2 CheckFloatingItem()

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

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

Definition at line 227 of file [InventorySlot.cs](#).

```

00228     {
00229         if(myInventory.floatingItem.itemStackCount <= 0)
00230         {
00231             myInventory.floatingItem = null;
00232         }
00233     }

```

3.3.2.3 OnDisable()

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

Destroys the item text when the inventory is closed

Definition at line 266 of file [InventorySlot.cs](#).

```

00267     {
00268         Destroy(itemText);
00269     }

```

3.3.2.4 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 87 of file [InventorySlot.cs](#).

```

00088     {
00089         if (myInventory.floatingItem != null)
00090         {
00091             /* Left click moves whole stacks if items
00092             if (eventData.button == PointerEventData.InputButton.Left)
00093             {
00094                 /* If the item in the slot is empty put the floating item into it then clear it
00095                 if (item == null)
00096                 {
00097                     item = myInventory.floatingItem;
00098                     myInventory.floatingItem = null;
00099                     myInventory.AddItemToSlots(
00100 slotIndex, item);
00101                     return;
00102                 }
00103                 /* if the items are the same
00104                 if(myInventory.floatingItem == item)
00105                 {
00106                     /* if the item in the inventoys stack count + the floating items stack count is
00107                     less than the max stack count
00108                     if (myInventory.floatingItem.
00109 itemStackCount + item.itemStackCount <= item.
00110 maxStackCount)
00111                     {
00112                         AddToSlot(myInventory.
00113 floatingItem.itemStackCount);
00114                         return;
00115                     }
00116                     /* if the item stack added is larger than the max count add as many as you can and
00117                     move on
00118                     else
00119                     {
00120                         AddToSlot(item.maxStackCount -
00121 item.itemStackCount);
00122                         return;
00123                     }
00124                 }
00125                 /* If the items were not == swap them
00126                 else
00127                 {
00128                     SwapItems();
00129                     return;
00130                 }
00131             }
00132             else if(eventData.button == PointerEventData.InputButton.Right)
00133             {
00134                 /* if the item in slot is null add 1 from the floating item to it
00135                 if(item == null)
00136                 {
00137                     AddToSlot(1);
00138                     return;
00139                 }
00140                 /* if the items are the same add 1 from the floating item to this item
00141                 else if(item == myInventory.floatingItem)
00142                 {
00143                     AddToSlot(1);
00144                     return;
00145                 }
00146             }
00147             /* if the floating item is null
00148             else
00149             {
00150                 /* add 1/2 of the stack into the floating item if right click was pressed
00151                 if(eventData.button == PointerEventData.InputButton.Right)
00152                 {
00153                     SplitStack();
00154                     return;
00155                 }
00156                 /* otherwie add the items into the floating item slot
00157                 SwapItems();
00158                 return;

```

```
00154         }
00155
00156     }
```

3.3.2.5 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 241 of file [InventorySlot.cs](#).

```
00242     {
00243         /* if the item is null or the floating item has something in it dont display the item text as
it is not necessary
00244         if (item != null && myInventory.floatingItem == null)
00245         {
00246             itemText = Instantiate(PrefabDictionary.
GetPrefab("ItemDetails"));
00247             /* sets the text to the correct postion
00248             itemText.transform.GetChild(0).position = Input.mousePosition;
00249             /* puts the correct text in the box
00250             itemText.transform.GetChild(0).GetChild(0).GetComponent<Text>().text = $"
{item.GetItemName()}\nStack: {item.itemStackCount}";
00251         }
00252     }
```

3.3.2.6 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 258 of file [InventorySlot.cs](#).

```
00259     {
00260         Destroy(itemText);
00261     }
```

3.3.2.7 SplitStack()

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

Halves a `Item.itemStackCount` between the slot and the `Inventory.floatingItem`

If the stack count is the slot is not an even number more items go to the floating item than go to the slot. This is so that right clicking on a slot when there is only 1 item in it actually make the item in that slot go into the floating item

Definition at line 193 of file `InventorySlot.cs`.

```
00194     {
00195         myInventory.floatingItem = item.CloneObject();
00196         int give = (item.itemStackCount + 1) / 2;
00197         myInventory.floatingItem.itemStackCount = give;
00198         item.itemStackCount -= give;
00199
00200         if (item.itemStackCount <= 0)
00201             item = null;
00202
00203         myInventory.AddItemToSlots(slotIndex,
00204             item);
00205         Destroy(itemText);
00205     }
```

3.3.2.8 SwapItems()

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

Swaps the Item in the `Inventory.floatingItem` with the slots `item`

Definition at line 210 of file `InventorySlot.cs`.

```
00211     {
00212         /* temp copy of the item
00213         Item temp = myInventory.floatingItem;
00214         /* sets the floating item
00215         myInventory.floatingItem = item;
00216         /* sets the item that was in the floating item to the item in the the slot
00217         item = temp;
00218         /* Saves the changes to the inventory
00219         myInventory.AddItemToSlots(slotIndex,
00220             item);
00220         /* destroys the text as it is not needed anymore
00221         Destroy(itemText);
00222     }
```

3.3.2.9 Update()

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

Updates the slot

Definition at line 37 of file `InventorySlot.cs`.

```
00038     {
00039         UpdateIcon();
00040     }
```


3.3.2.10 UpdateIcon()

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

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

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

```
00046     {
00047         if(item == null)
00048         {
00049             GetComponent<Image>().sprite = null;
00050         }
00051         else
00052         {
00053             GetComponent<Image>().sprite = item.GetItemSprite();
00054         }
00055
00056         /* if the slot is selected in the hotbar give the player some indication by colouring it grey
00057         if (selectedSlot)
00058         {
00059             GetComponent<Image>().color = Color.gray;
00060         }
00061         else
00062         {
00063             GetComponent<Image>().color = Color.white;
00064         }
00065
00066         /* sets the colour of the slot to the correct colour for the item
00067         /* make this easier then colouring many of the same sprite different colours
00068         if(item != null)
00069         {
00070             switch (item)
00071             {
00072                 case HoneyComb c:
00073                     GetComponent<Image>().color = c.CombColour;
00074                     break;
00075             }
00076         }
00077     }
```

3.3.3 Member Data Documentation

3.3.3.1 item

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

The item this slot has in it

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

3.3.3.2 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 27 of file [InventorySlot.cs](#).

3.3.3.3 myInventory

`Inventory` BeeGame.Inventory.InventorySlot.myInventory

The `Inventory` this slot is in

Definition at line 23 of file `InventorySlot.cs`.

3.3.3.4 selectedSlot

`bool` BeeGame.Inventory.InventorySlot.selectedSlot = false

Is this slot currently the selected slot in the hotbar?

Definition at line 31 of file `InventorySlot.cs`.

3.3.3.5 slotIndex

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

The slot in the inventory this is

Definition at line 15 of file `InventorySlot.cs`.

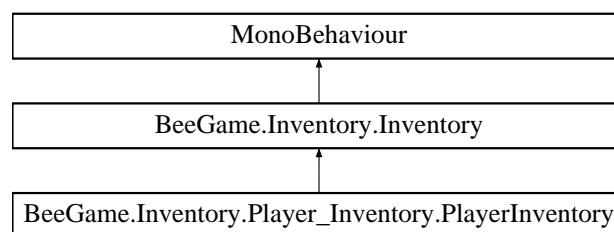
The documentation for this class was generated from the following file:

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

3.4 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

3.4.1 Detailed Description

Controlls the player inventory

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

3.4.2 Member Function Documentation

3.4.2.1 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     }
```

3.4.2.2 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     }

```

3.4.2.3 OpenPlayerInventory()

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

Show/Hide the player inventory

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

```

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

```

3.4.2.4 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 159 of file [PlayerInventory.cs](#).

```

00160         {
00161             item.item.itemStackCount = 1;
00162
00163             /* if the item can be added to the inventory do that
00164             if (AddItemToInventory(item.item))
00165             {
00166                 /* if the item was added destroy its gameobject and save the inventory
00167                 Destroy(item.gameObject);
00168                 Serialization.Serialization.SerializeInventory(this,
inventoryName);
00169             }
00170         }

```

3.4.2.5 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 140 of file [PlayerInventory.cs](#).

```

00141         {
00142             /* if the item is already null nothing needs to be removed
00143             if (GetAllItems().itemsInInventory[index] != null)
00144             {
00145                 /* remove 1 item and if that was the last in the stack remove the item from the inventory
00146                 GetAllItems().itemsInInventory[index].
itemStackCount -= 1;
00147
00148                 if (GetAllItems().itemsInInventory[index].itemStackCount <= 0)
00149                     GetAllItems().itemsInInventory[index] = null;
00150
00151                 Serialization.Serialization.SerializeInventory(this,
inventoryName);
00152             }
00153         }

```

3.4.2.6 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     }

```

3.4.2.7 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     }

```

3.4.2.8 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.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     }

```

3.4.3 Member Data Documentation

3.4.3.1 playerInventory

GameObject BeeGame.Inventory.Player_Inventory.PlayerInventory.playerInventory

Object that the inventory is

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

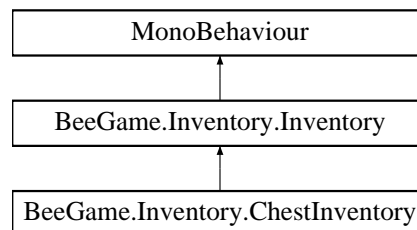
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/PlayerInventory/PlayerInventory.cs Inven-

3.5 BeeGame.Inventory.ChestInventory Class Reference

Incentory for the chests

Inheritance diagram for BeeGame.Inventory.ChestInventory:



Public Member Functions

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

3.5.1 Detailed Description

Inventory for the chests

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

3.5.2 Member Function Documentation

3.5.2.1 ApplyPlayerItems()

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

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

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

```
00081     {
00082         for (int i = 0; i < playerinventory.items.
00083             itemsInInventory.Length; i++)
00084         {
00085             playerinventory.items.itemsInInventory[i] =
00086             items.itemsInInventory[i + (inventorySize - 36)];
00087         }
00088         playerinventory.SaveInv();
00089     }
```

3.5.2.2 SetChestInventory()

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

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

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

```
00053     {
00054         SetInventorySize(inventorySize);
00055         /* sets the UI to not be seen as inventories cannot start open */
00056         inventory.SetActive(false);
00057
00058         /* sets the name and position if this inventory used during serialization and deserialization */
00059         inventoryName = $"Chest @ {(ChunkWorldPos) inventoryPosition}";
00060
00061         /* loads the inventory if it had had items put in it last time it existed */
00062         Serialization.Serializaton.DeSerializeInventory(this, inventoryName);
00063     }
```


3.5.2.3 SetPlayerItems()

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

Puts the player items into the chest

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

```
00070     {
00071         for (int i = 0; i < playerinventory.items.
itemsInInventory.Length; i++)
00072     {
00073         items.itemsInInventory[i + (inventorySize - 36)] =
playerinventory.items.itemsInInventory[i];
00074     }
00075 }
```

3.5.2.4 ToggleInventory()

```
void BeeGame.Inventory.ChestInventory.ToggleInventory (
    Inventory inv )
```

Opens and closes the inventory

Parameters

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

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

```
00096     {
00097         /* sets the player inventory
playerinventory = inv;
00098
00099         thisInventoryOpen = !thisInventoryOpen;
00100
00101         isAnotherInventoryOpen = thisInventoryOpen;
00102
00103         inventory.SetActive(!inventory.activeInHierarchy);
00104
00105         if (inventory.activeInHierarchy)
00106         {
00107             /* stops the player inventory from being opened immediately after this is closed
blockInventoryJustClosed = true;
00108             SetPlayerItems();
00109             /* hides and locks the cursor
Cursor.lockState = CursorLockMode.None;
00110             Cursor.visible = true;
00111
00112         }
00113         else
00114         {
00115             /* puts the items into the chest
/* shows and unlocks the cursor
00116             ApplyPlayerItems();
00117             Cursor.lockState = CursorLockMode.Locked;
00118             Cursor.visible = false;
00119
00120         }
00121     }
00122 }
00123 }
```

3.5.2.5 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         /** the chest should always have a player inventory when it does this but checks just in case
00040         if (playerinventory != null)
00041             UpdateBase();
00042
00043         /** checks if the inventory should be closed
00044         if (GetButtonDown("Player Inventory") && thisInventoryOpen)
00045             ToggleInventory(playerinventory);
00046     }
```

3.5.3 Member Data Documentation

3.5.3.1 inventory

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

The inventory gameobject that will be displayed

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

3.5.3.2 inventoryPosition

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

Position in worldspace of the chest

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

3.5.3.3 inventorySize

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

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

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

3.5.3.4 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:

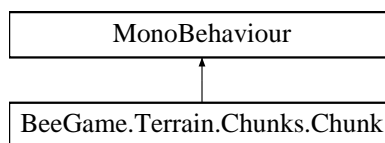
- `C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/ChestInventory.cs`

4 Chunk

4.1 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 colldier

4.1.1 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](#).

4.1.2 Member Function Documentation

4.1.2.1 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     }
```

4.1.2.2 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     }

```

4.1.2.3 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     }

```

4.1.2.4 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     }

```

4.1.2.5 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 their
00154             world.SetBlock(chunkWorldPos.x + x,
chunkWorldPos.y + y, chunkWorldPos.z + z, block);
00155     }

```

4.1.2.6 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     }
```

4.1.2.7 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     }
```

4.1.2.8 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     }
```


4.1.2.9 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     }
```

4.1.3 Member Data Documentation

4.1.3.1 applyCollisionMesh

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

Should the collision mesh be applied

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

4.1.3.2 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](#).

4.1.3.3 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](#).

4.1.3.4 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`.

4.1.3.5 filter

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

This `Chunks` mesh filter

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

4.1.3.6 mesh

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

`MeshData` of this chunk

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

4.1.3.7 meshCollider

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

This `Chunks` mesh collider

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

4.1.3.8 rendered

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

Is the `Chunk` rendered?

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

4.1.3.9 update

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

Should the [Chunk](#) be updated?

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

4.1.3.10 updateCollisionMesh

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

Should the chunks collision mesh be updated?

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

4.1.3.11 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](#)

4.2 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

4.2.1 Detailed Description

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

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

4.2.2 Member Function Documentation

4.2.2.1 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     }

```

4.2.2.2 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     }

```

4.2.2.3 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     }

```

4.2.3 Member Data Documentation

4.2.3.1 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](#).

4.2.3.2 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](#).

4.2.3.3 done

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

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

4.2.3.4 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](#).

4.2.3.5 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](#).

4.2.3.6 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](#).

4.2.3.7 verts

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

Verticies for the [Chunk](#) render Mesh

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

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

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

4.3 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](#)

4.3.1 Detailed Description

Serializable int version of THVector3

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

4.3.2 Constructor & Destructor Documentation

4.3.2.1 ChunkWorldPos()

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

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

Parameters

x	X Value
y	Y Value
z	Z Value

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

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

4.3.3 Member Function Documentation

4.3.3.1 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     }
```


4.3.3.2 GetHashCode()

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

Makes a unique hascode for the vector

Returns

unique int value for the vector

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

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

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

4.3.3.3 operator ChunkWorldPos()

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

Converts a [ChunkWorldPos](#) to a THVector3

Parameters

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

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

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

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

4.3.3.4 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         }
```

4.3.3.5 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         }
```

4.3.4 Member Data Documentation

4.3.4.1 x

```
int BeeGame.Terrain.ChunkWorldPos.x
```

x, y, z values for the vector

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

4.3.4.2 y

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

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

4.3.4.3 z

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

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

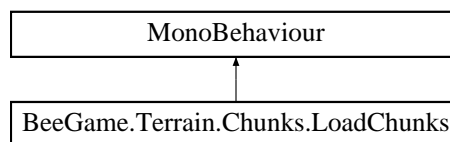
The documentation for this struct was generated from the following file:

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

4.4 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 [ApplyCollisionMeshToNearbyChunks](#) ()
Makes a collision 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

4.4.1 Detailed Description

Loads the [Chunks](#) around the player

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

4.4.2 Member Function Documentation

4.4.2.1 ApplyCollisionMeshToNearbyChunks()

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

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

We dont need to worry about removeing [Chunk](#) collision meshes as once PhysX has baked then they have minimal performance impact Doing things this way also spreads out the PhysX mesh baking

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

```
00112     {
00113         /** gets the player position in chunk coordinates
00114         ChunkWorldPos playerPos = new ChunkWorldPos(Mathf.FloorToInt(transform.position.x / Chunk.
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     }
```

4.4.2.2 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         }
```

4.4.2.3 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         }
```

4.4.2.4 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     }
```

4.4.2.5 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     }
```

4.4.2.6 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     }
```

4.4.2.7 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     }
```

4.4.3 Member Data Documentation

4.4.3.1 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](#).

4.4.3.2 chunkPositions

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

Positions to make chunks around the player ///

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

4.4.3.3 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](#).

4.4.3.4 timer

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

Timer for chunk removal

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

4.4.3.5 world

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

The world the player is in

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

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

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

4.5 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

4.5.1 Detailed Description

Saves a [Chunks](#) modified Blocks for save optimisation

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

4.5.2 Constructor & Destructor Documentation

4.5.2.1 SaveChunk()

```
BeeGame.Terrain.Chunks.SaveChunk.SaveChunk (
    Block blockArray[,,] )
```

Will search all the the given Blocks for modified blocks

Parameters

<i>blockArray</i>	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     }
```

4.5.3 Member Data Documentation

4.5.3.1 blocks

```
Dictionary<ChunkWorldPos, Block> BeeGame.Terrain.Chunks.SaveChunk.blocks = new Dictionary<ChunkWorldPos, Block>()
```

Blocks to be saved

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

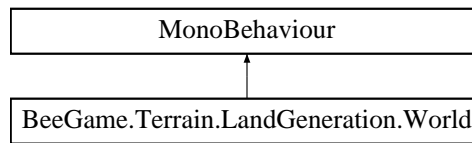
The documentation for this class was generated from the following file:

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

4.6 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

4.6.1 Detailed Description

Allows inter Chunk communication as it stores a list of active chunks

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

4.6.2 Member Function Documentation

4.6.2.1 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     }

```

4.6.2.2 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     }
```

4.6.2.3 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     }
```

4.6.2.4 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     }

```

4.6.2.5 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         }

```

4.6.2.6 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         }

```

4.6.3 Member Data Documentation

4.6.3.1 chunkHasMadeCollisionMesh

```
bool BeeGame.Terrain.LandGeneration.World.chunkHasMadeCollisionMesh = false
```

Has a Chunk made a collision mesh?

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

4.6.3.2 chunkPrefab

```
GameObject BeeGame.Terrain.LandGeneration.World.chunkPrefab
```

The chunk prefab

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

4.6.3.3 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](#)

4.7 BeeGame.Terrain.LandGeneration.Terrain Class Reference

Should use as an interface between the rest of the game and the terrain

Static Public Member Functions

- static [ChunkWorldPos GetBlockPos](#) ([THVector3](#) pos)
Gets a block position from a THVector3
- static [THVector3 GetBlockPos](#) ([RaycastHit](#) hit)
Returns the position of the block hit as a THVector3
- static [ChunkWorldPos GetBlockPosFromRayCast](#) ([RaycastHit](#) hit)
GetBlockPos(THVector3) does the same thing but returns a [ChunkWorldPos](#)
- static float [Round](#) (float pos, float norm, bool adjacent=false)
Rounds the given pos to the correct position
- static [ChunkWorldPos GetBlockPos](#) ([RaycastHit](#) hit, bool adjacent=false)
Gets a Chunks world position
- static [Block GetBlock](#) ([RaycastHit](#) hit, bool adjacent=false)
Get a Block at the given position
- static [Block GetBlock](#) ([THVector3](#) pos)
- static bool [BlockInPosition](#) ([THVector3](#) pos, [Chunk](#) chunk)
- static [Chunk GetChunk](#) ([THVector3](#) vec3)
- static bool [SetBlock](#) ([RaycastHit](#) hit, [Block](#) block, bool adjacent=false)
Sets the Block at the given point the given Block

Static Public Attributes

- static [World world](#)

Static Private Member Functions

- static float [RoundXZ](#) (float pos, float normal)
Used to round the X/Z values when getting a block
- static float [RoundY](#) (float pos, float normal)
Round the Y value of the given coord

4.7.1 Detailed Description

Should use as an interface between the rest of the game and the terrain

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

4.7.2 Member Function Documentation

4.7.2.1 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
Air())
00253             return true;
00254
00255         return false;
00256     }
```

4.7.2.2 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     }
```

4.7.2.3 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     }
```

4.7.2.4 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 }
```

4.7.2.5 GetBlockPos() [2/3]

```
static THVector3 BeeGame.Terrain.LandGeneration.Terrain.GetBlockPos (
    RaycastHit hit ) [static]
```

Returns the position 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 coordinates

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 }
```

4.7.2.6 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 }
```

4.7.2.7 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     }
```

4.7.2.8 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     }
```

4.7.2.9 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 recived

Returns

rounded value of *pos* as a float

Check how this performs. Possibly change all uses of this to [RoundXZ\(float, float\)](#) and [RoundY\(float, float\)](#)

Definition at line 179 of file [Terrain.cs](#).

```

00180     {
00181         if(pos - (int)pos == 0.5f || pos - (int)pos == -0.5f)
00182         {
00183             if(adjacent)
00184             {
00185                 pos += (norm / 2);
00186             }
00187             else
00188             {
00189                 pos -= (norm / 2);
00190             }
00191         }
00192
00193         return pos;
00194     }

```

4.7.2.10 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 }

```

4.7.2.11 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
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     }

```

4.7.2.12 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     }

```

4.7.3 Member Data Documentation

4.7.3.1 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](#)↔

4.8 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

4.8.1 Detailed Description

Generates the terrain for the game

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

4.8.2 Member Function Documentation

4.8.2.1 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         }

```

4.8.2.2 ChunkGenThread()

```

void BeeGame.Terrain.LandGeneration.TerrainGeneration.ChunkGenThread (
    Chunk chunk,
    out Chunk outChunk )

```

Generates a new Chunk

Parameters

<i>chunk</i>	Chunk to be generated
<i>outChunk</i>	Generated Chunk to return

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

```

00097         {
00098             /* for each x and z position in teh chunk
00099             for (int x = chunk.chunkWorldPos.x-3; x < chunk.
chunkWorldPos.x + Chunk.chunkSize + 3; x++)
00100             {
00101                 for (int z = chunk.chunkWorldPos.z-3; z < chunk.
chunkWorldPos.z + Chunk.chunkSize + 3; z++)
00102                 {
00103                     chunk = GenChunkColum(chunk, x, z);
00104                 }
00105             }
00106
00107             chunk.SetBlocksUnmodified();
00108             outChunk = chunk;
00109         }

```

4.8.2.3 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     }

```

4.8.2.4 GenChunkColumn()

```

Chunk BeeGame.Terrain.LandGeneration.TerrainGeneration.GenChunkColumn (
    Chunk chunk,
    int x,
    int z )

```

Generates a colum of the Chunk

Parameters

<i>chunk</i>	Chunk to generate a colum for
<i>x</i>	X pos to make the colum
<i>z</i>	Z pos to make the colum

Returns

Chunk with a new colum ob blocks generated

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

```

00119     {
00120         /** the height of the mountain
00121         int stoneHeight = Mathf.FloorToInt(stoneBaseHeight);
00122         stoneHeight += GetNoise(-x, 0, z, stoneMountainFrequency, Mathf.

```

```

    FloorToInt(stoneMountainHeight));
00123
00124     /* if the colum is currently to low make it not so low
00125     if (stoneHeight < stoneMinHeight)
00126         stoneHeight = Mathf.FloorToInt(stoneMinHeight);
00127
00128     /* add the height of normal stone on to the mountain
00129     stoneHeight += GetNoise(x, 0, -z, stoneBaseNoise, Mathf.RoundToInt(
stoneBaseNoiseHeight));
00130
00131     /*put dirt on top
00132     int dirtHeight = stoneHeight + Mathf.FloorToInt(dirtBaseHeight);
00133     dirtHeight += GetNoise(x, 100, z, dirtNoise, Mathf.FloorToInt(
dirtNoiseHeight));
00134
00135     /* set the colum to the correct blocks
00136     for (int y = chunk.chunkWorldPos.y - 8; y < chunk.
chunkWorldPos.y + Chunk.chunkSize; y++)
00137     {
00138         int caveChance = GetNoise(x + 40, y + 100, z - 50,
caveFrequency, 200);
00139
00140         /* puts a layer of bedrock at the 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)
                CreateTree(x, y + 1, z, chunk);
00153         }
00154         else
00155         {
00156             SetBlock(x, y, z, new Blocks.Air(), chunk);
00157         }
00158     }
00159 }
00160
00161     return chunk;
00162 }

```

4.8.2.5 GetNoise()

```

static int BeeGame.Terrain.LandGeneration.TerrainGeneration.GetNoise (
    int x,
    int y,
    int z,
    float scale,
    int max ) [static]

```

Get a noise value

Parameters

<i>x</i>	X pos of the noise
<i>y</i>	Y pos of the noise
<i>z</i>	Z pos of the noise
<i>scale</i>	What the step shout bee from the last x, y, z
<i>max</i>	Max value of the noise

Returns

A noise value as an int

Definition at line 173 of file [TerrainGeneration.cs](#).

```

00174         {
00175             return Mathf.FloorToInt((SimplexNoise.Generate(x * scale, y * scale, z *
00176                 scale) + 1f) * (max / 2f));
        }

```

4.8.2.6 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) &&
00196                 Chunk.InRange(z))
00197                 if (replacesBlocks || chunk.blocks[x, y, z] == null)
00198                     chunk.SetBlock(x, y, z, block, false);
        }

```

4.8.3 Member Data Documentation

4.8.3.1 caveFrequency

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.caveFrequency = 0.025f [private]
```

How often do caves happen

Definition at line 67 of file [TerrainGeneration.cs](#).

4.8.3.2 caveSize

```
int BeeGame.Terrain.LandGeneration.TerrainGeneration.caveSize = 8 [private]
```

Threshold for making a cave

Definition at line 71 of file [TerrainGeneration.cs](#).

4.8.3.3 dirtBaseHeight

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.dirtBaseHeight = 1 [private]
```

Where does dirt start

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

4.8.3.4 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](#).

4.8.3.5 dirtNoiseHeight

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.dirtNoiseHeight = 3 [private]
```

How tall dirt can be

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

4.8.3.6 stoneBaseHeight

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.stoneBaseHeight = -24 [private]
```

Base height of stone

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

4.8.3.7 stoneBaseNoise

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.stoneBaseNoise = 0.05f [private]
```

Base noise of stone

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

4.8.3.8 stoneBaseNoiseHeight

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.stoneBaseNoiseHeight = 4 [private]
```

Base noise heigh for stone

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

4.8.3.9 stoneMinHeight

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.stoneMinHeight = -12 [private]
```

Minimum height for stone

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

4.8.3.10 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](#).

4.8.3.11 stoneMountainHeight

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.stoneMountainHeight = 48 [private]
```

Base height for a mountain

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

4.8.3.12 treeDensity

```
int BeeGame.Terrain.LandGeneration.TerrainGeneration.treeDensity = 3 [private]
```

Desity of trees

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

4.8.3.13 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](#)

4.9 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)

4.9.1 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](#).

4.9.2 Member Function Documentation

4.9.2.1 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     }
```

4.9.2.2 Generate() [1/3]

```
static float BeeGame.Terrain.LandGeneration.Noise.SimplexNoise.Generate (
    float x ) [static]
```

1D simplex noise

Parameters

x	
---	--

Returns

Definition at line 44 of file [SimplexNoise.cs](#).

```

00045     {
00046         int i0 = FastFloor(x);
00047         int i1 = i0 + 1;
00048         float x0 = x - i0;
00049         float x1 = x0 - 1.0f;
00050
00051         float n0, n1;
00052
00053         float t0 = 1.0f - x0 * x0;
00054         t0 *= t0;
00055         n0 = t0 * t0 * grad(perm[i0 & 0xff], x0);
00056
00057         float t1 = 1.0f - x1 * x1;
00058         t1 *= t1;
00059         n1 = t1 * t1 * grad(perm[i1 & 0xff], x1);
00060         /** The maximum value of this noise is 8*(3/4)^4 = 2.53125
00061         /** A factor of 0.395 scales to fit exactly within [-1,1]
00062         return 0.395f * (n0 + n1);
00063     }

```

4.9.2.3 Generate() [2/3]

```

static float BeeGame.Terrain.LandGeneration.Noise.SimplexNoise.Generate (
    float x,
    float y ) [static]

```

2D simplex noise

Parameters

x	
y	

Returns

Definition at line 71 of file [SimplexNoise.cs](#).

```

00072     {
00073         const float F2 = 0.366025403f; /** F2 = 0.5*(sqrt(3.0)-1.0)
00074         const float G2 = 0.211324865f; /** G2 = (3.0-Math.sqrt(3.0))/6.0
00075
00076         float n0, n1, n2; /** Noise contributions from the three corners
00077
00078         /** Skew the input space to determine which simplex cell we're in
00079         float s = (x + y) * F2; /** Hairy factor for 2D
00080         float xs = x + s;
00081         float ys = y + s;
00082         int i = FastFloor(xs);
00083         int j = FastFloor(ys);
00084
00085         float t = (float)(i + j) * G2;
00086         float X0 = i - t; /** Unskew the cell origin back to (x,y) space
00087         float Y0 = j - t;
00088         float x0 = x - X0; /** The x,y distances from the cell origin
00089         float y0 = y - Y0;
00090
00091         /** For the 2D case, the simplex shape is an equilateral triangle.
00092         /** Determine which simplex we are in.
00093         int i1, j1; /** Offsets for second (middle) corner of simplex in (i,j) coords
00094         if (x0 > y0) { i1 = 1; j1 = 0; } /** lower triangle, XY order: (0,0)->(1,0)->(1,1)
00095         else { i1 = 0; j1 = 1; } /** upper triangle, YX order: (0,0)->(0,1)->(1,1)
00096
00097         /** A step of (1,0) in (i,j) means a step of (1-c,-c) in (x,y), and
00098         /** a step of (0,1) in (i,j) means a step of (-c,1-c) in (x,y), where
00099         /** c = (3-sqrt(3))/6

```

```

00100
00101     float x1 = x0 - i1 + G2; /* Offsets for middle corner in (x,y) unskewed coords
00102     float y1 = y0 - j1 + G2;
00103     float x2 = x0 - 1.0f + 2.0f * G2; /* Offsets for last corner in (x,y) unskewed coords
00104     float y2 = y0 - 1.0f + 2.0f * G2;
00105
00106     /* Wrap the integer indices at 256, to avoid indexing perm[] out of bounds
00107     int ii = i % 256;
00108     int jj = j % 256;
00109
00110     /* Calculate the contribution from the three corners
00111     float t0 = 0.5f - x0 * x0 - y0 * y0;
00112     if (t0 < 0.0f) n0 = 0.0f;
00113     else
00114     {
00115         t0 *= t0;
00116         n0 = t0 * t0 * grad(perm[ii + perm[jj]], x0, y0);
00117     }
00118
00119     float t1 = 0.5f - x1 * x1 - y1 * y1;
00120     if (t1 < 0.0f) n1 = 0.0f;
00121     else
00122     {
00123         t1 *= t1;
00124         n1 = t1 * t1 * grad(perm[ii + 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 }

```

4.9.2.4 Generate() [3/3]

```

static float BeeGame.Terrain.LandGeneration.Noise.SimplexNoise.Generate (
    float x,
    float y,
    float z ) [static]

```

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

```

00142 {
00143     /* Simple skewing factors for the 3D case
00144     const float F3 = 0.333333333f;
00145     const float G3 = 0.1666666667f;
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     }

```

4.9.25 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 }
```

4.9.2.6 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 }
```

4.9.2.7 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 }
```

4.9.2.8 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     }
```

4.9.2.9 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     }
```

4.9.3 Member Data Documentation

4.9.3.1 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:

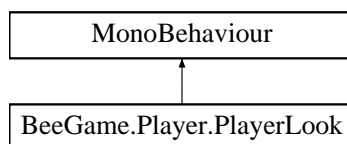
- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/LandGeneration/↔ Noise/[SimplexNoise.cs](#)

5 Player

5.1 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

5.1.1 Detailed Description

The look for the player

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

5.1.2 Member Function Documentation

5.1.2.1 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     }
```

5.1.2.2 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     }
```

5.1.2.3 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 will not update when a inventory GUI is open
00055         if (!THInput.isAnotherInventoryOpen)
00056         {
00057             Look();
00058         }
00059     }
```


5.1.3 Member Data Documentation

5.1.3.1 cameraTransform

`Transform BeeGame.Player.PlayerLook.cameraTransform`

Camera transform

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

5.1.3.2 myTransform

`Transform BeeGame.Player.PlayerLook.myTransform`

[Player](#) transform

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

5.1.3.3 rotationLock

`float BeeGame.Player.PlayerLook.rotationLock`

Lock for camera X rotation

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

5.1.3.4 speed

`float BeeGame.Player.PlayerLook.speed = 5`

Look move speed

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

5.1.3.5 xRot

`float BeeGame.Player.PlayerLook.xRot = 0 [private]`

Current X rotation

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

5.1.3.6 yRot

```
float BeeGame.Player.PlayerLook.yRot = 0 [private]
```

Current Y rotation

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

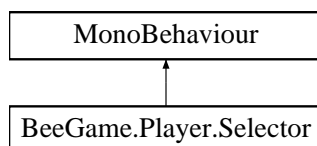
The documentation for this class was generated from the following file:

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

5.2 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

5.2.1 Detailed Description

Moves the Block selector

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

5.2.2 Member Function Documentation

5.2.2.1 Awake()

```
void BeeGame.Player.Selector.Awake ( ) [private]
```

Make the selector

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

```
00048         {
00049             selector = Instantiate(selector);
00050         }
```

5.2.2.2 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         }
```

5.2.2.3 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         }
```

5.2.2.4 PlaceBlock()

```
void BeeGame.Player.Selector.PlaceBlock ( ) [private]
```

Places s Block in the selector position

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

```
00142         {
00143             Chunk chunk = GetChunk(selector.transform.position);
00144
00145             if (chunk == null)
00146                 return;
00147
00148             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         }
```

5.2.2.5 SelectedSlot()

```
void BeeGame.Player.Selector.SelectedSlot ( ) [private]
```

Chanages 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
it to the last slot in the hotbar
00108             else if (Input.GetAxis("Mouse ScrollWheel") < 0)
00109             {
00110                 selectedHotbarSlot -= 1;
00111                 if (selectedHotbarSlot == 26)
00112                     selectedHotbarSlot = 35;
00113             }
00114
00115             transform.parent.GetComponentInChildren<PlayerInventory>().
SelectedSlot(selectedHotbarSlot);
00116         }
```

5.2.2.6 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     }
```

5.2.2.7 UpdateSelector()

```
void BeeGame.Player.Selector.UpdateSelector ( ) [private]
```

Updates teh selectors position

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

```
00081     {
00082         if (Physics.Raycast(transform.position, transform.forward, out hit, 15,
00083             layers))
00084         {
00085             selector.SetActive(true);
00086             selector.transform.position = GetBlockPos(hit);
00087             /*selector.SetActive(BlockInPosition(GetBlockPos(hit),
00088             hit.collider.GetComponent<Chunk>()));
00089         }
00090         else
00091         {
00092             selector.SetActive(false);
00093             SelectedSlot();
00094         }
```

5.2.3 Member Data Documentation

5.2.3.1 hit

```
RaycastHit BeeGame.Player.Selector.hit [private]
```

Where the raycast hit

Definition at line 35 of file [Selector.cs](#).

5.2.3.2 layers

```
LayerMask BeeGame.Player.Selector.layers
```

Layers for the selector to look at

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

5.2.3.3 playerInventory

```
PlayerInventory BeeGame.Player.Selector.playerInventory
```

[Player Inventory](#)

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

5.2.3.4 selectedHotbarSlot

```
int BeeGame.Player.Selector.selectedHotbarSlot = 27
```

What slot in the hotbar is selected

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

5.2.3.5 selector

```
GameObject BeeGame.Player.Selector.selector
```

[Selector](#)

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

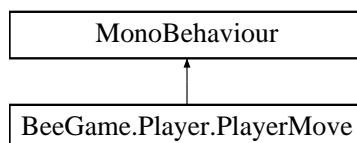
The documentation for this class was generated from the following file:

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

5.3 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

5.3.1 Detailed Description

Moves the player

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

5.3.2 Member Function Documentation

5.3.2.1 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     }
```

5.3.2.2 FixedUpdate()

```
void BeeGame.Player.PlayerMove.FixedUpdate ( ) [private]
```

Updates the player move

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

```
00062     {
00063         //If the player is grounded it can move
00064         if (canJump)
00065         {
00066             MovePlayer();
00067         }
00068
00069         //adds the downward force
00070         myRigidBody.AddForce(new Vector3(0, myRigidBody.mass * -
00071 gravity, 0));
00071     }
```

5.3.2.3 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 infinatly 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     }
```


5.3.2.4 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     }
```

5.3.2.5 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     }
```

5.3.3 Member Data Documentation

5.3.3.1 canJump

```
bool BeeGame.Player.PlayerMove.canJump = false [private]
```

Can the player jump?

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

5.3.3.2 gravity

```
float BeeGame.Player.PlayerMove.gravity = 9.81f
```

Gravity of the player

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

5.3.3.3 jumpHeight

```
float BeeGame.Player.PlayerMove.jumpHeight = 2f
```

How high can the player jump

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

5.3.3.4 maxVelocity

```
float BeeGame.Player.PlayerMove.maxVelocity = 10f
```

Max velocity of the player

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

5.3.3.5 myRigidBody

```
Rigidbody BeeGame.Player.PlayerMove.myRigidBody [private]
```

Rigidbody for the player

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

5.3.3.6 speed

```
float BeeGame.Player.PlayerMove.speed = 10f
```

Speed of the player

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

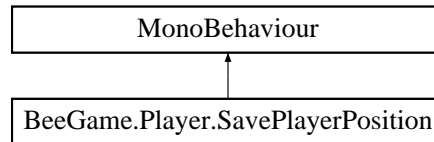
The documentation for this class was generated from the following file:

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

5.4 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

5.4.1 Detailed Description

Saves the player postion

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

5.4.2 Member Function Documentation

5.4.2.1 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     }
```

5.4.3 Member Data Documentation

5.4.3.1 counter

```
int BeeGame.Player.SavePlayerPosition.counter = 0 [private]
```

Timer for saveing the player

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

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

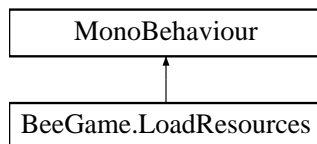
- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/[SavePlayerPosition.cs](#)

6 Resources

6.1 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

6.1.1 Detailed Description

Loads all of the resources in the game

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

6.1.2 Member Function Documentation

6.1.2.1 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](#)

6.2 BeeGame.Core.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

6.2.1 Detailed Description

The prefabs available to the game

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

6.2.2 Member Function Documentation

6.2.2.1 GetPrefab()

```
static GameObject BeeGame.Core.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         }
```

6.2.2.2 LoadPrefabs()

```
static void BeeGame.Core.PrefabDictionary.LoadPrefabs ( ) [static]
```

Loads the prefabs into the Dictionary

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

```
00020         {
00021             prefabDictionary = Resources.Resources.GetPrefabs();
00022         }
```

6.2.3 Member Data Documentation**6.2.3.1 prefabDictionary**

```
Dictionary<string, GameObject> BeeGame.Core.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](#)

6.3 BeeGame.Core.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

6.3.1 Detailed Description

All of the sprites available to the game

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

6.3.2 Member Function Documentation

6.3.2.1 GetSprite()

```
static Sprite BeeGame.Core.SpriteDictionary.GetSprite (
    string spriteName ) [static]
```

Get a sprite of the given name

Parameters

<i>spriteName</i>	Name of sprite to get
-------------------	-----------------------

Returns

A sprite of the given name, null if no sprite of that name exists

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

```
00022     {
00023         itemSpriteDictionary.TryGetValue(spriteName, out Sprite sprite);
00024
00025         if (sprite == null)
00026             return new Sprite();
00027
00028         return sprite;
00029     }
```

6.3.2.2 LoadSprites()

```
static void BeeGame.Core.SpriteDictionary.LoadSprites ( ) [static]
```

Loads the sprites into the dictionary

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

```
00035         {
00036             itemSpriteDictionary = Resources.Resources.GetSprites();
00037         }
```

6.3.3 Member Data Documentation

6.3.3.1 itemSpriteDictionary

```
Dictionary<string, Sprite> BeeGame.Core.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](#)

6.4 BeeGame.Core.BeeDictionarys Class Reference

Static Public Member Functions

- static Color [GetCombColour](#) ([HoneyCombType](#) type)
Returns colour if the given honey coumb

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 , optional an Alpha, a . Rangeing from 0f-255f

Static Private Attributes

- static Dictionary< [HoneyCombType](#), Color > [honeyCoumbColour](#)
The colour of the [BeeGame.Items.HoneyComb](#) for each of teh HoneyCombTypes

6.4.1 Detailed Description

Definition at line 7 of file [BeeDictionarys.cs](#).

6.4.2 Member Function Documentation

6.4.2.1 CombColour()

```
static Color BeeGame.Core.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 28 of file [BeeDictionarys.cs](#).

```
00029         {
00030             return new Color(r / 255f, g / 255f, b / 255f);
00031         }
```

6.4.2.2 GetCombColour()

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

Returns colour if the given honey coumb

Parameters

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

Returns

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

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

```

00039         {
00040             honeyCoubColour.TryGetValue(type, out var temp);
00041
00042             if (temp == null)
00043                 return new Color(1, 0, 0);
00044
00045             return temp;
00046         }

```

6.4.3 Member Data Documentation**6.4.3.1 honeyCoubColour**

Dictionary<[HoneyCombType](#), Color> BeeGame.Core.BeeDictionary.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 13 of file [BeeDictionary.cs](#).

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

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

6.5 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](#)

6.5.1 Detailed Description

A strongly-typed resource class, for looking up localized strings, etc.

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

6.5.2 Constructor & Destructor Documentation

6.5.2.1 Resources()

BeeGame.Resources.Resources.Resources () [package]

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

```
00033                                     {  
00034     }
```

6.5.3 Member Function Documentation

6.5.3.1 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         }
```

6.5.3.2 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         }
```

```

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     }

```

6.5.4 Member Data Documentation

6.5.4.1 resourceCulture

```
global.System.Globalization.CultureInfo BeeGame.Resources.Resources.resourceCulture [static],
[private]
```

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

6.5.4.2 resourceMan

```
global.System.Resources.ResourceManager BeeGame.Resources.Resources.resourceMan [static],
[private]
```

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

6.5.5 Property Documentation

6.5.5.1 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](#).

6.5.5.2 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](#).

6.5.5.3 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](#).

6.5.5.4 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](#)

7 Unity Type & Method Replacements

7.1 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 Private Attributes

- static Dictionary< string, object > [inputButtons](#)
Button identifiers and KeyCode

7.1.1 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 10 of file [THInput.cs](#).

7.1.2 Member Function Documentation

7.1.2.1.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 135 of file [THInput.cs](#).

```
00136     {
00137         int returnAxis = 0;
00138
00139         if (axis == "Horizontal")
00140         {
00141             if (GetButton("Right"))
00142             {
00143                 returnAxis += 1;
00144             }
00145             if (GetButton("Left"))
00146             {
00147                 returnAxis -= 1;
00148             }
00149         }
00150         else if (axis == "Vertical")
00151         {
00152             if (GetButton("Forward"))
00153             {
00154                 returnAxis += 1;
00155             }
00156             if (GetButton("Backward"))
00157             {
00158                 returnAxis -= 1;
00159             }
00160         }
00161         return returnAxis;
00162     }
00163
00164
00165 }
```

7.1.2.2 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 75 of file [THInput.cs](#).

```
00076         {
00077             if (!inputButtons.ContainsKey(button))
00078             {
00079                 throw new Exception("Input Manager: Key button name not defined: " + button);
00080             }
00081
00082             switch (inputButtons[button])
00083             {
00084                 case KeyCode[] array:
00085                     /*for each possible key, check if it was pressed and if it was return that it was, if
00086                     none of them was pressed return false
00087                     foreach (var item in array)
00088                     {
00089                         if (Input.GetKey(item))
00090                         {
00091                             return true;
00092                         }
00093                     }
00094                     return false;
00095                 default:
00096                     return Input.GetKey((KeyCode)inputButtons[button]);
00097             }
00098         }
```

7.1.2.3 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 45 of file [THInput.cs](#).

```

00046         {
00047             if (!inputButtons.ContainsKey(button))
00048             {
00049                 throw new Exception("Input Manager: Key button name not defined: " + button);
00050             }
00051
00052             switch (inputButtons[button])
00053             {
00054                 case KeyCode[] array:
00055                     /*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
00056                     foreach (var item in array)
00057                     {
00058                         if (Input.GetKeyDown(item))
00059                         {
00060                             return true;
00061                         }
00062                     }
00063
00064                     return false;
00065                 default:
00066                     return Input.GetKeyDown((KeyCode)inputButtons[button]);
00067             }
00068         }

```

7.1.2.4 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 105 of file [THInput.cs](#).

```

00106         {
00107             if (!inputButtons.ContainsKey(button))
00108             {
00109                 throw new Exception("Input Manager: Key button name not defined: " + button);
00110             }
00111
00112             switch (inputButtons[button])
00113             {
00114                 case KeyCode[] array:
00115                     /*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
00116                     foreach (var item in array)
00117                     {
00118                         if (Input.GetKeyUp(item))
00119                         {
00120                             return true;

```

```

00121         }
00122     }
00123
00124     return false;
00125     default:
00126         return Input.GetKeyUp((KeyCode) inputButtons[button]);
00127     }
00128 }

```

7.1.3 Member Data Documentation

7.1.3.1 blockInventoryJustClosed

`bool BeeGame.Core.THInput.blockInventoryJustClosed [static]`

Was a Block inventory just closed

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

7.1.3.2 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 15 of file [THInput.cs](#).

7.1.3.3 isAnotherInventoryOpen

`bool BeeGame.Core.THInput.isAnotherInventoryOpen [static]`

If another inventory is open true, else false

Definition at line 33 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](#)

7.2 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

7.2.1 Detailed Description

Serializable version of Vector2

Definition at line 10 of file [THVector2.cs](#).

7.2.2 Constructor & Destructor Documentation

7.2.2.1 THVector2() [1/3]

```
BeeGame.Core.THVector2.THVector2 (
    float x,
    float y )
```

Constructor from 2 floats

Parameters

<i>x</i>	X position
<i>y</i>	Y position

Definition at line 29 of file [THVector2.cs](#).

```
00030     {
00031         this.x = x;
00032         this.y = y;
00033     }
```

7.2.2.2 THVector2() [2/3]

```
BeeGame.Core.THVector2.THVector2 (
    THVector2 vec2 )
```

Constructor from another [THVector2](#)

Parameters

<i>vec2</i>	Vector to make this from
-------------	--------------------------

Definition at line 39 of file [THVector2.cs](#).

```
00040     {
00041         this = vec2;
00042     }
```

7.2.2.3 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         }
```

7.2.3 Member Function Documentation

7.2.3.1 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         }
```

7.2.3.2 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         }
```

7.2.3.3 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     }
```

7.2.3.4 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     }
```

7.2.3.5 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     }
```

7.2.3.6 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     }
```

7.2.3.7 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     }
```

7.2.3.8 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     }
```

7.2.3.9 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     }
```

7.2.3.10 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     }
```

7.2.3.11 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     }
```

7.2.3.12 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     }
```

7.2.3.13 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     }
```

7.2.3.14 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     }
```


7.2.3.15 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     }
```

7.2.3.16 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     }
```

7.2.3.17 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     }
```

7.2.3.18 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     }
```

7.2.3.19 ToString()

```
override string BeeGame.Core.THVector2.ToString ( )
```

Definition at line 77 of file [THVector2.cs](#).

```
00078         {
00079             return $"{x}, {y}";
00080         }
```

7.2.4 Member Data Documentation

7.2.4.1 x

```
float BeeGame.Core.THVector2.x
```

X position

Definition at line 16 of file [THVector2.cs](#).

7.2.4.2 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](#)

7.3 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

7.3.1 Detailed Description

Serializable version of Vector3

Definition at line 10 of file [THVector3.cs](#).

7.3.2 Constructor & Destructor Documentation

7.3.2.1 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     }
```

7.3.2.2 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     }
```

7.3.2.3 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         }
```

7.3.2.4 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         }
```

7.3.3 Member Function Documentation

7.3.3.1 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));
    }
```

7.3.3.2 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     }
```

7.3.3.3 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     }
```

7.3.3.4 operator Quaternion()

```
static BeeGame.Core.THVector3.operator Quaternion (
    THVector3 vec3 ) [explicit], [static]
```

Converts a [THVector3](#) to a Quaternion

Parameters

<code>vec3</code>	Vector to convert to Quaternion
-------------------	---------------------------------

Explicit as conversion is not exact

Definition at line 327 of file [THVector3.cs](#).

```
00328     {
00329         return new Quaternion(vec3.x, vec3.y, vec3.z, 0);
00330     }
```

7.3.3.5 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     }
```

7.3.3.6 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         }
```

7.3.3.7 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         }
```

7.3.3.8 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     }

```

7.3.3.9 operator*() [2/3]

```

static THVector3 BeeGame.Core.THVector3.operator* (
    THVector3 a,
    float b ) [static]

```

Multiples 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     }

```

7.3.3.10 operator*() [3/3]

```

static THVector3 BeeGame.Core.THVector3.operator* (
    float a,
    THVector3 b ) [static]

```

Multiples 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     }
```

7.3.3.11 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     }
```

7.3.3.12 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     }
```

7.3.3.13 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     }
```

7.3.3.14 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
00195         return a;
00196     }
```

7.3.3.15 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     }
```

7.3.3.16 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     }
```

7.3.3.17 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     }
```

7.3.3.18 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     }
```

7.3.3.19 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     }
```

7.3.3.20 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     }
```

7.3.3.21 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     }
```

7.3.4 Member Data Documentation**7.3.4.1 x**

```
float BeeGame.Core.THVector3.x
```

X position

Definition at line 16 of file [THVector3.cs](#).

7.3.4.2 y

```
float BeeGame.Core.THVector3.y
```

Y postion

Definition at line 20 of file [THVector3.cs](#).

7.3.4.3 z

```
float BeeGame.Core.THVector3.z
```

Z position

Definition at line 24 of file [THVector3.cs](#).

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

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityTypeReplacements/[T↔HVector3.cs](#)

8 Misc

8.1 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

8.1.1 Detailed Description

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

8.1.2 Member Function Documentation

8.1.2.1 [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 copied

Mush faster than the serialize method however alot more complicated

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


```

00020     {
00021         /** gets the tyoe of the given object
00022         Type typeSource = obj.GetType();
00023
00024         /** makes a new object of type T
00025         T objTarget = (T)Activator.CreateInstance(typeSource);
00026
00027         /** gets the properties in T
00028         PropertyInfo[] propertyInfo = typeSource.GetProperties(BindingFlags.Public | BindingFlags.
NonPublic | BindingFlags.Instance);
00029
00030         /** applies the properties in T to the new type T object
00031         foreach (var property in propertyInfo)
00032         {
00033             if (property.CanWrite)
00034             {
00035                 /** if the property is a value just set it
00036                 if (property.PropertyType.IsValueType || property.PropertyType.IsEnum || property.
PropertyType.Equals(typeof(string)))
00037                 {
00038                     property.SetValue(objTarget, property.GetValue(obj, null), null);
00039                 }
00040                 else
00041                 {
00042                     /** 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
00043                     object propertyValue = property.GetValue(obj, null);
00044
00045                     if (propertyValue == null)
00046                     {
00047                         property.SetValue(obj, null, null);
00048                     }
00049                     else
00050                     {
00051                         property.SetValue(obj, propertyValue.CloneObject(), null);
00052                     }
00053                 }
00054             }
00055         }
00056     }
00057     return objTarget;
00058 }

```

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

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

8.2 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 playerTransform)
Loads the players positon, roatation, and scale if it has previously been saved
- static void [SerializeInventory](#) ([Inventory.Inventory](#) inventory, string inventoryName)
Serializes a given [Inventory](#)
- static void [DeSerializeInventory](#) ([Inventory.Inventory](#) inventory, string inventoryName)
Deserializesd an [Inventory](#) from its name into a given inventory
- static void [SaveChunk](#) ([Chunk](#) chunk)

Saves a given Chunk if a block in it has been changed

- static bool [LoadChunk](#) ([Chunk](#) chunk)

Load a Chunk

- static string [FileName](#) ([ChunkWorldPos](#) pos)

Sets the file name of the Chunk

Static 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

8.2.1 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](#).

8.2.2 Member Function Documentation

8.2.2.1 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 = { "/", "\\ " };
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         }

```

8.2.2.2 DeSerializeInventory()

```

static void BeeGame.Serialization.Serialization.DeSerializeInventory (
    Inventory.Inventory inventory,
    string inventoryName ) [static]

```

Deserializes 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 = $"{savePath}/Inventorys/{inventoryName}.dat";
00136
00137             /* checks that the file exists
00138             if (!File.Exists(inventorySavePath))
00139                 return;
00140
00141             inventory.SetAllItems((ItemsInInventory)LoadFile($"{inventorySavePath}"
));
00142         }

```

8.2.2.3 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 195 of file [Serialization.cs](#).

```
00196         {
00197             return $"{pos.x}, {pos.y}, {pos.z}";
00198         }
```

8.2.2.4 LoadChunk()

```
static bool BeeGame.Serialization.Serialization.LoadChunk (
    Chunk chunk ) [static]
```

Load a Chunk

Parameters

<i>chunk</i>	
--------------	--

Returns

Definition at line 170 of file [Serialization.cs](#).

```
00171         {
00172             /** gets the save file
00173             string saveFile = $"{savePath}/{FileName(chunk.chunkWorldPos)}.dat";
00174
00175             /** if the file does not exist return false
00176             if (!File.Exists(saveFile))
00177                 return false;
00178
00179             /** set all of the changed blocks in the chunk
00180             SaveChunk save = (SaveChunk)LoadFile(saveFile);
00181
00182             foreach (var block in save.blocks)
00183             {
00184                 chunk.blocks[block.Key.x, block.Key.y, block.Key.z] = block.Value;
00185             }
00186
00187             return true;
00188         }
```

8.2.2.5 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 232 of file [Serialization.cs](#).

```

00233     {
00234         BinaryFormatter bf = new BinaryFormatter();
00235         FileStream fs = new FileStream(file, FileMode.Open);
00236
00237         try
00238         {
00239             return bf.Deserialize(fs);
00240         }
00241         catch (SerializationException e)
00242         {
00243             Debug.Log($"Deserialization Exception {e}");
00244             throw new SerializationException();
00245         }
00246         finally
00247         {
00248             fs.Close();
00249         }
00250     }

```

8.2.2.6 LoadPlayerPosition()

```

static void BeeGame.Serialization.Serialization.LoadPlayerPosition (
    Transform playerTransform ) [static]

```

Loads the players positon, roatation, and scale if it has previously been saved

Parameters

<i>playerTransform</i>	Transform to apply the data to
------------------------	--------------------------------

Definition at line 92 of file [Serialization.cs](#).

```

00093     {
00094         string playerPosSavePath = $"{savePath}/player.dat";
00095
00096         if (!File.Exists(playerPosSavePath))
00097             return;
00098
00099         THVector3[] pos = (THVector3[])LoadFile(playerPosSavePath);
00100
00101         playerTransform.position = pos[0];
00102         playerTransform.rotation = (Quaternion)pos[1];
00103         playerTransform.localScale = pos[2];
00104     }

```

8.2.2.7 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     }
```

8.2.2.8 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 150 of file [Serialization.cs](#).

```
00151     {
00152         /* saves the blocks
00153         SaveChunk save = new SaveChunk(chunk.blocks);
00154
00155         /* if no block was changed return early
00156         if (save.blocks.Count == 0)
00157             return;
00158
00159         /* otherwise save the file
00160         string saveFile = $"{savePath}/{FileName(chunk.chunkWorldPos)}.dat";
00161
00162         SaveFile(save, saveFile);
00163     }
```

8.2.2.9 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 207 of file [Serialization.cs](#).

```

00208     {
00209         BinaryFormatter bf = new BinaryFormatter();
00210         FileStream fs = new FileStream(file, FileMode.OpenOrCreate);
00211
00212         try
00213         {
00214             bf.Serialize(fs, obj);
00215         }
00216         catch (SerializationException e)
00217         {
00218             Debug.Log($"Serialization Exception: {e}");
00219             throw new SerializationException();
00220         }
00221         finally
00222         {
00223             fs.Close();
00224         }
00225     }

```

8.2.2.10 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     }

```

8.2.2.11 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         }
```

8.2.3 Member Data Documentation

8.2.3.1 saveFolderName

```
string BeeGame.Serialization.Serialization.saveFolderName = "Saves" [static]
```

Save folder

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

8.2.3.2 savePath

```
string BeeGame.Serialization.Serialization.savePath [static], [private]
```

Path to save things

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

8.2.3.3 worldName

```
string BeeGame.Serialization.Serialization.worldName = "World" [static]
```

Name if the world. If multiple world are ever added

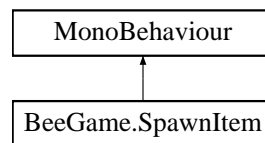
Definition at line 25 of file [Serialization.cs](#).

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

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

8.3 BeeGame.SpawnItem Class Reference

Inheritance diagram for BeeGame.SpawnItem:



Private Member Functions

- void [Start](#) ()
- void [OnDrawGizmos](#) ()

8.3.1 Detailed Description

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

8.3.2 Member Function Documentation

8.3.2.1 OnDrawGizmos()

```
void BeeGame.SpawnItem.OnDrawGizmos ( ) [private]
```

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

```
00024     {
00025         //Gizmos.color = Color.green;
00026         //Gizmos.DrawSphere(transform.position, 0.5f);
00027     }
```

8.3.2.2 Start()

```
void BeeGame.SpawnItem.Start ( ) [private]
```

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

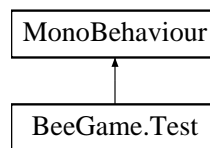
```
00015     {
00016         GameObject go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as
GameObject, transform.position, Quaternion.identity) as GameObject;
00017         go.GetComponent<ItemGameObject>().item = new HoneyComb(
HoneyCombType.ICEY);
00018
00019         go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
transform.position, Quaternion.identity) as GameObject;
00020         go.GetComponent<ItemGameObject>().item = new HoneyComb(
HoneyCombType.HONEY);
00021     }
```

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

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

8.4 BeeGame.Test Class Reference

Inheritance diagram for BeeGame.Test:



Private Member Functions

- void [Start](#) ()

8.4.1 Detailed Description

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

8.4.2 Member Function Documentation

8.4.2.1 Start()

```
void BeeGame.Test.Start ( ) [private]
```

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

```
00013     {  
00014         Instantiate(BeeGame.Core.PrefabDictionary.  
00015         GetPrefab("Selector"));  
00015     }
```

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

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

Part III

File Documentation

1 Items

1.1 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](#)

1.2 Item.cs

```

00001 using System;
00002 using UnityEngine;
00003 using BeeGame.Terrain.Chunks;
00004 using BeeGame.Core.Enums;
00005 using BeeGame.Core;
00006 using System.Runtime.Serialization.Formatters.Binary;
00007 using System.IO;
00008
00009 namespace BeeGame.Items
00010 {
00011     [Serializable]
00012     public class Item : ICloneable
00013     {
00014         #region Data
00015         internal string itemName = "Test Item";
00016         public bool placeable = false;
00017         public bool usesGameObject = false;
00018         private const float tileSize = 0.1f;
00019
00020         public int itemStackCount = 1;
00021         public int maxStackCount = 64;
00022         #endregion
00023
00024         #region Constructors
00025         public Item()
00026         {
00027             itemName = "TestItem";
00028         }
00029
00030         public Item(string name)
00031         {
00032             itemName = name;
00033         }
00034         #endregion
00035
00036         #region Item Stuff
00037         public virtual GameObject GetGameObject() { return null; }
00038
00039         public virtual string GetItemID()
00040         {
00041             return $"{GetHashCode()}";
00042         }
00043
00044         public virtual Sprite GetItemSprite()
00045         {
00046             return SpriteDictionary.GetSprite("TestSprite");
00047         }
00048
00049         public virtual string GetItemName()
00050         {
00051             return $"{itemName}";
00052         }
00053         #endregion
00054
00055         #region Item Mesh
00056         public virtual Tile TexturePosition(Direction direction)
00057         {
00058             return new Tile() { x = 1, y = 9 };
00059         }
00060
00061         public virtual MeshData ItemMesh(int x, int y, int z,
00062 MeshData meshData)
00063         {
00064             /* adds all faces of the item to the mesh as all faces could be seen at any time
00065             meshData = FaceDataUp(x, y, z, meshData, true, 0.25f);
00066             meshData = FaceDataDown(x, y, z, meshData, true, 0.25f);
00067             meshData = FaceDataNorth(x, y, z, meshData, true, 0.25f);
00068             meshData = FaceDataEast(x, y, z, meshData, true, 0.25f);
00069             meshData = FaceDataSouth(x, y, z, meshData, true, 0.25f);
00070             meshData = FaceDataWest(x, y, z, meshData, true, 0.25f);
00071
00072             return meshData;
00073         }
00074
00075         public virtual Vector2[] FaceUVs(Direction direction)
00076         {

```

```

00127         /** only 4 uvs per face
00128         Vector2[] UVs = new Vector2[4];
00129         Tile tilePos = TexturePosition(direction);
00130
00131         /** sets the UVs for each vertex
00132         UVs[0] = new THVector2(tileSize * tilePos.x + tileSize - 0.01f, tileSize * tilePos.
y + 0.01f);
00133         UVs[1] = new THVector2(tileSize * tilePos.x + tileSize - 0.01f, tileSize * tilePos.
y + tileSize - 0.01f);
00134         UVs[2] = new THVector2(tileSize * tilePos.x + 0.01f, tileSize * tilePos.
y + tileSize - 0.01f);
00135         UVs[3] = new THVector2(tileSize * tilePos.x + 0.01f, tileSize * tilePos.
y + 0.01f);
00136
00137         return UVs;
00138     }
00139
00150     protected virtual MeshData FaceDataUp(int x, int y, int z,
MeshData meshData, bool addToRenderMesh = true, float blockSize = 0.5f)
00151     {
00152         /** Adds vertices in a anti-clockwise order
00153         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z +
blockSize), addToRenderMesh, Direction.UP);
00154         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z +
blockSize), addToRenderMesh, Direction.UP);
00155         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z -
blockSize), addToRenderMesh, Direction.UP);
00156         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z -
blockSize), addToRenderMesh, Direction.UP);
00157
00158         /** adds teh tirs for the quad
00159         meshData.AddQuadTriangles(addToRenderMesh);
00160
00161         /** if the data should be added to the render mesh also add the uvs to the mesh
00162         if (addToRenderMesh)
00163             meshData.uv.AddRange(FaceUVs(Direction.UP));
00164
00165         return meshData;
00166     }
00167
00178     protected virtual MeshData FaceDataDown(int x, int y, int z,
MeshData meshData, bool addToRenderMesh = true, float blockSize = 0.5f)
00179     {
00180         /** Adds vertices in a anti-clockwise order
00181         meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z -
blockSize), addToRenderMesh);
00182         meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z -
blockSize), addToRenderMesh);
00183         meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z +
blockSize), addToRenderMesh);
00184         meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z +
blockSize), addToRenderMesh);
00185
00186         /** adds teh tirs for the quad
00187         meshData.AddQuadTriangles(addToRenderMesh);
00188
00189         /** if the data should be added to the render mesh also add the uvs to the mesh
00190         if (addToRenderMesh)
00191             meshData.uv.AddRange(FaceUVs(Direction.DOWN));
00192
00193         return meshData;
00194     }
00195
00206     protected virtual MeshData FaceDataNorth(int x, int y, int z,
MeshData meshData, bool addToRenderMesh = true, float blockSize = 0.5f)
00207     {
00208         /** Adds vertices in a anti-clockwise order
00209         meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z +
blockSize), addToRenderMesh);
00210         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z +
blockSize), addToRenderMesh);
00211         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z +
blockSize), addToRenderMesh);
00212         meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z +
blockSize), addToRenderMesh);
00213
00214         /** adds teh tirs for the quad
00215         meshData.AddQuadTriangles(addToRenderMesh);
00216
00217         /** if the data should be added to the render mesh also add the uvs to the mesh
00218         if (addToRenderMesh)
00219             meshData.uv.AddRange(FaceUVs(Direction.NORTH));
00220
00221         return meshData;
00222     }
00223
00234     protected virtual MeshData FaceDataEast(int x, int y, int z,

```

```

    MeshData meshData, bool addToRenderMesh = true, float blockSize = 0.5f)
00235     {
00236         /** Adds vertices in a anti-clockwise order
00237         meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z -
blockSize), addToRenderMesh);
00238         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z -
blockSize), addToRenderMesh);
00239         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z +
blockSize), addToRenderMesh);
00240         meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z +
blockSize), addToRenderMesh);
00241
00242         /** adds teh tirs for the quad
00243         meshData.AddQuadTriangles(addToRenderMesh);
00244
00245         /** if the data should be added to the render mesh also add the uvs to the mesh
00246         if (addToRenderMesh)
00247             meshData.uv.AddRange(FaceUVs(Direction.EAST));
00248
00249         return meshData;
00250     }
00251
00262     protected virtual MeshData FaceDataSouth(int x, int y, int z,
MeshData meshData, bool addToRenderMesh = true, float blockSize = 0.5f)
00263     {
00264         /** Adds vertices in a anti-clockwise order
00265         meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z -
blockSize), addToRenderMesh);
00266         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z -
blockSize), addToRenderMesh);
00267         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z -
blockSize), addToRenderMesh);
00268         meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z -
blockSize), addToRenderMesh);
00269
00270         /** adds teh tirs for the quad
00271         meshData.AddQuadTriangles(addToRenderMesh);
00272
00273         /** if the data should be added to the render mesh also add the uvs to the mesh
00274         if (addToRenderMesh)
00275             meshData.uv.AddRange(FaceUVs(Direction.SOUTH));
00276
00277         return meshData;
00278     }
00279
00290     protected virtual MeshData FaceDataWest(int x, int y, int z,
MeshData meshData, bool addToRenderMesh = true, float blockSize = 0.5f)
00291     {
00292         /** Adds vertices in a anti-clockwise order
00293         meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z +
blockSize), addToRenderMesh);
00294         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z +
blockSize), addToRenderMesh);
00295         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z -
blockSize), addToRenderMesh);
00296         meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z -
blockSize), addToRenderMesh);
00297
00298         /** adds teh tirs for the quad
00299         meshData.AddQuadTriangles(addToRenderMesh);
00300
00301         /** if the data should be added to the render mesh also add the uvs to the mesh
00302         if (addToRenderMesh)
00303             meshData.uv.AddRange(FaceUVs(Direction.WEST));
00304
00305         return meshData;
00306     }
00307     #endregion
00308
00309     #region Interfaces
00310     public object Clone()
00311     {
00312         /** Saves this to a file then reads it back so that a copy and not a reference is passed
00313         BinaryFormatter bf = new BinaryFormatter();
00314         MemoryStream ms = new MemoryStream();
00315
00316         bf.Serialize(ms, this);
00317         ms.Seek(0, SeekOrigin.Begin);
00318
00319         return bf.Deserialize(ms);
00320     }
00321     #endregion
00322
00323     #region Overrides
00324     public override string ToString()
00325     {
00326         return $"{itemName} \nID: {GetItemID()}";
00327     }
00328 }
00329
00330
00331
00332
00333
00334

```

```

00335     }
00336
00341     public override int GetHashCode()
00342     {
00343         return 1;
00344     }
00345
00351     public override bool Equals(object obj)
00352     {
00353         if (!(obj is Item))
00354             return false;
00355
00356         return this == (obj as Item);
00357     }
00358
00365     public static bool operator ==(Item a, Item b)
00366     {
00367         if (ReferenceEquals(a, null) && ReferenceEquals(b, null))
00368             return true;
00369         if (ReferenceEquals(a, null) || ReferenceEquals(b, null))
00370             return false;
00371
00372         if (a.GetItemID() == b.GetItemID())
00373             return true;
00374
00375         return false;
00376     }
00377
00384     public static bool operator !=(Item a, Item b)
00385     {
00386         return !(a == b);
00387     }
00388     #endregion
00389 }
00390
00394 [Serializable]
00395 public struct Tile
00396 {
00400     public int x;
00404     public int y;
00405 }
00406 }

```

1.3 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](#)

1.4 HoneyComb.cs

```

00001 using System;
00002 using System.Globalization;
00003 using BeeGame.Core;
00004 using BeeGame.Core.Enums;
00005 using UnityEngine;
00006
00007 namespace BeeGame.Items
00008 {
00012     [Serializable]
00013     public class HoneyComb : Item
00014     {
00015         #region Data
00016         public HoneyCombType type;
00020
00024         public Color CombColour

```

```

00025         {
00026             get
00027             {
00028                 return BeeDictionary.GetCombColour(type);
00029             }
00030         }
00031     #endregion
00032
00033     #region Constructors
00034     public HoneyComb() : base(new CultureInfo("en-US", false).TextInfo.ToTitleCase($"{
{HoneyCombType.HONEY} Comb".ToLower()))
00038     {
00039         usesGameObject = true;
00040         type = HoneyCombType.HONEY;
00041     }
00042
00043     public HoneyComb(HoneyCombType type) : base(new CultureInfo("en-US", false).
TextInfo.ToTitleCase($"{type.ToString()} Comb".ToLower()))
00048     {
00049         usesGameObject = true;
00050         this.type = type;
00051     }
00052     #endregion
00053
00054     #region Item Overrides
00055     public override Sprite GetItemSprite()
00060     {
00061         return SpriteDictionary.GetSprite("HoneyComb");
00062     }
00063
00064     public override GameObject GetGameObject()
00069     {
00070         GameObject obj = PrefabDictionary.GetPrefab("HoneyComb");
00071         /* cannot access the instance material from here have to do it on the object
00072         obj.GetComponent<ApplyColour>().colour = CombColour;
00073         return obj;
00074     }
00075
00076     public override string GetItemID()
00081     {
00082         return $"{GetHashCode()}\\{(int)type}";
00083     }
00084     #endregion
00085
00086     #region Overrides
00087     public override int GetHashCode()
00092     {
00093         return 8;
00094     }
00095     #endregion
00096 }
00097 }

```

1.5 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](#)

1.6 ItemGameObject.cs

```

00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using System.Text;
00005 using BeeGame.Terrain.Chunks;

```

```
00006 using BeeGame.Blocks;
00007 using UnityEngine;
00008
00009 namespace BeeGame.Items
00010 {
00014     [RequireComponent(typeof(Rigidbody))]
00015     [RequireComponent(typeof(MeshFilter))]
00016     [RequireComponent(typeof(MeshRenderer))]
00017     [RequireComponent(typeof(BoxCollider))]
00018     public class ItemGameObject : MonoBehaviour
00019     {
00023         public Item item;
00027         public GameObject go;
00028
00032         private void Start()
00033         {
00034             if (!item.usesGameObject)
00035                 MakeMesh();
00036
00037             if (item.usesGameObject)
00038             {
00039                 Instantiate(item.GetGameObject(), transform, false);
00040                 transform.localScale = new Vector3(0.5f, 0.5f, 0.5f);
00041             }
00042         }
00043
00047         private void Update()
00048         {
00049             if (transform.position.y < -100)
00050             {
00051                 Destroy(gameObject);
00052             }
00053         }
00054
00058         void MakeMesh()
00059         {
00060             MeshData meshData = new MeshData();
00061             if (item != null)
00062                 meshData = item.ItemMesh(0, 0, 0, meshData);
00063
00064             Mesh mesh = new Mesh()
00065             {
00066                 vertices = meshData.verts.ToArray(),
00067                 triangles = meshData.tris.ToArray(),
00068                 uv = meshData.uv.ToArray()
00069             };
00070
00071             mesh.RecalculateNormals();
00072
00073             GetComponent<MeshFilter>().mesh = mesh;
00074         }
00075     }
00076 }
```

1.7 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](#)

1.8 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 }

```

2 Blocks

2.1 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](#)

2.2 Block.cs

```

00001 using UnityEngine;
00002 using BeeGame.Terrain.Chunks;
00003 using BeeGame.Core.Enums;
00004 using BeeGame.Items;
00005 using BeeGame.Core;
00006
00007 namespace BeeGame.Blocks
00008 {
00012     [System.Serializable]
00013     public class Block : Item
00014     {
00015         #region Data
00016         public bool breakable = true;
00023         public bool changed = true;
00024         #endregion
00025
00026         #region Constructor
00027         public Block() : base()
00031         {
00032             itemName = "Stone";
00033             placeable = true;
00034         }
00035
00040         public Block(string name) : base(name)
00041         {
00042             placeable = true;
00043         }
00044         #endregion

```

```

00045
00046     #region Item Stuff
00047     public override Sprite GetItemSprite()
00048     {
00049         return SpriteDictionary.GetSprite("Stone");
00050     }
00051     #endregion
00052
00053     #region Update/Break Block
00054     public virtual void BreakBlock(THVector3 pos)
00055     {
00060         GameObject go = Object.Instantiate(UnityEngine.Resources.Load("
Prefabs/ItemGameObject") as GameObject, pos, Quaternion.identity) as GameObject;
00061         go.GetComponent<ItemGameObject>().item = this;
00062     }
00063
00071     public virtual void UpdateBlock(int x, int y, int z, Chunk chunk) { }
00072
00077     public virtual bool InteractWithBlock(BeeGame.
Inventory.Inventory inv)
00078     {
00079         return false;
00080     }
00081     #endregion
00082
00083     #region Mesh
00084     public virtual MeshData BlockData(Chunk chunk, int x, int y, int z,
MeshData meshData, bool addToRenderMesh = true)
00099     {
00100         /* Adds the Top face of the block
00101         if (!chunk.GetBlock(x, y + 1, z, false).IsSolid(Direction.DOWN))
00102         {
00103             meshData = FaceDataUp(x, y, z, meshData, addToRenderMesh);
00104         }
00105
00106         /* Adds the Bottom face of the block
00107         if (!chunk.GetBlock(x, y - 1, z, false).IsSolid(Direction.UP))
00108         {
00109             meshData = FaceDataDown(x, y, z, meshData, addToRenderMesh);
00110         }
00111
00112         /* Adds the North face of the block
00113         if (!chunk.GetBlock(x, y, z + 1, false).IsSolid(Direction.SOUTH))
00114         {
00115             meshData = FaceDataNorth(x, y, z, meshData, addToRenderMesh);
00116         }
00117
00118         /* Adds the South face of the block
00119         if (!chunk.GetBlock(x, y, z - 1, false).IsSolid(Direction.NORTH))
00120         {
00121             meshData = FaceDataSouth(x, y, z, meshData, addToRenderMesh);
00122         }
00123
00124         /* Adds the East face of the block
00125         if (!chunk.GetBlock(x + 1, y, z, false).IsSolid(Direction.WEST))
00126         {
00127             meshData = FaceDataEast(x, y, z, meshData, addToRenderMesh);
00128         }
00129
00130         /* Adds the West face of the block
00131         if (!chunk.GetBlock(x - 1, y, z, false).IsSolid(Direction.EAST))
00132         {
00133             meshData = FaceDataWest(x, y, z, meshData, addToRenderMesh);
00134         }
00135
00136         return meshData;
00137     }
00138
00144     public virtual bool IsSolid(Direction direction)
00145     {
00146         return true;
00147     }
00148     #endregion
00149
00150     #region Overrides
00151     public override int GetHashCode()
00152     {
00157         return 1;
00158     }
00159
00164     public override string ToString()
00165     {
00166         return $"{itemName} \nID: {GetHashCode()}";
00167     }
00168     #endregion
00169 }
00170 }

```

2.3 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](#)

2.4 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 Air() : base("Air")
00015         {
00016         }
00017
00022         public override void BreakBlock(THVector3 pos)
00023         {
00024             return;
00025         }
00026
00031         public override MeshData BlockData(CHUNK chunk, int x, int y, int z,
MeshData meshData, bool addRenderMesh = true)
00032         {
00033             return meshData;
00034         }
00035
00041         public override bool IsSolid(DIRECTION direction)
00042         {
00043             return false;
00044         }
00045
00050         public override int GetHashCode()
00051         {
00052             return 2;
00053         }
00054
00059         public override string ToString()
00060         {
00061             return $"{itemName} \nID: {GetItemID()}";
00062         }
00063     }
00064 }
```

2.5 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](#)

2.6 Dirt.cs

```

00001 using System;
00002 using BeeGame.Core.Enums;
00003 using BeeGame.Items;
00004 using BeeGame.Core;
00005 using UnityEngine;
00006
00007 namespace BeeGame.Blocks
00008 {
00012     [Serializable]
00013     public class Dirt : Block
00014     {
00015         #region Constructor
00016         public Dirt() : base("Dirt"){ }
00020         #endregion
00021
00022         #region Item Stuff
00023         public override Sprite GetItemSprite()
00024         {
00025             return SpriteDictionary.GetSprite("Dirt");
00026         }
00027         #endregion
00028
00029         #region Mesh
00030         public override Tile TexturePosition(Direction direction)
00036         {
00037             return new Tile { x = 2, y = 9 };
00038         }
00039         #endregion
00040
00041         #region Overrides
00042         public override int GetHashCode()
00047         {
00048             return 5;
00049         }
00050
00055         public override string ToString()
00056         {
00057             return $"{itemName} \nID: {GetItemID()}";
00058         }
00059         #endregion
00060     }
00061 }

```

2.7 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](#)

2.8 Grass.cs

```

00001 using System;
00002 using UnityEngine;
00003 using BeeGame.Core.Enums;
00004 using BeeGame.Terrain.Chunks;
00005 using BeeGame.Core;
00006 using BeeGame.Items;
00007
00008 namespace BeeGame.Blocks
00009 {
00010     [Serializable]
00011     public class Grass : Block
00012     {
00013         #region Constructor
00014         public Grass() : base("Grass") {}
00015         #endregion
00016
00017         #region Item Stuff
00018         public override Sprite GetItemSprite()
00019         {
00020             return SpriteDictionary.GetSprite("Grass");
00021         }
00022         #endregion
00023
00024         #region Mesh
00025         public override void UpdateBlock(int x, int y, int z, Chunk chunk)
00026         {
00027             if (chunk.GetBlock(x, y + 1, z, false).IsSolid(Direction.DOWN))
00028                 chunk.blocks[x, y, z] = new Dirt() { changed = changed };
00029         }
00030
00031         public override Tile TexturePosition(Direction direction)
00032         {
00033             //All textures are on the same Y value for the texture atlas so Y can be set
00034             Tile tile = new Tile()
00035             {
00036                 y = 9
00037             };
00038
00039             switch (direction)
00040             {
00041                 //if we want the top face return the full grass texture
00042                 case Direction.UP:
00043                     tile.x = 3;
00044                     return tile;
00045                 //if we want the bottom face return the dirt texture
00046                 case Direction.DOWN:
00047                     tile.x = 2;
00048                     return tile;
00049                 //return the 1/2 grass texture if a side face is wanted
00050                 default:
00051                     tile.x = 4;
00052                     return tile;
00053             }
00054         }
00055         #endregion
00056
00057         #region Overrides
00058         public override string GetItemName()
00059         {
00060             return "Grass";
00061         }
00062
00063         public override int GetHashCode()
00064         {
00065             return 4;
00066         }
00067
00068         public override string ToString()
00069         {
00070             return $"{itemName} \nID: {GetItemID()}";
00071         }
00072         #endregion
00073     }
00074 }
00075
00076
00077
00078
00079
00080
00081
00082
00083
00084
00085
00086
00087
00088
00089
00090
00091
00092
00093
00094
00095
00096
00097
00098
00099
00100

```

2.9 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Chest.cs File Reference

Classes

- class [BeeGame.Blocks.Chest](#)

Namespaces

- namespace [BeeGame.Blocks](#)

2.10 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
00009 namespace BeeGame.Blocks
00010 {
00011     [Serializable]
00012     public class Chest : Block
00013     {
00014         [NonSerialized]
00015         private GameObject myGameObject;
00016
00017         public Chest() : base("Chest")
00018         {
00019             usesGameObject = true;
00020         }
00021
00022         public override GameObject GetGameObject()
00023         {
00024
00025             return PrefabDictionary.GetPrefab("Chest");
00026         }
00027
00028         public override Tile TexturePosition(Direction direction)
00029         {
00030             return new Tile() { x = 0, y = 9 };
00031         }
00032
00033         public override MeshData BlockData(Chunk chunk, int x, int y, int z,
MeshData meshData, bool addToRenderMesh = true)
00034         {
00035             if (myGameObject == null)
00036             {
00037                 myGameObject = UnityEngine.Object.Instantiate(
PrefabDictionary.GetPrefab("Chest"), new THVector3(x, y, z) + chunk.
chunkWorldPos, Quaternion.identity, chunk.transform);
00038                 myGameObject.GetComponent<ChestInventory>().inventoryPosition = new
THVector3(x, y, z) + chunk.chunkWorldPos;
00039                 myGameObject.GetComponent<ChestInventory>().SetChestInventory();
00040             }
00041             return base.BlockData(chunk, x, y, z, meshData, true);
00042         }
00043
00044         public override bool InteractWithBlock(BeeGame.
Inventory.Inventory inv)
00045         {
00046             myGameObject.GetComponent<ChestInventory>().ToggleInventory(inv);
00047             return true;
00048         }
00049
00050         public override void BreakBlock(THVector3 pos)
00051         {
00052             Serialization.Serialization.DeleteFile(myGameObject.GetComponent<
ChestInventory>().inventoryName);
00053             UnityEngine.Object.Destroy(myGameObject);
00054             base.BreakBlock(pos);
00055         }
00056
00057         public override int GetHashCode()
00058         {
00059             return 8;
00060         }
00061
00062         public override string ToString()
00063         {
00064             return $"{itemName}\nID{GetItemID()}";
00065         }
00066     }
00067 }

```

2.11 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](#)

2.12 Apiary.cs

```
00001 using System.Runtime.Serialization;
00002
00003 namespace BeeGame.Blocks
00004 {
00008     public class Apiary : Block
00009     {
00010         #region Constructor
00011         public Apiary() : base("Apiary")
00015         {
00016         }
00017         #endregion
00018
00019         #region Overrides
00020         public override int GetHashCode()
00025         {
00026             return 3;
00027         }
00028
00033         public override string ToString()
00034         {
00035             return $"{itemName} \nID: {GetItemID()}";
00036         }
00037         #endregion
00038     }
00039 }
```

2.13 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Wood.cs File Reference

Classes

- class [BeeGame.Blocks.Wood](#)

Namespaces

- namespace [BeeGame.Blocks](#)

2.14 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;
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 Wood() : base("Wood")
00016         {
00017         }
00018     }
00019
00020     #region Item Stuff
00021     public override Sprite GetItemSprite()
00022     {
00023         return SpriteDictionary.GetSprite("Wood");
00024     }
00025     #endregion
00026
00027     public override Tile TexturePosition(Direction direction)
00028     {
00029         return new Tile() { x = 7, y = 9 };
00030     }
00031
00032     #region Overrides
00033     public override int GetHashCode()
00034     {
00035         return 6;
00036     }
00037
00038     public override string ToString()
00039     {
00040         return $"{itemName} \nID: {GetItemID()}";
00041     }
00042     #endregion
00043 }
00044
00045 }
00046
00047
00048
00049
00050
00051
00052

```

2.15 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Leaves.cs File Reference

Classes

- class [BeeGame.Blocks.Leaves](#)

Namespaces

- namespace [BeeGame.Blocks](#)

2.16 Leaves.cs

```

00001 using System;
00002 using UnityEngine;
00003 using BeeGame.Core;
00004 using BeeGame.Core.Enums;
00005 using BeeGame.Items;
00006
00007 namespace BeeGame.Blocks
00008 {
00009     [Serializable]
00010     public class Leaves : Block
00011     {

```



```

00012
00013     public Leaves() : base("Leaves")
00014     {
00015     }
00016 }
00017
00018 #region Item Stuff
00019 public override Sprite GetItemSprite()
00020 {
00021     return SpriteDictionary.GetSprite("Leaves");
00022 }
00023 #endregion
00024
00025 public override Tile TexturePosition(Direction direction)
00026 {
00027     return new Tile() { x = 5, y = 9 };
00028 }
00029
00030 public override bool IsSolid(Direction direction)
00031 {
00032     return false;
00033 }
00034
00035 #region Overrides
00036 public override int GetHashCode()
00041 {
00042     return 7;
00043 }
00044
00049 public override string ToString()
00050 {
00051     return $"{itemName} \nID: {GetItemID()}";
00052 }
00053 #endregion
00054 }
00055 }

```

2.17 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](#)

2.18 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 Constructor
00015         public Bedrock() : base("Bedrock")
00019         {
00020             breakable = false;
00021         }
00022         #endregion
00023
00024         #region Break Block
00025         public override void BreakBlock(THVector3 pos)
00030         {
00031             return;

```

```

00032     }
00033     #endregion
00034
00035     #region Mesh
00036     public override Tile TexturePosition(Direction direction)
00042     {
00043         return new Tile() { x = 0, y = 0 };
00044     }
00045     #endregion
00046
00047     #region Overrides
00048     public override int GetHashCode()
00053     {
00054         return -1;
00055     }
00056
00061     public override string ToString()
00062     {
00063         return $"{itemName} \nID: {GetItemID()}";
00064     }
00065     #endregion
00066 }
00067 }

```

3 Inventorys

3.1 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](#)

3.2 Inventory.cs

```

00001 using UnityEngine;
00002 using BeeGame.Items;
00003
00004 namespace BeeGame.Inventory
00005 {
00009     public class Inventory : MonoBehaviour
00010     {
00011         #region Data
00012         public ItemsInInventory items;
00019         public InventorySlot[] slots;
00023         internal Item floatingItem;
00027         public string inventoryName = "";
00031         protected bool thisInventoryOpen = false;
00032         #endregion
00033
00034         #region Init
00035         public bool InventorySet()
00040         {
00041             if (items == null)
00042                 return true;
00043
00044             return false;
00045         }
00046
00051         public void SetInventorySize(int inventorySize)
00052         {
00053             items = new ItemsInInventory(slots.Length);
00054         }

```

3.3

```
00055
00063     public void SetAllItems(ItemsInInventory items)
00064     {
00065         this.items = items;
00066     }
00067     #endregion
00068
00069     #region Update
00070     public void UpdateBase()
00074     {
00075         PutItemsInSlots();
00076     }
00077     #endregion
00078
00079     #region Edit Inventory
00080     public void SaveInv()
00087     {
00088         Serialization.Serialization.SerializeInventory(this, inventoryName);
00089     }
00090
00094     void PutItemsInSlots()
00095     {
00096         /* goes through all of the items in the array setting them all to a slot
00097         for (int i = 0; i < slots.Length; i++)
00098         {
00099             slots[i].slotIndex = i;
00100             slots[i].myInventory = this;
00101             slots[i].item = items.itemsInInventory[i];
00102         }
00103     }
00104
00109     public ItemsInInventory GetAllItems()
00110     {
00111         return items;
00112     }
00113
00119     public void AddItemToSlots(int slotIndex, Item item)
00120     {
00121         items.AddItem(slotIndex, item);
00122         /* saves the inventory changes
00123         Serialization.Serialization.SerializeInventory(this, inventoryName);
00124     }
00125
00131     public bool AddItemToInventory(Item item)
00132     {
00133         return items.AddItem(item);
00134     }
00135     #endregion
00136 }
00137 }
```

3.3 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/InventorySlot.cs File Reference

Classes

- class [BeeGame.Inventory.InventorySlot](#)

Namespaces

- namespace [BeeGame.Inventory](#)

3.4 InventorySlot.cs

```
00001 using UnityEngine;
00002 using UnityEngine.UI;
00003 using UnityEngine.EventSystems;
00004 using BeeGame.Items;
00005 using BeeGame.Core;
00006
00007 namespace BeeGame.Inventory
00008 {
```

```

00009     public class InventorySlot : MonoBehaviour, IPointerClickHandler, IPointerEnterHandler,
IPointerExitHandler
00010     {
00011         #region Data
00012         internal int slotIndex;
00019         public Item item;
00023         public Inventory myInventory;
00027         public GameObject itemText;
00031         public bool selectedSlot = false;
00032     #endregion
00033
00037     private void Update()
00038     {
00039         UpdateIcon();
00040     }
00041
00045     void UpdateIcon()
00046     {
00047         if(item == null)
00048         {
00049             GetComponent<Image>().sprite = null;
00050         }
00051         else
00052         {
00053             GetComponent<Image>().sprite = item.GetItemSprite();
00054         }
00055
00056         /* if the slot is selected in the hotbar give the player some indication by colouring it grey
00057         if (selectedSlot)
00058         {
00059             GetComponent<Image>().color = Color.gray;
00060         }
00061         else
00062         {
00063             GetComponent<Image>().color = Color.white;
00064         }
00065
00066         /* sets the colour of the slot to the correct colour for the item
00067         /* make this easier then colouring many of the same sprite different colours
00068         if(item != null)
00069         {
00070             switch (item)
00071             {
00072                 case HoneyComb c:
00073                     GetComponent<Image>().color = c.CombColour;
00074                     break;
00075             }
00076         }
00077     }
00078
00079     #region Interact With Slot
00080     public void OnPointerClick(PointerEventData eventData)
00081     {
00082         if (myInventory.floatingItem != null)
00083         {
00084             /* Left click moves whole stacks if items
00085             if (eventData.button == PointerEventData.InputButton.Left)
00086             {
00087                 /* If the item in the slot is empty put the floating item into it then clear it
00088                 if (item == null)
00089                 {
00090                     item = myInventory.floatingItem;
00091                     myInventory.floatingItem = null;
00092                     myInventory.AddItemToSlots(slotIndex, item);
00093                     return;
00094                 }
00095                 /* if the items are the same
00096                 if(myInventory.floatingItem == item)
00097                 {
00098                     /* if the item in the inventoys stack count + the floating items stack count is
00099                     less than the max stack count
00100                     if (myInventory.floatingItem.itemStackCount + item.
00101                     itemStackCount <= item.maxStackCount)
00102                     {
00103                         AddToSlot(myInventory.floatingItem.
00104                         itemStackCount);
00105                     }
00106                     return;
00107                 }
00108                 /* if the item stack added is larger than the max count add as many as you can and
00109                 move on
00110                 else
00111                 {
00112                     AddToSlot(item.maxStackCount - item.
00113                     itemStackCount);
00114                     return;
00115                 }
00116             }
00117         }

```

```

00118             /** If the items were not == swap them
00119             else
00120             {
00121                 SwapItems();
00122                 return;
00123             }
00124         }
00125         else if (eventData.button == PointerEventData.InputButton.Right)
00126         {
00127             /** if the item in slot is null add 1 from the floating item to it
00128             if (item == null)
00129             {
00130                 AddToSlot(1);
00131                 return;
00132             }
00133             /** if the items are the same add 1 from the floating item to this item
00134             else if (item == myInventory.floatingItem)
00135             {
00136                 AddToSlot(1);
00137                 return;
00138             }
00139         }
00140     }
00141     /** if the floating item is null
00142     else
00143     {
00144         /** add 1/2 of the stack into the floating item if right click was pressed
00145         if (eventData.button == PointerEventData.InputButton.Right)
00146         {
00147             SplitStack();
00148             return;
00149         }
00150
00151         /** otherwie add the items into the floating item slot
00152         SwapItems();
00153         return;
00154     }
00155 }
00156
00157
00162 void AddToSlot(int numerToAdd)
00163 {
00164     /** if the item in the slot is null create it
00165     if (item == null)
00166     {
00167         item = myInventory.floatingItem.CloneObject();
00168         item.itemStackCount = 0;
00169     }
00170
00171     /** add to number to add to the stack count
00172     item.itemStackCount += numerToAdd;
00173
00174     /** if the stack count is now larger than it should be dont let it be
00175     if (item.itemStackCount > item.maxStackCount)
00176     {
00177         item.itemStackCount = item.maxStackCount;
00178     }
00179
00180     /** remove the numebr if items form the floating item then check the floating item is not null
00181     myInventory.floatingItem.itemStackCount -= numerToAdd;
00182     CheckFloatingItem();
00183     /** save the inventory changes
00184     myInventory.AddItemToSlots(slotIndex, item);
00185 }
00186
00193 void SplitStack()
00194 {
00195     myInventory.floatingItem = item.CloneObject();
00196     int give = (item.itemStackCount + 1) / 2;
00197     myInventory.floatingItem.itemStackCount = give;
00198     item.itemStackCount -= give;
00199
00200     if (item.itemStackCount <= 0)
00201     {
00202         item = null;
00203     }
00204     myInventory.AddItemToSlots(slotIndex, item);
00205     Destroy(itemText);
00206 }
00210 void SwapItems()
00211 {
00212     /** temp copy of the item
00213     Item temp = myInventory.floatingItem;
00214     /** sets the floating item
00215     myInventory.floatingItem = item;
00216     /** sets the item that was in the floating item to the item in the the slot
00217     item = temp;

```

```

00218         /** Saves the changes to the inventory
00219         myInventory.AddItemToSlots(slotIndex, item);
00220         /** destroys the text as it is not needed anymore
00221         Destroy(itemText);
00222     }
00223
00227     void CheckFloatingItem()
00228     {
00229         if(myInventory.floatingItem.itemStackCount <= 0)
00230         {
00231             myInventory.floatingItem = null;
00232         }
00233     }
00234     #endregion
00235
00236     #region Display Item On Hover
00237     public void OnPointerEnter(PointerEventData eventData)
00242     {
00243         /** if the item is null or the floating item has something in it dont display the item text as
00244         it is not necessary
00245         if (item != null && myInventory.floatingItem == null)
00246         {
00247             itemText = Instantiate(PrefabDictionary.
GetPrefab("ItemDetails"));
00248             /** sets the text to the correct position
00249             itemText.transform.GetChild(0).position = Input.mousePosition;
00250             /** puts the correct text in the box
00251             itemText.transform.GetChild(0).GetChild(0).GetComponent<Text>().text = $"
{item.GetItemName()}\nStack: {item.itemStackCount}";
00252         }
00253     }
00254
00258     public void OnPointerExit(PointerEventData eventData)
00259     {
00260         Destroy(itemText);
00261     }
00262
00266     void OnDisable()
00267     {
00268         Destroy(itemText);
00269     }
00270     #endregion
00271 }
00272 }

```

3.5 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](#)

3.6 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()
00021     {
00022         SetPlayerInventory();
00023         inventoryName = "PlayerInventory";
00024         Serialization.Serialization.DeSerializeInventory(this, inventoryName);
00025     }
00026
00027     void SetPlayerInventory()
00028     {
00029         if (!InventorySet())
00030             SetInventorySize(36);
00031     }
00032 #endregion
00033
00034     void Update()
00035     {
00036         UpdateBase();
00037
00038         /* checks if the inventory should be opened/closed
00039         if ((thisInventoryOpen || !playerInventory.activeInHierarchy) &&
00040             THInput.GetButtonDown("Player Inventory"))
00041         {
00042             if (THInput.blockInventoryJustClosed)
00043             {
00044                 THInput.blockInventoryJustClosed = false;
00045                 return;
00046             }
00047             else
00048             {
00049                 OpenPlayerInventory();
00050             }
00051         }
00052
00053         /* dont pickup items if the inventory is open
00054         if (THInput.isAnotherInventoryOpen)
00055             return;
00056
00057         /* checks if somethig should be picked up and put into the inventory
00058         RaycastHit[] hit = Physics.SphereCastAll(transform.position, 1f, transform.forward);
00059
00060         for (int i = hit.Length - 1; i >= 0; i--)
00061         {
00062             if (hit[i].collider.GetComponent<ItemGameObject>())
00063                 PickupItem(hit[i].collider.GetComponent<ItemGameObject>());
00064         }
00065     }
00066
00067 #region Hotbar
00068 public void SelectedSlot(int index)
00069 {
00070     for (int i = 0; i < slots.Length; i++)
00071     {
00072         slots[i].selectedSlot = false;
00073     }
00074     slots[index].selectedSlot = true;
00075 }
00076
00077 public bool GetItemFromHotBar(int slotIndex, out Item outItem)
00078 {
00079     /* get the item
00080     outItem = GetAllItems().itemsInInventory[slotIndex];
00081
00082     if (outItem == null)
00083         return false;
00084
00085     /* if the item is placeable and is not null remove 1 from the inventory as it is assumed it is
00086     about to be placed in the world
00087     if(outItem.placeable)
00088         RemoveItemFromInventory(slotIndex);
00089
00090     return outItem.placeable;
00091 }
00092 #endregion
00093
00094 #region Interact With Inventory
00095 void OpenPlayerInventory()
00096 {
00097     thisInventoryOpen = !thisInventoryOpen;
00098     playerInventory.SetActive(!playerInventory.activeInHierarchy);
00099     THInput.isAnotherInventoryOpen = !
00100     THInput.isAnotherInventoryOpen;
00101
00102     /* hides/shows the mouse depending on if te inventory is open or not
00103     if (playerInventory.activeInHierarchy)
00104     {

```

3.7

C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/ChestInventory.cs

File Reference

222

```
00126         Cursor.lockState = CursorLockMode.None;
00127         Cursor.visible = true;
00128     }
00129     else
00130     {
00131         Cursor.visible = false;
00132         Cursor.lockState = CursorLockMode.Locked;
00133     }
00134 }
00135
00136 public void RemoveItemFromInventory(int index)
00137 {
00138     /* if the item is already null nothign needs to be removed
00139     if (GetAllItems().itemsInInventory[index] != null)
00140     {
00141         /* remove 1 item and if that was the last in the stack remove the item from the inventory
00142         GetAllItems().itemsInInventory[index].itemStackCount -= 1;
00143
00144         if (GetAllItems().itemsInInventory[index].itemStackCount <= 0)
00145             GetAllItems().itemsInInventory[index] = null;
00146
00147         Serialization.Serialization.SerializeInventory(this, inventoryName);
00148     }
00149 }
00150
00151 void PickupItem(ItemGameObject item)
00152 {
00153     item.item.itemStackCount = 1;
00154
00155     /* if the item can be added to the inventory do that
00156     if (AddItemToInventory(item.item))
00157     {
00158         /* if the item was added destroy its gameobject and save the inventory
00159         Destroy(item.gameObject);
00160         Serialization.Serialization.SerializeInventory(this, inventoryName);
00161     }
00162 }
00163 #endregion
00164 }
00165 }
```

3.7 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/ChestInventory.cs

File Reference

Classes

- class [BeeGame.Inventory.ChestInventory](#)
Inventory for the chests

Namespaces

- namespace [BeeGame.Inventory](#)

3.8 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 {
00008     public class ChestInventory : Inventory
00009     {
00010         #region Data
00011         public THVector3 inventoryPosition;
00012         public Inventory playerInventory;
00013         public GameObject inventory;
00014
00015         public int inventorySize;
00016     }
00017 #endregion
00018 }
```



```

00033         #region Unity Methods
00034         void Update()
00038         {
00039             /** the chest should always have a player inventory when it does this but checks just in case
00040             if (playerinventory != null)
00041                 UpdateBase();
00042
00043             /** checks if the inventory should be closed
00044             if (GetButtonDown("Player Inventory") && thisInventoryOpen)
00045                 ToggleInventory(playerinventory);
00046         }
00047     #endregion
00048
00052     public void SetChestInventory()
00053     {
00054         SetInventorySize(inventorySize);
00055         /** sets the UI to not be seen as inventorys cannot start open
00056         inventory.SetActive(false);
00057
00058         /** sets the name and postion if this inventory used during serialization and deserialization
00059         inventoryName = $"Chest @ {(ChunkWorldPos)inventoryPosition}";
00060
00061         /** loads the inventory if it had had items put in it last time it existed
00062         Serialization.Serialization.DeSerializeInventory(this, inventoryName);
00063     }
00064
00065     #region Player Inventory
00066     void SetPlayerItems()
00070     {
00071         for (int i = 0; i < playerinventory.items.itemsInInventory.Length; i++)
00072         {
00073             items.itemsInInventory[i + (inventorySize - 36)] = playerinventory.
00074             items.itemsInInventory[i];
00075         }
00076
00080     void ApplyPlayerItems()
00081     {
00082         for (int i = 0; i < playerinventory.items.itemsInInventory.Length; i++)
00083         {
00084             playerinventory.items.itemsInInventory[i] = items.itemsInInventory[i +
(inventorySize - 36)];
00085         }
00086
00087         playerinventory.SaveInv();
00088     }
00089     #endregion
00090
00095     public void ToggleInventory(Inventory inv)
00096     {
00097         /** sets the player inventory
00098         playerinventory = inv;
00099
00100         thisInventoryOpen = !thisInventoryOpen;
00101
00102         isAnotherInventoryOpen = thisInventoryOpen;
00103
00104         inventory.SetActive(!inventory.activeInHierarchy);
00105
00106         if (inventory.activeInHierarchy)
00107         {
00108             /** stops the player invnetory from being opened immidiatly after this is closed
00109             blockInventoryJustClosed = true;
00110             SetPlayerItems();
00111             /** hides and locks the cursor
00112             Cursor.lockState = CursorLockMode.None;
00113             Cursor.visible = true;
00114         }
00115         else
00116         {
00117             /** puts the items into the chest
00118             /** shows and unlocks the cursor
00119             ApplyPlayerItems();
00120             Cursor.lockState = CursorLockMode.Locked;
00121             Cursor.visible = false;
00122         }
00123     }
00124 }
00125 }

```

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](#)

3.10 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 }

```

4 Chunks

4.1 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](#)

4.2 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()
00078         {
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)
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     }
00135
00143     public void SetBlock(int x, int y, int z, Block block, bool checkNeighbouringChunks =
true)
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 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 alot 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

```

4.3

C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/MeshData.cs

File Reference

227

```
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
00244         mesh otherwise make a collision mesh
00245         if (this.mesh.shareMeshes)
00246         {
00247             world.chunkHasMadeCollisionMesh = true;
00248             applyCollisionMesh = false;
00249             meshCollider.sharedMesh = filter.mesh;
00250             return;
00251         }
00252         world.chunkHasMadeCollisionMesh = true;
00253         /* Applying the mesh takes the longest but nothing can be done with the mesh class in a
00254         secondary thread...thanks Unity
00255
00256         /* makes a new mesh setting the name for convenience
00257         Mesh mesh = new Mesh()
00258         {
00259             name = "Collider Mesh",
00260             vertices = this.mesh.colVerts.ToArray(),
00261             triangles = this.mesh.colTris.ToArray()
00262         };
00263
00264         /* recalcs the normals and applies the mesh
00265         mesh.RecalculateNormals();
00266
00267         meshCollider.sharedMesh = mesh;
00268
00269         applyCollisionMesh = false;
00270     }
00271     #endregion
00272 }
```

4.3 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/MeshData.cs File Reference

Classes

- class [BeeGame.Terrain.Chunks.MeshData](#)
The data for a [Chunks](#)'s Mesh

Namespaces

- namespace [BeeGame.Terrain.Chunks](#)

4.4 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 positon 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 }

```

4.5 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](#)

4.6 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.
00096         z);
00097     }
00098 }

```

4.7 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](#)

4.8 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
ChunkWorldPos[] { new ChunkWorldPos( 0, 0, 0), new
ChunkWorldPos(-1, 0, 0), new ChunkWorldPos( 0, 0, -1), new
ChunkWorldPos( 0, 0, 1), new ChunkWorldPos( 1, 0, 0),
00028         new ChunkWorldPos(-1, 0, -1), new
ChunkWorldPos(-1, 0, 1), new ChunkWorldPos( 1, 0, -1), new
ChunkWorldPos( 1, 0, 1), new ChunkWorldPos(-2, 0, 0),
00029         new ChunkWorldPos( 0, 0, -2), new
ChunkWorldPos( 0, 0, 2), new ChunkWorldPos( 2, 0, 0), new
ChunkWorldPos(-2, 0, -1), new ChunkWorldPos(-2, 0, 1),
00030         new ChunkWorldPos(-1, 0, -2), new
ChunkWorldPos(-1, 0, 2), new ChunkWorldPos( 1, 0, -2), new
ChunkWorldPos( 1, 0, 2), new ChunkWorldPos( 2, 0, -1),
00031         new ChunkWorldPos( 2, 0, 1), new
ChunkWorldPos(-2, 0, -2), new ChunkWorldPos(-2, 0, 2), new
ChunkWorldPos( 2, 0, -2), new ChunkWorldPos( 2, 0, 2),
00032         new ChunkWorldPos(-3, 0, 0), new
ChunkWorldPos( 0, 0, -3), new ChunkWorldPos( 0, 0, 3), new
ChunkWorldPos( 3, 0, 0), new ChunkWorldPos(-3, 0, -1),
00033         new ChunkWorldPos(-3, 0, 1), new
ChunkWorldPos(-1, 0, -3), new ChunkWorldPos(-1, 0, 3), new
ChunkWorldPos( 1, 0, -3), new ChunkWorldPos( 1, 0, 3),
00034         new ChunkWorldPos( 3, 0, -1), new
ChunkWorldPos( 3, 0, 1), new ChunkWorldPos(-3, 0, -2), new
ChunkWorldPos(-3, 0, 2), new ChunkWorldPos(-2, 0, -3),
00035         new ChunkWorldPos(-2, 0, 3), new
ChunkWorldPos( 2, 0, -3), new ChunkWorldPos( 2, 0, 3), new
ChunkWorldPos( 3, 0, -2), new ChunkWorldPos( 3, 0, 2),
00036         new ChunkWorldPos(-4, 0, 0), new
ChunkWorldPos( 0, 0, -4), new ChunkWorldPos( 0, 0, 4), new
ChunkWorldPos( 4, 0, 0), new ChunkWorldPos(-4, 0, -1),
00037         new ChunkWorldPos(-4, 0, 1), new
ChunkWorldPos(-1, 0, -4), new ChunkWorldPos(-1, 0, 4), new
ChunkWorldPos( 1, 0, -4), new ChunkWorldPos( 1, 0, 4),
00038         new ChunkWorldPos( 4, 0, -1), new
ChunkWorldPos( 4, 0, 1), new ChunkWorldPos(-3, 0, -3), new
ChunkWorldPos(-3, 0, 3), new ChunkWorldPos( 3, 0, -3),
00039         new ChunkWorldPos( 3, 0, 3), new
ChunkWorldPos(-4, 0, -2), new ChunkWorldPos(-4, 0, 2), new
ChunkWorldPos(-2, 0, -4), new ChunkWorldPos(-2, 0, 4),
00040         new ChunkWorldPos( 2, 0, -4), new
ChunkWorldPos( 2, 0, 4), new ChunkWorldPos( 4, 0, -2), new
ChunkWorldPos( 4, 0, 2), new ChunkWorldPos(-5, 0, 0),
00041         new ChunkWorldPos(-4, 0, -3), new
ChunkWorldPos(-4, 0, 3), new ChunkWorldPos(-3, 0, -4), new
ChunkWorldPos(-3, 0, 4), new ChunkWorldPos( 0, 0, -5),
00042         new ChunkWorldPos( 0, 0, 5), new
ChunkWorldPos( 3, 0, -4), new ChunkWorldPos( 3, 0, 4), new
ChunkWorldPos( 4, 0, -3), new ChunkWorldPos( 4, 0, 3),
00043         new ChunkWorldPos( 5, 0, 0), new
ChunkWorldPos(-5, 0, -1), new ChunkWorldPos(-5, 0, 1), new
ChunkWorldPos(-1, 0, -5), new ChunkWorldPos(-1, 0, 5),
00044         new ChunkWorldPos( 1, 0, -5), new
ChunkWorldPos( 1, 0, 5), new ChunkWorldPos( 5, 0, -1), new
ChunkWorldPos( 5, 0, 1), new ChunkWorldPos(-5, 0, -2),
00045         new ChunkWorldPos(-5, 0, 2), new
ChunkWorldPos(-2, 0, -5), new ChunkWorldPos(-2, 0, 5), 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 calls 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 }

```

4.9 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/SaveChunk.cs File Reference

Classes

- class [BeeGame.Terrain.Chunks.SaveChunk](#)
Saves a [Chunk](#)s modified Blocks for save optimisation

Namespaces

- namespace [BeeGame.Terrain.Chunks](#)

4.10 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>();
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 }

```

4.11 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](#)

4.12 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 {
00011     public class World : MonoBehaviour
00012     {
00013         #region Data
00014         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 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         }
00090
00097         public void DestroyChunk(int x, int y, int z)
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    }
00108    #endregion
00109
00110    #region Block
00111    public void SetBlock(int x, int y, int z, Block block, bool saveChunk = false)
00112    {
00113        /**gets the chunk for the block to be placed in
00114        Chunk chunk = GetChunk(x, y, z);
00115
00116        /**if the chunk is not null and the block trying to be replaced is replaceable, replace it
00117        if(chunk != null && chunk.blocks[x - chunk.chunkWorldPos.
00118        x, y - chunk.chunkWorldPos.y, z - chunk.chunkWorldPos.
00119        z].breakable)
00120        {
00121            chunk.SetBlock(x - chunk.chunkWorldPos.x, y - chunk.
00122            chunkWorldPos.y, z - chunk.chunkWorldPos.z, block);
00123            chunk.update = true;
00124
00125            /**updates the nebouring chunks as when a block is broken it may be in the edge of the
00126            chunk so their meshes also need to be updated
00127            /**only updates chunks that need to be updated as not every chunk will need to be and
00128            sometimes none of them will need to be
00129
00130            /**checks if the block chaged is in the edge if the x value for the chunk
00131            UpdateIfEqual(x - chunk.chunkWorldPos.x, 0, new
00132            ChunkWorldPos(x - 1, y, z));
00133            UpdateIfEqual(x - chunk.chunkWorldPos.x, Chunk.
00134            chunkSize - 1, new ChunkWorldPos(x + 1, y, z));
00135
00136            /**checks if the block chaged is in the edge if the y value for the chunk
00137            UpdateIfEqual(y - chunk.chunkWorldPos.y, 0, new
00138            ChunkWorldPos(x, y - 1, z));
00139            UpdateIfEqual(y - chunk.chunkWorldPos.y, Chunk.
00140            chunkSize - 1, new ChunkWorldPos(x, y + 1, z));
00141
00142            /**checks if the block chaged is in the edge if the z value for the chunk
00143            UpdateIfEqual(z - chunk.chunkWorldPos.z, 0, new
00144            ChunkWorldPos(x, y, z - 1));
00145            UpdateIfEqual(z - chunk.chunkWorldPos.z, Chunk.
00146            chunkSize - 1, new ChunkWorldPos(x, y, z + 1));
00147
00148            if (saveChunk)
00149                Serialization.Serialization.SaveChunk(chunk);
00150        }
00151    }
00152    #endregion
00153
00154    #region Get Things
00155    public Chunk GetChunk(int x, int y, int z)
00156    {
00157        float multiple = Chunk.chunkSize;
00158        /** rounds the given x, y, z to a multiple of 16 as chunks are 16x16x16 in size
00159        ChunkWorldPos pos = new ChunkWorldPos()
00160        {
00161            x = Mathf.FloorToInt(x / multiple) * Chunk.chunkSize,
00162            y = Mathf.FloorToInt(y / multiple) * Chunk.chunkSize,
00163            z = Mathf.FloorToInt(z / multiple) * Chunk.chunkSize
00164        };
00165
00166        /** gets the chunk if it exists
00167        chunks.TryGetValue(pos, out Chunk chunk);
00168        /** if the chunk does not exist will return null
00169        return chunk;
00170    }
00171
00172    public Block GetBlock(int x, int y, int z)
00173    {
00174        /** gets the chunk that the block is in
00175        Chunk chunk = GetChunk(x, y, z);
00176
00177        if(chunk != null)
00178        {
00179            /** gets the block in the chunk
00180            return chunk.GetBlock(x - chunk.chunkWorldPos.
00181            x, y - chunk.chunkWorldPos.y, z - chunk.chunkWorldPos.
00182            z) ?? new Air();
00183        }
00184    }
00185
00186
00187
00188
00189
00190
00191
00192
00193
00194

```

```

00195         /** returns an empty block is the chunk was not found
00196         return new Air();
00197     }
00198     #endregion
00199
00206     void UpdateIfEqual(int value1, int value2, ChunkWorldPos pos)
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     }
00216 }
00217 }

```

4.13 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](#)

4.14 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
00055 hit)
00056         {
00057             return new ChunkWorldPos((int)RoundXZ(hit.point.x, hit.normal.x), (int)RoundY(hit.
00058 point.y, hit.normal.y), (int)RoundXZ(hit.point.z, hit.normal.z));
00059         }
00060     }
00061 }

```

```

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
00154                 sometimes 0.5 would round to 0 not 1
00155                 return Mathf.RoundToInt((float)Math.Round(pos, 1)) - normal;
00156
00157             return Mathf.RoundToInt((float)Math.Round(pos, 1)); // - normal;
00158         }
00159
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     return Mathf.RoundToInt((float)Math.Round(pos, 1));
00166 }
00167
00168 public static float Round(float pos, float norm, bool adjacent = false)
00169 {
00170     if(pos - (int)pos == 0.5f || pos - (int)pos == -0.5f)
00171     {
00172         if(adjacent)
00173         {
00174             pos += (norm / 2);
00175         }
00176         else
00177         {
00178             pos -= (norm / 2);
00179         }
00180     }
00181
00182     return pos;
00183 }
00184 #endregion
00185
00186 #region Get Block
00187 public static ChunkWorldPos GetBlockPos(RaycastHit hit, bool adjacent = false)
00188 {
00189     return GetBlockPos(new THVector3()
00190     {
00191         /* rounds the hit to the correct position
00192         x = Round(hit.point.x, hit.normal.x, adjacent),
00193         y = Round(hit.point.y, hit.normal.y, adjacent),
00194         z = Round(hit.point.z, hit.normal.z, adjacent)
00195     });
00196 }
00197
00198 public static Block GetBlock(RaycastHit hit, bool adjacent = false)
00199 {
00200     /* checks that a chunk was hit and if it wasnt return early
00201     Chunk chunk = hit.collider.GetComponent<Chunk>();
00202
00203     if (chunk == null)
00204         return null;
00205
00206     /* allignes the hit to the block grid and returns the block
00207     ChunkWorldPos pos = GetBlockPos(hit, adjacent);
00208
00209     return chunk.world.GetBlock(pos.x, pos.y, pos.z);
00210 }
00211
00212 public static Block GetBlock(THVector3 pos)
00213 {
00214     Chunk chunk = GetChunk(pos);
00215
00216     if (chunk == null)
00217         return new Air();
00218
00219     chunk.world.GetBlock((int)pos.x, (int)pos.y, (int)pos.z);
00220
00221     return new Block();
00222 }
00223
00224 public static bool BlockInPosition(THVector3 pos,
00225 Chunk chunk)
00226 {
00227     if (chunk == null)
00228         return false;
00229
00230     if (chunk.GetBlock((int)pos.x, (int)pos.y, (int)pos.z) != new
00231 Air())
00232         return true;
00233
00234     return false;
00235 }
00236 #endregion
00237
00238 public static Chunk GetChunk(THVector3 vec3)
00239 {
00240     return world.GetChunk((int)vec3.x, (int)vec3.y, (int)vec3.
00241 z);
00242 }
00243
00244 #region Set Block
00245 public static bool SetBlock(RaycastHit hit, Block block, bool adjacent = false)
00246 {
00247     /* 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 }

```

4.15 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](#)

4.16 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 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     }
00199
00210     void CreateTree(int x, int y, int z, Chunk chunk)
00211     {
00212         /** makes the leaves of the 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 }
```

4.17 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](#)

4.18 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*(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*(sqrt(3.0)-1.0)
00058             const float G2 = 0.211324865f; /** G2 = (3.0-Math.sqrt(3.0))/6.0
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;
00093
00094             /** Calculate the contribution from the three corners
00095             float t0 = 0.5f - x0 * x0 - y0 * y0;
00096             if (t0 < 0.0f) n0 = 0.0f;
00097             else
00098             {
00099                 t0 *= t0;
00100                 n0 = t0 * t0 * grad(perm[ii + perm[jj]], x0, y0);
00101             }
00102
00103             float t1 = 0.5f - x1 * x1 - y1 * y1;
00104             if (t1 < 0.0f) n1 = 0.0f;
00105             else
00106             {
00107                 t1 *= t1;
00108                 n1 = t1 * t1 * grad(perm[ii + i1 + perm[jj + j1]], x1, y1);
00109             }
00110
00111             float t2 = 0.5f - x2 * x2 - y2 * y2;
00112             if (t2 < 0.0f) n2 = 0.0f;

```

```

00129         else
00130         {
00131             t2 *= t2;
00132             n2 = t2 * t2 * grad(perm[ii + 1 + perm[jj + 1]], x2, y2);
00133         }
00134
00135         /** Add contributions from each corner to get the final noise value.
00136         /** The result is scaled to return values in the interval [-1,1].
00137         return 40.0f * (n0 + n1 + n2); /** TODO: The scale factor is preliminary!
00138     }
00139
00140
00141     public static float Generate(float x, float y, float z)
00142     {
00143         /** Simple skewing factors for the 3D case
00144         const float F3 = 0.333333333f;
00145         const float G3 = 0.1666666667f;
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 }

```

5 Player

5.1 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](#)

5.2 PlayerLook.cs

```

00001 using UnityEngine;
00002 using BeeGame.Core;
00003
00004 namespace BeeGame.Player
00005 {
00009     public class PlayerLook : MonoBehaviour
00010     {
00011         #region Data
00012         public Transform myTransform;
00019         public Transform cameraTransform;
00023         [Range(0, 360)]
00024         public float rotationLock;
00028         public float speed = 5;
00032         float yRot = 0;
00036         float xRot = 0;
00037         #endregion
00038
00039         #region Unity Methods
00040         void Start()
00044         {
00045             Cursor.lockState = CursorLockMode.Locked;
00046             Cursor.visible = false;
00047         }
00048
00052         void Update()
00053         {
00054             /*the look will not update when a inventory GUI is open
00055             if (!THInput.isAnotherInventoryOpen)
00056             {
00057                 Look();
00058             }
00059         }
00060         #endregion
00061
00062         #region Methods
00063         void Look()
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     }
00078     #endregion
00079 }
00080 }

```

5.3 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](#)

5.4 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
00022         public PlayerInventory playerInventory;
00023
00031         public LayerMask layers;
00032         private RaycastHit hit;
00033
00040         public int selectedHotbarSlot = 27;
00041         #endregion
00042
00043         #region Unity Methods
00044         void Awake()
00045         {
00046             selector = Instantiate(selector);
00047         }
00048
00055         void FixedUpdate()
00056         {
00057             if (!IsAnotherInventoryOpen)
00058                 UpdateSelector();
00059         }
00060
00064         void Update()
00065         {
00066             if (!IsAnotherInventoryOpen)
00067             {
00068                 if (GetButtonDown("Break Block"))
00069                     BreakBlock();
00070                 if (GetButtonDown("Place"))
00071                     PlaceBlock();

```



```

00072     }
00073 }
00074 #endregion
00075
00076 #region Update
00077 void UpdateSelector()
00081 {
00082     if (Physics.Raycast(transform.position, transform.forward, out hit, 15, layers))
00083     {
00084         selector.SetActive(true);
00085         selector.transform.position = GetBlockPos(hit);
00086         /*selector.SetActive(BlockInPosition(GetBlockPos(hit),
hit.collider.GetComponent<Chunk>()));
00087     }
00088     else
00089     {
00090         selector.SetActive(false);
00091     }
00092     SelectedSlot();
00093 }
00094
00098 void SelectedSlot()
00099 {
00100     /* adds 1 to the selected slot and if that is out of range set it to the first hotbar slot
00101     if(Input.GetAxis("Mouse ScrollWheel") > 0)
00102     {
00103         selectedHotbarSlot += 1;
00104         if (selectedHotbarSlot == 36)
00105             selectedHotbarSlot = 27;
00106     }
00107     /* removes one from the hotbar selector and if the selector would be inside the inventory set
it to the last slot in the hotbar
00108     else if (Input.GetAxis("Mouse ScrollWheel") < 0)
00109     {
00110         selectedHotbarSlot -= 1;
00111         if (selectedHotbarSlot == 26)
00112             selectedHotbarSlot = 35;
00113     }
00114
00115     transform.parent.GetComponentInChildren<PlayerInventory>().SelectedSlot(
selectedHotbarSlot);
00116 }
00117 #endregion
00118
00119 #region Break/Place
00120 void BreakBlock()
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 }
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 }

```

5.5 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](#)

5.6 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 {
00010     [RequireComponent(typeof(Rigidbody))]
00011     public class PlayerMove : MonoBehaviour
00012     {
00013         #region Data
00014         public float speed = 10f;
00015         public float gravity = 9.81f;
00016         public float maxVelocity = 10f;
00017
00018         private bool canJump = false;
00019         public float jumpHeight = 2f;
00020
00021         private Rigidbody myRigidBody;
00022         #endregion
00023
00024         #region Unity Methods
00025         private void Awake()
00026         {
00027             myRigidBody = GetComponent<Rigidbody>();
00028
00029             //i want to use myown gravity and rotation
00030             myRigidBody.useGravity = false;
00031             myRigidBody.freezeRotation = true;
00032         }
00033
00034         void FixedUpdate()
00035         {
00036             //If the player is grounded it can move
00037             if (canJump)
00038             {
00039                 MovePlayer();
00040             }
00041
00042             //adds the downward force
00043             myRigidBody.AddForce(new Vector3(0, myRigidBody.mass * -gravity, 0));
00044         }
00045
00046         private void OnCollisionStay(Collision collision)
00047         {
00048             canJump = true;
00049         }
00050         #endregion
00051
00052         #region Movement Methods
00053         void MovePlayer()
00054         {
00055             //Calculate the speed we want to achieve
00056             Vector3 targetVelocity = new Vector3(THInput.GetAxis("Horizontal"), 0,
00057             THInput.GetAxis("Vertical"));
00058             targetVelocity = transform.TransformDirection(targetVelocity);
00059             targetVelocity *= speed;
00060
00061             //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
00114     float VerticalJumpSpeed()
00115     {
00116         //Gets the correct of fore required for the player to reach the desired apex
00117         //Can this be done without Square Root as that take alot of work?
00118         return Mathf.Sqrt(2 * jumpHeight * gravity);
00119     }
00120     #endregion
00121 }
00122
00123 }
00124
00125 }
00126 }

```

5.7 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](#)

5.8 SavePlayerPosition.cs

```

00001 using UnityEngine;
00002 using BeeGame.Serialization;
00003
00004 namespace BeeGame.Player
00005 {
00009     public class SavePlayerPosition : MonoBehaviour
00010     {
00014         int counter = 0;
00015
00019         void Update()
00020         {
00021             if(counter == 0)
00022             {
00023                 counter = 1000;
00024                 Serialization.Serialization.SavePlayerPosition(transform);
00025                 //print("saved player");
00026             }
00027             counter--;
00028         }
00029     }
00030 }
00031 }

```

6 Resources

6.1 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/PrefabDictionary.cs File Reference

Classes

- class [BeeGame.Core.PrefabDictionary](#)
The prefabs available to the game

Namespaces

- namespace [BeeGame.Core](#)

6.2 PrefabDictionary.cs

```
00001 using System.Collections.Generic;
00002 using UnityEngine;
00003
00004 namespace BeeGame.Core
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 }
```

6.3 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/SpriteDictionary.cs File Reference

Classes

- class [BeeGame.Core.SpriteDictionary](#)
All of the sprites available to the game

Namespaces

- namespace [BeeGame.Core](#)

6.4 SpriteDictionary.cs

```

00001 using System.Collections.Generic;
00002 using UnityEngine;
00003
00004 namespace BeeGame.Core
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 }

```

6.5 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/BeeDictionary.cs File Reference

Classes

- class [BeeGame.Core.BeeDictionarys](#)

Namespaces

- namespace [BeeGame.Core](#)

6.6 BeeDictionary.cs

```

00001 using System.Collections.Generic;
00002 using BeeGame.Core.Enums;
00003 using UnityEngine;
00004
00005 namespace BeeGame.Core
00006 {
00007     public static class BeeDictionarys
00008     {
00009         #region Bee Colours
00010         private static Dictionary<HoneyCombType, Color> honeyCombColour = new Dictionary<HoneyCombType,
00014         Color>()
00015         {
00016             {HoneyCombType.HONEY, CombColour(255, 164, 56) },
00017             {HoneyCombType.ICEY, CombColour(78, 231, 231) }
00018         };
00019
00028         private static Color CombColour(float r, float g, float b, float a = 255f)
00029         {
00030             return new Color(r / 255f, g / 255f, b / 255f);
00031         }
00032
00038         public static Color GetCombColour(HoneyCombType type)
00039         {
00040             honeyCombColour.TryGetValue(type, out var temp);
00041
00042             if (temp == null)
00043                 return new Color(1, 0, 0);
00044
00045             return temp;
00046         }
00047         #endregion
00048     }
00049 }

```

6.7 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](#)

6.8 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
00016     /** This class was auto-generated by the StronglyTypedResourceBuilder
00017     /** class via a tool like ResGen or Visual Studio.
00018     /** To add or remove a member, edit your .ResX file then rerun ResGen
00019     /** with the /str option, or rebuild your VS project.
00020     [global::System.CodeDom.Compiler.GeneratedCodeAttribute("
00021     System.Resources.Tools.StronglyTypedResourceBuilder", "4.0.0.0")]
00022     [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
00023     [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
00024     internal class Resources {
00025
00026         private static global::System.Resources.ResourceManager
00027         resourceMan;
00028
00029         private static global::System.Globalization.CultureInfo resourceCulture;
00030
00031         [global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance", "
00032         CA1811:AvoidUncalledPrivateCode")]
00033         internal Resources() {
00034         }
00035
00036         [global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.
00037         EditorBrowsableState.Advanced)]
00038         internal static global::System.Resources.ResourceManager ResourceManager {
00039             get {
00040                 if (object.ReferenceEquals(resourceMan, null)) {
00041                     global::System.Resources.ResourceManager temp = new global::System.Resources.
00042                     ResourceManager("BeeGame.Resources.Resources", typeof(Resources).Assembly);
00043                     resourceMan = temp;
00044                 }
00045                 return resourceMan;
00046             }
00047         }
00048
00049         [global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.
00050         EditorBrowsableState.Advanced)]
00051         internal static global::System.Globalization.CultureInfo Culture {
00052             get {
00053                 return resourceCulture;
00054             }
00055             set {
00056                 resourceCulture = value;
00057             }
00058         }
00059
00060         internal static byte[] Prefabs {
00061             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
00118     return sprites;
00119 }
00120
00121 internal static Dictionary<string, GameObject> GetPrefabs()
00122 {
00123     string[] splitCharacters = new string[] { ",", " " };
00124     object obj = ResourceManager.GetObject("Prefabs", resourceCulture);
00125
00126     string text = System.Text.Encoding.Default.GetString((byte[])obj);
00127     text = text.Remove(0, 3);
00128     string lineText = "";
00129     string[] splitText;
00130     Dictionary<string, GameObject> objects = new Dictionary<string, GameObject>();
00131
00132     for (int i = 0; i < text.Length; i++)
00133     {
00134         if (text[i] != '\n')
00135         {
00136             lineText += text[i];
00137         }
00138         else
00139         {
00140             splitText = lineText.Split(splitCharacters, StringSplitOptions.RemoveEmptyEntries);
00141             lineText = "";
00142             objects.Add(splitText[0], UnityEngine.Resources.Load("Prefabs/" + splitText[
00143 1].Remove(splitText[1].Length - 1, 1)) as GameObject);
00144         }
00145
00146         splitText = lineText.Split(splitCharacters, StringSplitOptions.RemoveEmptyEntries);
00147         lineText = "";
00148         objects.Add(splitText[0], UnityEngine.Resources.Load("Prefabs/" + splitText[1]) as
00149 GameObject);
00150     }
00151
00152     return objects;
00153 }
00154 }
00155 }

```

6.9 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](#)

6.10 LoadResources.cs

```
00001 using UnityEngine;
00002 using BeeGame.Core;
00003
00004 namespace BeeGame
00005 {
00009     public class LoadResources : MonoBehaviour
00010     {
00014         void Awake()
00015         {
00016             Serialization.Serialization.MakeDirectorys();
00017             Serialization.Serialization.LoadPlayerPosition(GameObject.Find("Player").GetComponent<Transform
00018 >());
00019
00020             SpriteDictionary.LoadSprites();
00021             PrefabDictionary.LoadPrefabs();
00022         }
00023     }
00024 }
```

7 Unity Type & Method Replacements

7.1 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](#)

7.2 THInput.cs

```

00001 using System;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004
00005 namespace BeeGame.Core
00006 {
00010     public static class THInput
00011     {
00015         private static Dictionary<string, object> inputButtons = new Dictionary<string, object>()
00016         {
00017             {"Forward" , KeyCode.W},
00018             {"Backward", KeyCode.S },
00019             {"Right", KeyCode.D },
00020             {"Left", KeyCode.A },
00021             {"Player Inventory", KeyCode.E },
00022             {"Quest Book", KeyCode.Mouse1 },
00023             {"Interact", KeyCode.Mouse1 },
00024             {"Place", KeyCode.Mouse1 },
00025             {"Break Block", KeyCode.Mouse0 },
00026             {"Close Menu/Inventory", new KeyCode[2] { KeyCode.Escape, KeyCode.E } },
00027             {"Jump", KeyCode.Space }
00028         };
00029
00033         public static bool isAnotherInventoryOpen;
00034
00038         public static bool blockInventoryJustClosed;
00039
00045         public static bool GetButtonDown(string button)
00046         {
00047             if (!inputButtons.ContainsKey(button))
00048             {
00049                 throw new Exception("Input Manager: Key button name not defined: " + button);
00050             }
00051
00052             switch (inputButtons[button])
00053             {
00054                 case KeyCode[] array:
00055                     /*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
00056                     foreach (var item in array)
00057                     {
00058                         if (Input.GetKeyDown(item))
00059                         {
00060                             return true;
00061                         }
00062                     }
00063
00064                     return false;
00065                 default:
00066                     return Input.GetKeyDown((KeyCode)inputButtons[button]);
00067             }
00068         }
00069
00075         public static bool GetButton(string button)
00076         {
00077             if (!inputButtons.ContainsKey(button))
00078             {
00079                 throw new Exception("Input Manager: Key button name not defined: " + button);
00080             }
00081
00082             switch (inputButtons[button])
00083             {
00084                 case KeyCode[] array:
00085                     /*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
00086                     foreach (var item in array)
00087                     {
00088                         if (Input.GetKey(item))
00089                         {
00090                             return true;
00091                         }
00092                     }
00093
00094                     return false;
00095                 default:
00096                     return Input.GetKey((KeyCode)inputButtons[button]);
00097             }
00098         }
00099
00105         public static bool GetButtonUp(string button)
00106         {
00107             if (!inputButtons.ContainsKey(button))
00108             {
00109                 throw new Exception("Input Manager: Key button name not defined: " + button);

```

```

00110         }
00111
00112         switch (inputButtons[button])
00113         {
00114             case KeyCode[] array:
00115                 /*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
00116                 foreach (var item in array)
00117                 {
00118                     if (Input.GetKeyUp(item))
00119                     {
00120                         return true;
00121                     }
00122                 }
00123
00124                 return false;
00125             default:
00126                 return Input.GetKeyUp((KeyCode)inputButtons[button]);
00127         }
00128     }
00129
00135     public static int.GetAxis(string axis)
00136     {
00137         int returnAxis = 0;
00138
00139         if (axis == "Horizontal")
00140         {
00141             if (GetButton("Right"))
00142             {
00143                 returnAxis += 1;
00144             }
00145
00146             if (GetButton("Left"))
00147             {
00148                 returnAxis -= 1;
00149             }
00150         }
00151         else if (axis == "Vertical")
00152         {
00153             if (GetButton("Forward"))
00154             {
00155                 returnAxis += 1;
00156             }
00157
00158             if (GetButton("Backward"))
00159             {
00160                 returnAxis -= 1;
00161             }
00162         }
00163
00164         return returnAxis;
00165     }
00166 }
00167 }

```

7.3 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](#)

7.4 THVector2.cs

```

00001 using System;
00002 using UnityEngine;
00003
00004 namespace BeeGame.Core
00005 {
00006     [Serializable]
00010     public struct THVector2
00011     {
00012         #region Data
00013         public float x;
00020         public float y;
00021         #endregion
00022
00023         #region Constructor
00024         public THVector2(float x, float y)
00030         {
00031             this.x = x;
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 }

```

7.5 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](#)

7.6 THVector3.cs

```

00001 using System;
00002 using UnityEngine;
00003
00004 namespace BeeGame.Core
00005 {
00009     [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.
00079 y), 2) + Math.Pow((a.z - b.z), 2));
00080         }
00081         #endregion
00082
00083         #region Overrides
00084         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 }

```

8 Misc

8.1 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](#)

8.2 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 MakeDirectories()
00040         {
00041             savePath = $"{Application.dataPath}/{saveFolderName}/{worldName}";
00042         }
00043     }
00044 }

```

```

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 = { "/", "\\ " };
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     #endregion
00098
00099     #region Inventorys
00100     public static void SerializeInventory(Inventory.Inventory inventory, string inventoryName)
00101     {
00102         string inventorySavePath = $"{savePath}/Inventorys";
00103
00104         if (!Directory.Exists(inventorySavePath))
00105             Directory.CreateDirectory(inventorySavePath);
00106
00107         SaveFile(inventory.GetAllItems(), $"{inventorySavePath}/{inventoryName}.dat");
00108     }
00109
00110     public static void DeSerializeInventory(Inventory.Inventory inventory,
string inventoryName)
00111     {
00112         /* make the path
00113         string inventorySavePath = $"{savePath}/Inventorys/{inventoryName}.dat";
00114
00115         /* checks that the file exists
00116         if (!File.Exists(inventorySavePath))
00117             return;
00118
00119         inventory.SetAllItems((ItemsInInventory)LoadFile($"{inventorySavePath}"));
00120     }
00121     #endregion
00122
00123     #region Chunk
00124     public static void SaveChunk(Chunk chunk)
00125     {
00126         /* saves the blocks
00127         SaveChunk save = new SaveChunk(chunk.blocks);
00128
00129         /* if no block was changed return early
00130         if (save.blocks.Count == 0)

```



```
00157         return;
00158
00159         /** otherwise save the file
00160         string saveFile = $"{savePath}/{FileName(chunk.chunkWorldPos)}.dat";
00161
00162         SaveFile(save, saveFile);
00163     }
00164
00170     public static bool LoadChunk(Chunk chunk)
00171     {
00172         /** gets the save file
00173         string saveFile = $"{savePath}/{FileName(chunk.chunkWorldPos)}.dat";
00174
00175         /** if the file does not exist return false
00176         if (!File.Exists(saveFile))
00177             return false;
00178
00179         /** set all of the changed blocks in the chunk
00180         SaveChunk save = (SaveChunk)LoadFile(saveFile);
00181
00182         foreach (var block in save.blocks)
00183         {
00184             chunk.blocks[block.Key.x, block.Key.y, block.Key.z] = block.Value;
00185         }
00186
00187         return true;
00188     }
00189
00195     public static string FileName(ChunkWorldPos pos)
00196     {
00197         return $"{pos.x}, {pos.y}, {pos.z}";
00198     }
00199 #endregion
00200
00201 #region Save/Load Files
00202 private static void SaveFile(object obj, string file)
00203 {
00204     BinaryFormatter bf = new BinaryFormatter();
00205     FileStream fs = new FileStream(file, FileMode.OpenOrCreate);
00206
00207     try
00208     {
00209         bf.Serialize(fs, obj);
00210     }
00211     catch (SerializationException e)
00212     {
00213         Debug.Log($"Serialization Exception: {e}");
00214         throw new SerializationException();
00215     }
00216     finally
00217     {
00218         fs.Close();
00219     }
00220 }
00221
00232 private static object LoadFile(string file)
00233 {
00234     BinaryFormatter bf = new BinaryFormatter();
00235     FileStream fs = new FileStream(file, FileMode.Open);
00236
00237     try
00238     {
00239         return bf.Deserialize(fs);
00240     }
00241     catch (SerializationException e)
00242     {
00243         Debug.Log($"Deserialization Exception {e}");
00244         throw new SerializationException();
00245     }
00246     finally
00247     {
00248         fs.Close();
00249     }
00250 }
00251 #endregion
00252 }
00253 }
```

Classes

- class [BeeGame.Core.Extensions](#)

Namespaces

- namespace [BeeGame.Core](#)

8.4 Extensions.cs

```

00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using System.Reflection;
00005 using System.Text;
00006
00007 namespace BeeGame.Core
00008 {
00009     public static class Extensions
00010     {
00011         public static T CloneObject<T>(this T obj)
00012         {
00013             /** gets the tyoe of the given object
00014             Type typeSource = obj.GetType();
00015
00016             /** makes a new object of type T
00017             T objTarget = (T)Activator.CreateInstance(typeSource);
00018
00019             /** gets the properties in T
00020             PropertyInfo[] propertyInfo = typeSource.GetProperties(BindingFlags.Public | BindingFlags.
NonPublic | BindingFlags.Instance);
00021
00022             /** applies the properties in T to the new type T object
00023             foreach (var property in propertyInfo)
00024             {
00025                 if (property.CanWrite)
00026                 {
00027                     /** if the property is a value just set it
00028                     if (property.PropertyType.IsValueType || property.PropertyType.IsEnum || property.
PropertyType.Equals(typeof(string)))
00029                     {
00030                         property.SetValue(objTarget, property.GetValue(obj, null), null);
00031                     }
00032                     else
00033                     {
00034                         /** if the property is not a value type this function will need to be called
00035                         recursively as it could also have non value type variables
00036                         object propertyValue = property.GetValue(obj, null);
00037
00038                         if (propertyValue == null)
00039                         {
00040                             property.SetValue(obj, null, null);
00041                         }
00042                         else
00043                         {
00044                             property.SetValue(obj, propertyValue.CloneObject(), null);
00045                         }
00046                     }
00047                 }
00048             }
00049             return objTarget;
00050         }
00051     }
00052 }

```

8.5 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.ItemType { BeeGame.Core.Enums.ItemType.ITEM, BeeGame.Core.Enums.ItemType.BEE }

The item types

- enum BeeGame.Core.Enums.HoneyCombType { BeeGame.Core.Enums.HoneyCombType.HONEY, BeeGame.Core.Enums.HoneyCombType.ICEY }

Honey Comb Types

- enum BeeGame.Core.Enums.BeeSpecies { BeeGame.Core.Enums.BeeSpecies.FOREST, BeeGame.Core.Enums.BeeSpecies.MEADOWS, BeeGame.Core.Enums.BeeSpecies.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.PRINCESS, BeeGame.Core.Enums.BeeType.DRONE }

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

8.6 Enums.cs

```

00001 namespace BeeGame.Core.Enums
00002 {
00006     public enum ItemType
00007     {
00008         ITEM, BEE
00009     };
00010
00014     public enum HoneyCombType
00015     {
00016         HONEY, ICEY
00017     };
00018
00019     #region BeeStuff
00020     public enum BeeSpecies
00024     {
00025         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
00026     };
00027
00031     public enum BeeType
00032     {
00033         QUEEN, PRINCESS, DRONE
00034     };
00035
00039     public enum BeeTempPreference
00040     {
00041         FROZEN, COLD, TEMPERATE, HOT, HELL
00042     };
00043
00047     public enum BeeLifeSpan
00048     {
00049         HUMMINGBIRD, SHORTEST, SHORT, NORMAL, LONG,
LONGEST, SEATURTLE
00050     };
00051
00055     public enum BeeProductionSpeed
00056     {
00057         SLOW, NORMAL, FAST
00058     };
00059
00063     public enum BeeEffect
00064     {
00065         NONE, POSION
00066     };
00067
00071     public enum BeeHumidityPreference
00072     {
00073         ARID, DRY, TEMPERATE, MOIST, HUMID
00074     };
00075     #endregion BeeStuff
00076
00080     public enum Direction
00081     {
00082         NORTH, EAST, SOUTH, WEST, UP, DOWN
00083     };
00084 }

```

8.7 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/SpawnItem.cs File Reference

Classes

- class [BeeGame.SpawnItem](#)

Namespaces

- namespace [BeeGame](#)

8.8 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 HoneyComb(
HoneyCombType.ICEY);
00018
00019             go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
transform.position, Quaternion.identity) as GameObject;
00020             go.GetComponent<ItemGameObject>().item = new HoneyComb(
HoneyCombType.HONEY);
00021         }
00022
00023         private void OnDrawGizmos()
00024         {
00025             //Gizmos.color = Color.green;
00026             //Gizmos.DrawSphere(transform.position, 0.5f);
00027         }
00028     }
00029 }
```

8.9 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/test.cs File Reference

Classes

- class [BeeGame.Test](#)

Namespaces

- namespace [BeeGame](#)

8.10 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
00008 namespace BeeGame
00009 {
00010     public class Test : MonoBehaviour
00011     {
00012         private void Start()
00013         {
00014             Instantiate(BeeGame.Core.PrefabDictionary.
GetPrefab("Selector"));
00015         }
00016     }
00017 }
```

Index

- AddItem
 - BeeGame::Inventory::ItemsInInventory, [72](#)
- AddItemToInventory
 - BeeGame::Inventory::Inventory, [67](#)
- AddItemToSlots
 - BeeGame::Inventory::Inventory, [67](#)
- AddQuadTriangles
 - BeeGame::Terrain::Chunks::MeshData, [98](#)
- AddToSlot
 - BeeGame::Inventory::InventorySlot, [74](#)
- AddTriangle
 - BeeGame::Terrain::Chunks::MeshData, [99](#)
- AddVertices
 - BeeGame::Terrain::Chunks::MeshData, [99](#)
- Air
 - BeeGame::Blocks::Air, [42](#)
- Apiary
 - BeeGame::Blocks::Apiary, [56](#)
- applyCollisionMesh
 - BeeGame::Terrain::Chunks::Chunk, [95](#)
- ApplyCollsionMeshToNearbyChunks
 - BeeGame::Terrain::Chunks::LoadChunks, [106](#)
- ApplyPlayerItems
 - BeeGame::Inventory::ChestInventory, [86](#)
- Awake
 - BeeGame::Inventory::Player_Inventory::Player↔Inventory, [81](#)
 - BeeGame::LoadResources, [154](#)
 - BeeGame::Player::PlayerMove, [149](#)
 - BeeGame::Player::Selector, [145](#)
- Bedrock
 - BeeGame::Blocks::Bedrock, [63](#)
- BeeEffect
 - BeeGame::Core::Enums, [3](#)
- BeeGame, [1](#)
- BeeGame.Blocks, [2](#)
- BeeGame.Blocks.Air, [41](#)
- BeeGame.Blocks.Apiary, [55](#)
- BeeGame.Blocks.Bedrock, [62](#)
- BeeGame.Blocks.Block, [33](#)
- BeeGame.Blocks.Chest, [51](#)
- BeeGame.Blocks.Dirt, [44](#)
- BeeGame.Blocks.Grass, [47](#)
- BeeGame.Blocks.Leaves, [59](#)
- BeeGame.Blocks.Wood, [57](#)
- BeeGame.Core, [2](#)
- BeeGame.Core.BeeDictionarys, [158](#)
- BeeGame.Core.Enums, [3](#)
- BeeGame.Core.Extensions, [190](#)
- BeeGame.Core.PrefabDictionary, [155](#)
- BeeGame.Core.SpriteDictionary, [156](#)
- BeeGame.Core.THInput, [164](#)
- BeeGame.Core.THVector2, [169](#)
- BeeGame.Core.THVector3, [176](#)
- BeeGame.Inventory, [8](#)
- BeeGame.Inventory.ChestInventory, [85](#)
- BeeGame.Inventory.Inventory, [65](#)
- BeeGame.Inventory.InventorySlot, [73](#)
- BeeGame.Inventory.ItemsInInventory, [71](#)
- BeeGame.Inventory.Player_Inventory, [9](#)
- BeeGame.Inventory.Player_Inventory.PlayerInventory, [80](#)
- BeeGame.Items, [9](#)
- BeeGame.Items.ApplyColour, [32](#)
- BeeGame.Items.HoneyComb, [26](#)
- BeeGame.Items.Item, [11](#)
- BeeGame.Items.ItemGameObject, [30](#)
- BeeGame.Items.Tile, [40](#)
- BeeGame.LoadResources, [154](#)
- BeeGame.Player, [9](#)
- BeeGame.Player.PlayerLook, [141](#)
- BeeGame.Player.PlayerMove, [148](#)
- BeeGame.Player.SavePlayerPosition, [153](#)
- BeeGame.Player.Selector, [144](#)
- BeeGame.Resources, [9](#)
- BeeGame.Resources.Resources, [160](#)
- BeeGame.Serialization, [9](#)
- BeeGame.Serialization.Serialization, [191](#)
- BeeGame.SpawnItem, [199](#)
- BeeGame.Terrain, [10](#)
- BeeGame.Terrain.ChunkWorldPos, [101](#)
- BeeGame.Terrain.Chunks, [10](#)
- BeeGame.Terrain.Chunks.Chunk, [89](#)
- BeeGame.Terrain.Chunks.LoadChunks, [105](#)
- BeeGame.Terrain.Chunks.MeshData, [97](#)
- BeeGame.Terrain.Chunks.SaveChunk, [110](#)
- BeeGame.Terrain.LandGeneration, [10](#)
- BeeGame.Terrain.LandGeneration.Noise, [11](#)
- BeeGame.Terrain.LandGeneration.Noise.SimplexNoise, [134](#)
- BeeGame.Terrain.LandGeneration.Terrain, [117](#)
- BeeGame.Terrain.LandGeneration.TerrainGeneration, [126](#)
- BeeGame.Terrain.LandGeneration.World, [112](#)
- BeeGame.Test, [200](#)
- BeeGame::Blocks::Air
 - Air, [42](#)
 - BlockData, [42](#)
 - BreakBlock, [42](#)
 - GetHashCode, [43](#)
 - IsSolid, [43](#)
 - ToString, [43](#)
- BeeGame::Blocks::Apiary
 - Apiary, [56](#)
 - GetHashCode, [56](#)
 - ToString, [56](#)
- BeeGame::Blocks::Bedrock
 - Bedrock, [63](#)
 - BreakBlock, [63](#)

- GetHashCode, 64
- TexturePosition, 64
- ToString, 65
- BeeGame::Blocks::Block
 - Block, 34, 35
 - BlockData, 35
 - BreakBlock, 36
 - breakable, 40
 - changed, 40
 - GetHashCode, 37
 - GetItemSprite, 37
 - InteractWithBlock, 37
 - IsSolid, 38
 - ToString, 39
 - UpdateBlock, 39
- BeeGame::Blocks::Chest
 - BlockData, 52
 - BreakBlock, 53
 - Chest, 52
 - GetGameObject, 53
 - GetHashCode, 54
 - InteractWithBlock, 54
 - myGameobject, 55
 - TexturePosition, 54
 - ToString, 55
- BeeGame::Blocks::Dirt
 - Dirt, 45
 - GetHashCode, 45
 - GetItemSprite, 45
 - TexturePosition, 46
 - ToString, 47
- BeeGame::Blocks::Grass
 - GetHashCode, 48
 - GetItemName, 49
 - GetItemSprite, 49
 - Grass, 48
 - TexturePosition, 49
 - ToString, 50
 - UpdateBlock, 50
- BeeGame::Blocks::Leaves
 - GetHashCode, 60
 - GetItemSprite, 60
 - IsSolid, 61
 - Leaves, 60
 - TexturePosition, 61
 - ToString, 62
- BeeGame::Blocks::Wood
 - GetHashCode, 58
 - GetItemSprite, 58
 - TexturePosition, 58
 - ToString, 59
 - Wood, 58
- BeeGame::Core::BeeDictionarys
 - CombColour, 159
 - GetCombColour, 159
 - honeyCoubColour, 160
- BeeGame::Core::Enums
 - BeeEffect, 3
 - BeeHumidityPreference, 4
 - BeeLifeSpan, 4
 - BeeProductionSpeed, 5
 - BeeSpecies, 5
 - BeeTempPreference, 6
 - BeeType, 7
 - Direction, 7
 - HoneyCombType, 7
 - ItemType, 8
- BeeGame::Core::Extensions
 - CloneObject< T >, 190
- BeeGame::Core::PrefabDictionary
 - GetPrefab, 155
 - LoadPrefabs, 156
 - prefabDictionary, 156
- BeeGame::Core::SpriteDictionary
 - GetSprite, 157
 - itemSpriteDictionary, 158
 - LoadSprites, 157
- BeeGame::Core::THInput
 - blockInventoryJustClosed, 168
 - GetAxis, 165
 - GetButton, 166
 - GetButtonDown, 166
 - GetButtonUp, 167
 - inputButtons, 168
 - isAnotherInventoryOpen, 168
- BeeGame::Core::THVector2
 - Equals, 171
 - GetHashCode, 171
 - operator THVector2, 171
 - operator Vector2, 172
 - operator!=, 172
 - operator*, 172, 173
 - operator+, 173
 - operator-, 174
 - operator/, 174, 175
 - operator==, 175
 - THVector2, 170
 - ToString, 175
 - x, 176
 - y, 176
- BeeGame::Core::THVector3
 - Distance, 179
 - Equals, 180
 - GetHashCode, 180
 - operator Quaternion, 180
 - operator THVector3, 181
 - operator Vector3, 181
 - operator!=, 182
 - operator*, 182, 183
 - operator+, 184, 185
 - operator-, 185, 186
 - operator/, 187, 188
 - operator==, 188
 - THVector3, 178, 179
 - ToString, 189
 - x, 189

- y, [189](#)
 - z, [189](#)
- BeeGame::Inventory::ChestInventory
 - ApplyPlayerItems, [86](#)
 - inventory, [88](#)
 - inventoryPosition, [88](#)
 - inventorySize, [88](#)
 - playerInventory, [88](#)
 - SetChestInventory, [86](#)
 - SetPlayerItems, [86](#)
 - ToggleInventory, [87](#)
 - Update, [87](#)
- BeeGame::Inventory::Inventory
 - AddItemToInventory, [67](#)
 - AddItemToSlots, [67](#)
 - floatingItem, [70](#)
 - GetAllItems, [67](#)
 - inventoryName, [70](#)
 - InventorySet, [68](#)
 - items, [70](#)
 - PutItemsInSlots, [68](#)
 - SaveInv, [68](#)
 - SetAllItems, [69](#)
 - SetInventorySize, [69](#)
 - slots, [70](#)
 - thisInventoryOpen, [70](#)
 - UpdateBase, [70](#)
- BeeGame::Inventory::InventorySlot
 - AddToSlot, [74](#)
 - CheckFloatingItem, [75](#)
 - item, [79](#)
 - itemText, [79](#)
 - myInventory, [79](#)
 - OnDisable, [75](#)
 - OnPointerClick, [75](#)
 - OnPointerEnter, [77](#)
 - OnPointerExit, [77](#)
 - selectedSlot, [80](#)
 - slotIndex, [80](#)
 - SplitStack, [77](#)
 - SwapItems, [78](#)
 - Update, [78](#)
 - UpdateIcon, [78](#)
- BeeGame::Inventory::ItemsInInventory
 - AddItem, [72](#)
 - ItemsInInventory, [71](#)
 - itemsInInventory, [73](#)
- BeeGame::Inventory::Player_Inventory::PlayerInventory
 - Awake, [81](#)
 - GetItemFromHotBar, [81](#)
 - OpenPlayerInventory, [82](#)
 - PickupItem, [82](#)
 - playerInventory, [85](#)
 - RemoveItemFromInventory, [83](#)
 - SelectedSlot, [83](#)
 - SetPlayerInventory, [84](#)
 - Update, [84](#)
- BeeGame::Items::ApplyColour
 - colour, [33](#)
 - objects, [33](#)
 - Start, [32](#)
- BeeGame::Items::HoneyComb
 - CombColour, [29](#)
 - GetGameObject, [27](#)
 - GetHashCode, [28](#)
 - GetItemID, [28](#)
 - GetItemSprite, [28](#)
 - HoneyComb, [27](#)
 - type, [29](#)
- BeeGame::Items::Item
 - Clone, [14](#)
 - Equals, [14](#)
 - FaceDataDown, [15](#)
 - FaceDataEast, [15](#)
 - FaceDataNorth, [16](#)
 - FaceDataSouth, [17](#)
 - FaceDataUp, [18](#)
 - FaceDataWest, [19](#)
 - FaceUVs, [19](#)
 - GetGameObject, [20](#)
 - GetHashCode, [20](#)
 - GetItemID, [21](#)
 - GetItemName, [21](#)
 - GetItemSprite, [21](#)
 - Item, [13](#)
 - ItemMesh, [22](#)
 - itemName, [24](#)
 - itemStackCount, [24](#)
 - maxStackCount, [25](#)
 - operator!=, [22](#)
 - operator==, [23](#)
 - placeable, [25](#)
 - TexturePosition, [23](#)
 - tileSize, [25](#)
 - ToString, [24](#)
 - usesGameObject, [25](#)
- BeeGame::Items::ItemGameObject
 - go, [31](#)
 - item, [31](#)
 - MakeMesh, [30](#)
 - Start, [30](#)
 - Update, [31](#)
- BeeGame::Items::Tile
 - x, [41](#)
 - y, [41](#)
- BeeGame::LoadResources
 - Awake, [154](#)
- BeeGame::Player::PlayerLook
 - cameraTransform, [143](#)
 - Look, [142](#)
 - myTransform, [143](#)
 - rotationLock, [143](#)
 - speed, [143](#)
 - Start, [142](#)
 - Update, [142](#)
 - xRot, [143](#)

- yRot, [143](#)
- BeeGame::Player::PlayerMove
 - Awake, [149](#)
 - canJump, [151](#)
 - FixedUpdate, [150](#)
 - gravity, [151](#)
 - jumpHeight, [152](#)
 - maxVelocity, [152](#)
 - MovePlayer, [150](#)
 - myRigidBody, [152](#)
 - OnCollisionStay, [150](#)
 - speed, [152](#)
 - VerticalJumpSpeed, [151](#)
- BeeGame::Player::SavePlayerPosition
 - counter, [154](#)
 - Update, [153](#)
- BeeGame::Player::Selector
 - Awake, [145](#)
 - BreakBlock, [145](#)
 - FixedUpdate, [145](#)
 - hit, [147](#)
 - layers, [147](#)
 - PlaceBlock, [146](#)
 - playerInventory, [148](#)
 - selectedHotbarSlot, [148](#)
 - SelectedSlot, [146](#)
 - selector, [148](#)
 - Update, [146](#)
 - UpdateSelector, [147](#)
- BeeGame::Resources::Resources
 - Culture, [163](#)
 - GetPrefabs, [161](#)
 - GetSprites, [162](#)
 - Prefabs, [163](#)
 - resourceCulture, [163](#)
 - resourceMan, [163](#)
 - ResourceManager, [163](#)
 - Resources, [161](#)
 - Sprites, [164](#)
- BeeGame::Serialization::Serialization
 - DeSerializeInventory, [193](#)
 - DeleteFile, [192](#)
 - FileName, [193](#)
 - LoadChunk, [194](#)
 - LoadFile, [194](#)
 - LoadPlayerPosition, [195](#)
 - MakeDirectorys, [195](#)
 - SaveChunk, [196](#)
 - SaveFile, [196](#)
 - saveFolderName, [198](#)
 - savePath, [198](#)
 - SavePlayerPosition, [197](#)
 - SerializeInventory, [197](#)
 - worldName, [198](#)
- BeeGame::SpawnItem
 - OnDrawGizmos, [199](#)
 - Start, [199](#)
- BeeGame::Terrain::ChunkWorldPos
 - ChunkWorldPos, [102](#)
 - Equals, [102](#)
 - GetHashCode, [102](#)
 - operator ChunkWorldPos, [103](#)
 - operator THVector3, [103](#)
 - ToString, [104](#)
 - x, [104](#)
 - y, [104](#)
 - z, [104](#)
- BeeGame::Terrain::Chunks::Chunk
 - applyCollisionMesh, [95](#)
 - blocks, [95](#)
 - chunkSize, [95](#)
 - chunkWorldPos, [95](#)
 - ColliderMesh, [91](#)
 - filter, [96](#)
 - GetBlock, [91](#)
 - InRange, [92](#)
 - mesh, [96](#)
 - meshCollider, [96](#)
 - RenderMesh, [92](#)
 - rendered, [96](#)
 - SetBlock, [93](#)
 - SetBlocksUnmodified, [93](#)
 - Start, [94](#)
 - Update, [94](#)
 - update, [96](#)
 - UpdateChunk, [94](#)
 - updateCollisionMesh, [97](#)
 - world, [97](#)
- BeeGame::Terrain::Chunks::LoadChunks
 - ApplyCollisionMeshToNearbyChunks, [106](#)
 - BuildChunk, [106](#)
 - buildList, [109](#)
 - chunkPositions, [109](#)
 - DeleteChunks, [107](#)
 - FindChunksToLoad, [107](#)
 - LoadAndRenderChunks, [108](#)
 - nearbyChunks, [109](#)
 - Start, [108](#)
 - timer, [110](#)
 - Update, [109](#)
 - world, [110](#)
- BeeGame::Terrain::Chunks::MeshData
 - AddQuadTriangles, [98](#)
 - AddTriangle, [99](#)
 - AddVertices, [99](#)
 - colTris, [100](#)
 - colVerts, [100](#)
 - done, [100](#)
 - shareMeshes, [100](#)
 - tris, [100](#)
 - uv, [100](#)
 - verts, [101](#)
- BeeGame::Terrain::Chunks::SaveChunk
 - blocks, [111](#)
 - SaveChunk, [111](#)

- BeeGame::Terrain::LandGeneration::Noise::Simplex↔
 - Noise
 - FastFloor, [135](#)
 - Generate, [135–137](#)
 - grad, [138](#), [139](#)
 - Mod, [140](#)
 - perm, [140](#)
- BeeGame::Terrain::LandGeneration::Terrain
 - BlockInPosition, [118](#)
 - GetBlock, [119](#)
 - GetBlockPos, [120](#), [121](#)
 - GetBlockPosFromRayCast, [121](#)
 - GetChunk, [122](#)
 - Round, [122](#)
 - RoundXZ, [123](#)
 - RoundY, [124](#)
 - SetBlock, [125](#)
 - world, [126](#)
- BeeGame::Terrain::LandGeneration::TerrainGeneration
 - caveFrequency, [131](#)
 - caveSize, [132](#)
 - ChunkGen, [127](#)
 - ChunkGenThread, [128](#)
 - CreateTree, [128](#)
 - dirtBaseHeight, [132](#)
 - dirtNoise, [132](#)
 - dirtNoiseHeight, [132](#)
 - GenChunkColum, [129](#)
 - GetNoise, [130](#)
 - SetBlock, [131](#)
 - stoneBaseHeight, [132](#)
 - stoneBaseNoise, [133](#)
 - stoneBaseNoiseHeight, [133](#)
 - stoneMinHeight, [133](#)
 - stoneMountainFrequency, [133](#)
 - stoneMountainHeight, [133](#)
 - treeDensity, [134](#)
 - treeFrequency, [134](#)
- BeeGame::Terrain::LandGeneration::World
 - chunkHasMadeCollisionMesh, [117](#)
 - chunkPrefab, [117](#)
 - chunks, [117](#)
 - CreateChunk, [112](#)
 - DestroyChunk, [113](#)
 - GetBlock, [114](#)
 - GetChunk, [114](#)
 - SetBlock, [115](#)
 - UpdateIfEqual, [116](#)
- BeeGame::Test
 - Start, [200](#)
- BeeHumidityPreference
 - BeeGame::Core::Enums, [4](#)
- BeeLifeSpan
 - BeeGame::Core::Enums, [4](#)
- BeeProductionSpeed
 - BeeGame::Core::Enums, [5](#)
- BeeSpecies
 - BeeGame::Core::Enums, [5](#)
- BeeTempPreference
 - BeeGame::Core::Enums, [6](#)
- BeeType
 - BeeGame::Core::Enums, [7](#)
- Block
 - BeeGame::Blocks::Block, [34](#), [35](#)
- BlockData
 - BeeGame::Blocks::Air, [42](#)
 - BeeGame::Blocks::Block, [35](#)
 - BeeGame::Blocks::Chest, [52](#)
- BlockInPosition
 - BeeGame::Terrain::LandGeneration::Terrain, [118](#)
- blockInventoryJustClosed
 - BeeGame::Core::THInput, [168](#)
- blocks
 - BeeGame::Terrain::Chunks::Chunk, [95](#)
 - BeeGame::Terrain::Chunks::SaveChunk, [111](#)
- BreakBlock
 - BeeGame::Blocks::Air, [42](#)
 - BeeGame::Blocks::Bedrock, [63](#)
 - BeeGame::Blocks::Block, [36](#)
 - BeeGame::Blocks::Chest, [53](#)
 - BeeGame::Player::Selector, [145](#)
- breakable
 - BeeGame::Blocks::Block, [40](#)
- BuildChunk
 - BeeGame::Terrain::Chunks::LoadChunks, [106](#)
- buildList
 - BeeGame::Terrain::Chunks::LoadChunks, [109](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↔
 - Code/BeeGame/BeeGame/Blocks/Air.cs, [209](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↔
 - Code/BeeGame/BeeGame/Blocks/Apiary.cs, [213](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↔
 - Code/BeeGame/BeeGame/Blocks/Bedrock.↔
cs, [215](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↔
 - Code/BeeGame/BeeGame/Blocks/Block.cs, [207](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↔
 - Code/BeeGame/BeeGame/Blocks/Chest.cs, [211](#), [212](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↔
 - Code/BeeGame/BeeGame/Blocks/Dirt.cs, [209](#), [210](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↔
 - Code/BeeGame/BeeGame/Blocks/Grass.cs, [210](#), [211](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↔
 - Code/BeeGame/BeeGame/Blocks/Leaves.cs, [214](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↔
 - Code/BeeGame/BeeGame/Blocks/Wood.cs, [213](#), [214](#)
- C:/Users/Toothless/Documents/GitHub/BeeGame/↔
 - Code/BeeGame/BeeGame/Core/Dictionarys/↔
BeeDictionarys.cs, [251](#)

C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Core/Dictionary/↵
 PrefabDictionary.cs, 250
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Core/Dictionary/↵
 SpriteDictionary.cs, 250, 251
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Core/Enums/↵
 Enums.cs, 264, 266
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Core/Extensions.↵
 cs, 263, 264
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Core/UnityType↵
 Replacements/THInput.cs, 254, 255
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Core/UnityType↵
 Replacements/THVector2.cs, 256, 257
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Core/UnityType↵
 Replacements/THVector3.cs, 258, 259
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Inventory/Chest↵
 Inventory.cs, 222
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Inventory/↵
 Inventory.cs, 216
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Inventory/↵
 InventorySlot.cs, 217
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Inventory/Items↵
 InInventory.cs, 223, 224
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Inventory/Player
 Inventory/PlayerInventory.cs, 220
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Items/Apply↵
 Colour.cs, 206
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Items/Honey↵
 Comb.cs, 204
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Items/Item.cs,
 200, 201
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Items/Item↵
 GameObject.cs, 205
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/LoadResources.↵
 cs, 254
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Player/Player↵
 Look.cs, 245
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Player/Player↵
 Move.cs, 248
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Player/Save↵
 PlayerPosition.cs, 249
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Player/Selector.↵
 cs, 246
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Resources/↵
 Resources.Designer.cs, 252
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Serialization/↵
 Serialization.cs, 261
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/SpawnItem.cs,
 267
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Terrain/Chunk↵
 WorldPos.cs, 228
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Terrain/Chunks/↵
 Chunk.cs, 224, 225
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Terrain/Chunks/↵
 LoadChunks.cs, 229
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Terrain/Chunks/↵
 MeshData.cs, 227
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Terrain/Chunks/↵
 SaveChunk.cs, 233
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Terrain/Land↵
 Generation/Noise/SimplexNoise.cs, 241
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Terrain/Land↵
 Generation/Terrain.cs, 236
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Terrain/Land↵
 Generation/TerrainGeneration.cs, 239
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Terrain/Land↵
 Generation/World.cs, 233, 234
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/test.cs, 267, 268
 cameraTransform
 BeeGame::Player::PlayerLook, 143
 canJump
 BeeGame::Player::PlayerMove, 151
 caveFrequency
 BeeGame::Terrain::LandGeneration::Terrain↵
 Generation, 131
 caveSize
 BeeGame::Terrain::LandGeneration::Terrain↵
 Generation, 132
 changed
 BeeGame::Blocks::Block, 40
 CheckFloatingItem
 BeeGame::Inventory::InventorySlot, 75
 Chest

- BeeGame::Blocks::Chest, 52
- ChunkGen
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, 127
- ChunkGenThread
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, 128
- chunkHasMadeCollisionMesh
 - BeeGame::Terrain::LandGeneration::World, 117
- chunkPositions
 - BeeGame::Terrain::Chunks::LoadChunks, 109
- chunkPrefab
 - BeeGame::Terrain::LandGeneration::World, 117
- chunkSize
 - BeeGame::Terrain::Chunks::Chunk, 95
- ChunkWorldPos
 - BeeGame::Terrain::ChunkWorldPos, 102
- chunkWorldPos
 - BeeGame::Terrain::Chunks::Chunk, 95
- chunks
 - BeeGame::Terrain::LandGeneration::World, 117
- Clone
 - BeeGame::Items::Item, 14
- CloneObject< T >
 - BeeGame::Core::Extensions, 190
- colTris
 - BeeGame::Terrain::Chunks::MeshData, 100
- colVerts
 - BeeGame::Terrain::Chunks::MeshData, 100
- ColliderMesh
 - BeeGame::Terrain::Chunks::Chunk, 91
- colour
 - BeeGame::Items::ApplyColour, 33
- CombColour
 - BeeGame::Core::BeeDictionarys, 159
 - BeeGame::Items::HoneyComb, 29
- counter
 - BeeGame::Player::SavePlayerPosition, 154
- CreateChunk
 - BeeGame::Terrain::LandGeneration::World, 112
- CreateTree
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, 128
- Culture
 - BeeGame::Resources::Resources, 163
- DeSerializeInventory
 - BeeGame::Serialization::Serialization, 193
- DeleteChunks
 - BeeGame::Terrain::Chunks::LoadChunks, 107
- DeleteFile
 - BeeGame::Serialization::Serialization, 192
- DestroyChunk
 - BeeGame::Terrain::LandGeneration::World, 113
- Direction
 - BeeGame::Core::Enums, 7
- Dirt
 - BeeGame::Blocks::Dirt, 45
- dirtBaseHeight
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, 132
- dirtNoise
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, 132
- dirtNoiseHeight
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, 132
- Distance
 - BeeGame::Core::THVector3, 179
- done
 - BeeGame::Terrain::Chunks::MeshData, 100
- Equals
 - BeeGame::Core::THVector2, 171
 - BeeGame::Core::THVector3, 180
 - BeeGame::Items::Item, 14
 - BeeGame::Terrain::ChunkWorldPos, 102
- FaceDataDown
 - BeeGame::Items::Item, 15
- FaceDataEast
 - BeeGame::Items::Item, 15
- FaceDataNorth
 - BeeGame::Items::Item, 16
- FaceDataSouth
 - BeeGame::Items::Item, 17
- FaceDataUp
 - BeeGame::Items::Item, 18
- FaceDataWest
 - BeeGame::Items::Item, 19
- FaceUVs
 - BeeGame::Items::Item, 19
- FastFloor
 - BeeGame::Terrain::LandGeneration::Noise::↔
SimplexNoise, 135
- FileName
 - BeeGame::Serialization::Serialization, 193
- filter
 - BeeGame::Terrain::Chunks::Chunk, 96
- FindChunksToLoad
 - BeeGame::Terrain::Chunks::LoadChunks, 107
- FixedUpdate
 - BeeGame::Player::PlayerMove, 150
 - BeeGame::Player::Selector, 145
- floatingItem
 - BeeGame::Inventory::Inventory, 70
- GenChunkColum
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, 129
- Generate
 - BeeGame::Terrain::LandGeneration::Noise::↔
SimplexNoise, 135–137
- GetAllItems
 - BeeGame::Inventory::Inventory, 67
- GetAxis
 - BeeGame::Core::THInput, 165
- GetBlock

- BeeGame::Terrain::Chunks::Chunk, 91
- BeeGame::Terrain::LandGeneration::Terrain, 119
- BeeGame::Terrain::LandGeneration::World, 114
- GetBlockPos
 - BeeGame::Terrain::LandGeneration::Terrain, 120, 121
- GetBlockPosFromRayCast
 - BeeGame::Terrain::LandGeneration::Terrain, 121
- GetButton
 - BeeGame::Core::THInput, 166
- GetButtonDown
 - BeeGame::Core::THInput, 166
- GetButtonUp
 - BeeGame::Core::THInput, 167
- GetChunk
 - BeeGame::Terrain::LandGeneration::Terrain, 122
 - BeeGame::Terrain::LandGeneration::World, 114
- GetCombColour
 - BeeGame::Core::BeeDictionarys, 159
- GetGameObject
 - BeeGame::Blocks::Chest, 53
 - BeeGame::Items::HoneyComb, 27
 - BeeGame::Items::Item, 20
- GetHashCode
 - BeeGame::Blocks::Air, 43
 - BeeGame::Blocks::Apiary, 56
 - BeeGame::Blocks::Bedrock, 64
 - BeeGame::Blocks::Block, 37
 - BeeGame::Blocks::Chest, 54
 - BeeGame::Blocks::Dirt, 45
 - BeeGame::Blocks::Grass, 48
 - BeeGame::Blocks::Leaves, 60
 - BeeGame::Blocks::Wood, 58
 - BeeGame::Core::THVector2, 171
 - BeeGame::Core::THVector3, 180
 - BeeGame::Items::HoneyComb, 28
 - BeeGame::Items::Item, 20
 - BeeGame::Terrain::ChunkWorldPos, 102
- GetItemFromHotBar
 - BeeGame::Inventory::Player_Inventory::Player↔Inventory, 81
- GetItemID
 - BeeGame::Items::HoneyComb, 28
 - BeeGame::Items::Item, 21
- GetItemName
 - BeeGame::Blocks::Grass, 49
 - BeeGame::Items::Item, 21
- GetItemSprite
 - BeeGame::Blocks::Block, 37
 - BeeGame::Blocks::Dirt, 45
 - BeeGame::Blocks::Grass, 49
 - BeeGame::Blocks::Leaves, 60
 - BeeGame::Blocks::Wood, 58
 - BeeGame::Items::HoneyComb, 28
 - BeeGame::Items::Item, 21
- GetNoise
 - BeeGame::Terrain::LandGeneration::Terrain↔Generation, 130
- GetPrefab
 - BeeGame::Core::PrefabDictionary, 155
- GetPrefabs
 - BeeGame::Resources::Resources, 161
- GetSprite
 - BeeGame::Core::SpriteDictionary, 157
- GetSprites
 - BeeGame::Resources::Resources, 162
- go
 - BeeGame::Items::ItemGameObject, 31
- grad
 - BeeGame::Terrain::LandGeneration::Noise::↔SimplexNoise, 138, 139
- Grass
 - BeeGame::Blocks::Grass, 48
- gravity
 - BeeGame::Player::PlayerMove, 151
- hit
 - BeeGame::Player::Selector, 147
- HoneyComb
 - BeeGame::Items::HoneyComb, 27
- HoneyCombType
 - BeeGame::Core::Enums, 7
- honeyCombColour
 - BeeGame::Core::BeeDictionarys, 160
- InRange
 - BeeGame::Terrain::Chunks::Chunk, 92
- inputButtons
 - BeeGame::Core::THInput, 168
- InteractWithBlock
 - BeeGame::Blocks::Block, 37
 - BeeGame::Blocks::Chest, 54
- inventory
 - BeeGame::Inventory::ChestInventory, 88
- inventoryName
 - BeeGame::Inventory::Inventory, 70
- inventoryPosition
 - BeeGame::Inventory::ChestInventory, 88
- InventorySet
 - BeeGame::Inventory::Inventory, 68
- inventorySize
 - BeeGame::Inventory::ChestInventory, 88
- isAnotherInventoryOpen
 - BeeGame::Core::THInput, 168
- IsSolid
 - BeeGame::Blocks::Air, 43
 - BeeGame::Blocks::Block, 38
 - BeeGame::Blocks::Leaves, 61
- Item
 - BeeGame::Items::Item, 13
- item
 - BeeGame::Inventory::InventorySlot, 79
 - BeeGame::Items::ItemGameObject, 31
- ItemMesh
 - BeeGame::Items::Item, 22
- itemName
 - BeeGame::Items::Item, 24

- itemSpriteDictionary
 - BeeGame::Core::SpriteDictionary, 158
- itemStackCount
 - BeeGame::Items::Item, 24
- itemText
 - BeeGame::Inventory::InventorySlot, 79
- ItemType
 - BeeGame::Core::Enums, 8
- items
 - BeeGame::Inventory::Inventory, 70
- ItemsInInventory
 - BeeGame::Inventory::ItemsInInventory, 71
- itemsInInventory
 - BeeGame::Inventory::ItemsInInventory, 73
- jumpHeight
 - BeeGame::Player::PlayerMove, 152
- layers
 - BeeGame::Player::Selector, 147
- Leaves
 - BeeGame::Blocks::Leaves, 60
- LoadAndRenderChunks
 - BeeGame::Terrain::Chunks::LoadChunks, 108
- LoadChunk
 - BeeGame::Serialization::Serialization, 194
- LoadFile
 - BeeGame::Serialization::Serialization, 194
- LoadPlayerPosition
 - BeeGame::Serialization::Serialization, 195
- LoadPrefabs
 - BeeGame::Core::PrefabDictionary, 156
- LoadSprites
 - BeeGame::Core::SpriteDictionary, 157
- Look
 - BeeGame::Player::PlayerLook, 142
- MakeDirectorys
 - BeeGame::Serialization::Serialization, 195
- MakeMesh
 - BeeGame::Items::ItemGameObject, 30
- maxStackCount
 - BeeGame::Items::Item, 25
- maxVelocity
 - BeeGame::Player::PlayerMove, 152
- mesh
 - BeeGame::Terrain::Chunks::Chunk, 96
- meshCollider
 - BeeGame::Terrain::Chunks::Chunk, 96
- Mod
 - BeeGame::Terrain::LandGeneration::Noise::↔
SimplexNoise, 140
- MovePlayer
 - BeeGame::Player::PlayerMove, 150
- myGameObject
 - BeeGame::Blocks::Chest, 55
- myInventory
 - BeeGame::Inventory::InventorySlot, 79
- myRigidBody
 - BeeGame::Player::PlayerMove, 152
- myTransform
 - BeeGame::Player::PlayerLook, 143
- nearbyChunks
 - BeeGame::Terrain::Chunks::LoadChunks, 109
- objects
 - BeeGame::Items::ApplyColour, 33
- OnCollisionStay
 - BeeGame::Player::PlayerMove, 150
- OnDisable
 - BeeGame::Inventory::InventorySlot, 75
- OnDrawGizmos
 - BeeGame::SpawnItem, 199
- OnPointerClick
 - BeeGame::Inventory::InventorySlot, 75
- OnPointerEnter
 - BeeGame::Inventory::InventorySlot, 77
- OnPointerExit
 - BeeGame::Inventory::InventorySlot, 77
- OpenPlayerInventory
 - BeeGame::Inventory::Player_Inventory::Player↔
Inventory, 82
- operator ChunkWorldPos
 - BeeGame::Terrain::ChunkWorldPos, 103
- operator Quaternion
 - BeeGame::Core::THVector3, 180
- operator THVector2
 - BeeGame::Core::THVector2, 171
- operator THVector3
 - BeeGame::Core::THVector3, 181
 - BeeGame::Terrain::ChunkWorldPos, 103
- operator Vector2
 - BeeGame::Core::THVector2, 172
- operator Vector3
 - BeeGame::Core::THVector3, 181
- operator!=
 - BeeGame::Core::THVector2, 172
 - BeeGame::Core::THVector3, 182
 - BeeGame::Items::Item, 22
- operator*
 - BeeGame::Core::THVector2, 172, 173
 - BeeGame::Core::THVector3, 182, 183
- operator+
 - BeeGame::Core::THVector2, 173
 - BeeGame::Core::THVector3, 184, 185
- operator-
 - BeeGame::Core::THVector2, 174
 - BeeGame::Core::THVector3, 185, 186
- operator/
 - BeeGame::Core::THVector2, 174, 175
 - BeeGame::Core::THVector3, 187, 188
- operator==
 - BeeGame::Core::THVector2, 175
 - BeeGame::Core::THVector3, 188
 - BeeGame::Items::Item, 23
- perm

- BeeGame::Terrain::LandGeneration::Noise::↔
SimplexNoise, 140
- PickupItem
 - BeeGame::Inventory::Player_Inventory::Player↔
Inventory, 82
- PlaceBlock
 - BeeGame::Player::Selector, 146
- placeable
 - BeeGame::Items::Item, 25
- playerInventory
 - BeeGame::Inventory::Player_Inventory::Player↔
Inventory, 85
 - BeeGame::Player::Selector, 148
- playerinventory
 - BeeGame::Inventory::ChestInventory, 88
- prefabDictionary
 - BeeGame::Core::PrefabDictionary, 156
- Prefabs
 - BeeGame::Resources::Resources, 163
- PutItemsInSlots
 - BeeGame::Inventory::Inventory, 68
- RemoveItemFromInventory
 - BeeGame::Inventory::Player_Inventory::Player↔
Inventory, 83
- RenderMesh
 - BeeGame::Terrain::Chunks::Chunk, 92
- rendered
 - BeeGame::Terrain::Chunks::Chunk, 96
- resourceCulture
 - BeeGame::Resources::Resources, 163
- resourceMan
 - BeeGame::Resources::Resources, 163
- ResourceManager
 - BeeGame::Resources::Resources, 163
- Resources
 - BeeGame::Resources::Resources, 161
- rotationLock
 - BeeGame::Player::PlayerLook, 143
- Round
 - BeeGame::Terrain::LandGeneration::Terrain, 122
- RoundXZ
 - BeeGame::Terrain::LandGeneration::Terrain, 123
- RoundY
 - BeeGame::Terrain::LandGeneration::Terrain, 124
- SaveChunk
 - BeeGame::Serialization::Serialization, 196
 - BeeGame::Terrain::Chunks::SaveChunk, 111
- SaveFile
 - BeeGame::Serialization::Serialization, 196
- saveFolderName
 - BeeGame::Serialization::Serialization, 198
- SaveInv
 - BeeGame::Inventory::Inventory, 68
- savePath
 - BeeGame::Serialization::Serialization, 198
- SavePlayerPosition
 - BeeGame::Serialization::Serialization, 197
- selectedHotbarSlot
 - BeeGame::Player::Selector, 148
- SelectedSlot
 - BeeGame::Inventory::Player_Inventory::Player↔
Inventory, 83
 - BeeGame::Player::Selector, 146
- selectedSlot
 - BeeGame::Inventory::InventorySlot, 80
- selector
 - BeeGame::Player::Selector, 148
- SerializeInventory
 - BeeGame::Serialization::Serialization, 197
- SetAllItems
 - BeeGame::Inventory::Inventory, 69
- SetBlock
 - BeeGame::Terrain::Chunks::Chunk, 93
 - BeeGame::Terrain::LandGeneration::Terrain, 125
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, 131
 - BeeGame::Terrain::LandGeneration::World, 115
- SetBlocksUnmodified
 - BeeGame::Terrain::Chunks::Chunk, 93
- SetChestInventory
 - BeeGame::Inventory::ChestInventory, 86
- SetInventorySize
 - BeeGame::Inventory::Inventory, 69
- SetPlayerInventory
 - BeeGame::Inventory::Player_Inventory::Player↔
Inventory, 84
- SetPlayerItems
 - BeeGame::Inventory::ChestInventory, 86
- shareMeshes
 - BeeGame::Terrain::Chunks::MeshData, 100
- slotIndex
 - BeeGame::Inventory::InventorySlot, 80
- slots
 - BeeGame::Inventory::Inventory, 70
- speed
 - BeeGame::Player::PlayerLook, 143
 - BeeGame::Player::PlayerMove, 152
- SplitStack
 - BeeGame::Inventory::InventorySlot, 77
- Sprites
 - BeeGame::Resources::Resources, 164
- Start
 - BeeGame::Items::ApplyColour, 32
 - BeeGame::Items::ItemGameObject, 30
 - BeeGame::Player::PlayerLook, 142
 - BeeGame::SpawnItem, 199
 - BeeGame::Terrain::Chunks::Chunk, 94
 - BeeGame::Terrain::Chunks::LoadChunks, 108
 - BeeGame::Test, 200
- stoneBaseHeight
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, 132
- stoneBaseNoise
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, 133

- stoneBaseNoiseHeight
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, [133](#)
- stoneMinHeight
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, [133](#)
- stoneMountainFrequency
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, [133](#)
- stoneMountainHeight
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, [133](#)
- SwapItems
 - BeeGame::Inventory::InventorySlot, [78](#)
- THVector2
 - BeeGame::Core::THVector2, [170](#)
- THVector3
 - BeeGame::Core::THVector3, [178](#), [179](#)
- TexturePosition
 - BeeGame::Blocks::Bedrock, [64](#)
 - BeeGame::Blocks::Chest, [54](#)
 - BeeGame::Blocks::Dirt, [46](#)
 - BeeGame::Blocks::Grass, [49](#)
 - BeeGame::Blocks::Leaves, [61](#)
 - BeeGame::Blocks::Wood, [58](#)
 - BeeGame::Items::Item, [23](#)
- thisInventoryOpen
 - BeeGame::Inventory::Inventory, [70](#)
- tileSize
 - BeeGame::Items::Item, [25](#)
- timer
 - BeeGame::Terrain::Chunks::LoadChunks, [110](#)
- ToString
 - BeeGame::Blocks::Air, [43](#)
 - BeeGame::Blocks::Apiary, [56](#)
 - BeeGame::Blocks::Bedrock, [65](#)
 - BeeGame::Blocks::Block, [39](#)
 - BeeGame::Blocks::Chest, [55](#)
 - BeeGame::Blocks::Dirt, [47](#)
 - BeeGame::Blocks::Grass, [50](#)
 - BeeGame::Blocks::Leaves, [62](#)
 - BeeGame::Blocks::Wood, [59](#)
 - BeeGame::Core::THVector2, [175](#)
 - BeeGame::Core::THVector3, [189](#)
 - BeeGame::Items::Item, [24](#)
 - BeeGame::Terrain::ChunkWorldPos, [104](#)
- ToggleInventory
 - BeeGame::Inventory::ChestInventory, [87](#)
- treeDensity
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, [134](#)
- treeFrequency
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, [134](#)
- tris
 - BeeGame::Terrain::Chunks::MeshData, [100](#)
- type
 - BeeGame::Items::HoneyComb, [29](#)
- Update
 - BeeGame::Inventory::ChestInventory, [87](#)
 - BeeGame::Inventory::InventorySlot, [78](#)
 - BeeGame::Inventory::Player_Inventory::Player↔
Inventory, [84](#)
 - BeeGame::Items::ItemGameObject, [31](#)
 - BeeGame::Player::PlayerLook, [142](#)
 - BeeGame::Player::SavePlayerPosition, [153](#)
 - BeeGame::Player::Selector, [146](#)
 - BeeGame::Terrain::Chunks::Chunk, [94](#)
 - BeeGame::Terrain::Chunks::LoadChunks, [109](#)
- update
 - BeeGame::Terrain::Chunks::Chunk, [96](#)
- UpdateBase
 - BeeGame::Inventory::Inventory, [70](#)
- UpdateBlock
 - BeeGame::Blocks::Block, [39](#)
 - BeeGame::Blocks::Grass, [50](#)
- UpdateChunk
 - BeeGame::Terrain::Chunks::Chunk, [94](#)
- updateCollisionMesh
 - BeeGame::Terrain::Chunks::Chunk, [97](#)
- UpdateIcon
 - BeeGame::Inventory::InventorySlot, [78](#)
- UpdateIfEqual
 - BeeGame::Terrain::LandGeneration::World, [116](#)
- UpdateSelector
 - BeeGame::Player::Selector, [147](#)
- usesGameObject
 - BeeGame::Items::Item, [25](#)
- uv
 - BeeGame::Terrain::Chunks::MeshData, [100](#)
- VerticalJumpSpeed
 - BeeGame::Player::PlayerMove, [151](#)
- verts
 - BeeGame::Terrain::Chunks::MeshData, [101](#)
- Wood
 - BeeGame::Blocks::Wood, [58](#)
- world
 - BeeGame::Terrain::Chunks::Chunk, [97](#)
 - BeeGame::Terrain::Chunks::LoadChunks, [110](#)
 - BeeGame::Terrain::LandGeneration::Terrain, [126](#)
- worldName
 - BeeGame::Serialization::Serialization, [198](#)
- x
 - BeeGame::Core::THVector2, [176](#)
 - BeeGame::Core::THVector3, [189](#)
 - BeeGame::Items::Tile, [41](#)
 - BeeGame::Terrain::ChunkWorldPos, [104](#)
- xRot
 - BeeGame::Player::PlayerLook, [143](#)
- y
 - BeeGame::Core::THVector2, [176](#)
 - BeeGame::Core::THVector3, [189](#)
 - BeeGame::Items::Tile, [41](#)

BeeGame::Terrain::ChunkWorldPos, [104](#)
yRot
BeeGame::Player::PlayerLook, [143](#)

z
BeeGame::Core::THVector3, [189](#)
BeeGame::Terrain::ChunkWorldPos, [104](#)