

Bee Game Namespace, Class, & File Documentation

Bee Game Version 0.1 - Minecraft Pre 0.0.9a Equivlent

Max Rose

Generated by Doxygen 1.8.13

Contents

I	Namespace Documentation	2
0.1	BeeGame Namespace Reference	2
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	4
0.6	BeeGame.Inventory.Player_Inventory Namespace Reference	4
0.7	BeeGame.Items Namespace Reference	4
0.8	BeeGame.Player Namespace Reference	4
0.9	BeeGame.Resources Namespace Reference	4
0.10	BeeGame.Serialization Namespace Reference	5
0.11	BeeGame.Terrain Namespace Reference	5
0.12	BeeGame.Terrain.Chunks Namespace Reference	5
0.13	BeeGame.Terrain.LandGeneration Namespace Reference	5
0.14	BeeGame.Terrain.LandGeneration.Noise Namespace Reference	6
II	Class Documentation	6
1	Items	6
1.1	BeeGame.Items.Item Class Reference	6
1.1.1	Detailed Description	8
1.1.2	Constructor & Destructor Documentation	8
1.1.3	Member Function Documentation	8
1.1.4	Member Data Documentation	18
1.2	BeeGame.Items.ItemGameObject Class Reference	20
1.2.1	Detailed Description	20
1.2.2	Member Function Documentation	20
1.2.3	Member Data Documentation	21

2	Blocks	21
2.1	BeeGame.Items.Tile Struct Reference	21
2.1.1	Detailed Description	22
2.1.2	Member Data Documentation	22
2.2	BeeGame.Blocks.Block Class Reference	22
2.2.1	Detailed Description	23
2.2.2	Constructor & Destructor Documentation	23
2.2.3	Member Function Documentation	24
2.2.4	Member Data Documentation	28
2.3	BeeGame.Blocks.Air Class Reference	28
2.3.1	Detailed Description	29
2.3.2	Constructor & Destructor Documentation	29
2.3.3	Member Function Documentation	29
2.4	BeeGame.Blocks.Bedrock Class Reference	32
2.4.1	Detailed Description	32
2.4.2	Constructor & Destructor Documentation	32
2.4.3	Member Function Documentation	33
2.5	BeeGame.Blocks.Apiary Class Reference	34
2.5.1	Detailed Description	35
2.5.2	Constructor & Destructor Documentation	35
2.5.3	Member Function Documentation	36
2.6	BeeGame.Blocks.Dirt Class Reference	36
2.6.1	Detailed Description	37
2.6.2	Constructor & Destructor Documentation	37
2.6.3	Member Function Documentation	37
2.7	BeeGame.Blocks.Grass Class Reference	38
2.7.1	Detailed Description	39
2.7.2	Constructor & Destructor Documentation	39
2.7.3	Member Function Documentation	39

3	Inventory	42
3.1	BeeGame.Inventory.Inventory Class Reference	42
3.1.1	Detailed Description	43
3.1.2	Member Function Documentation	43
3.1.3	Member Data Documentation	46
3.2	BeeGame.Inventory.ItemsInInventory Class Reference	47
3.2.1	Detailed Description	48
3.2.2	Constructor & Destructor Documentation	48
3.2.3	Member Function Documentation	48
3.2.4	Member Data Documentation	49
3.3	BeeGame.Inventory.InventorySlot Class Reference	50
3.3.1	Detailed Description	51
3.3.2	Member Function Documentation	51
3.3.3	Member Data Documentation	55
3.4	BeeGame.Inventory.Player_Inventory.PlayerInventory Class Reference	56
3.4.1	Detailed Description	57
3.4.2	Member Function Documentation	57
3.4.3	Member Data Documentation	61
4	Chunk	61
4.1	BeeGame.Terrain.Chunks.Chunk Class Reference	61
4.1.1	Detailed Description	63
4.1.2	Member Function Documentation	63
4.1.3	Member Data Documentation	67
4.2	BeeGame.Terrain.Chunks.MeshData Class Reference	70
4.2.1	Detailed Description	70
4.2.2	Member Function Documentation	70
4.2.3	Member Data Documentation	72
4.3	BeeGame.Terrain.Chunks.LoadChunks Class Reference	73
4.3.1	Detailed Description	74
4.3.2	Member Function Documentation	74

4.3.3	Member Data Documentation	78
4.4	BeeGame.Terrain.Chunks.SaveChunk Class Reference	79
4.4.1	Detailed Description	79
4.4.2	Constructor & Destructor Documentation	79
4.4.3	Member Data Documentation	80
4.5	BeeGame.Terrain.ChunkWorldPos Struct Reference	80
4.5.1	Detailed Description	81
4.5.2	Constructor & Destructor Documentation	81
4.5.3	Member Function Documentation	81
4.5.4	Member Data Documentation	83
4.6	BeeGame.Terrain.LandGeneration.World Class Reference	84
4.6.1	Detailed Description	85
4.6.2	Member Function Documentation	85
4.6.3	Member Data Documentation	89
4.7	BeeGame.Terrain.LandGeneration.Terrain Class Reference	90
4.7.1	Detailed Description	91
4.7.2	Member Function Documentation	91
4.7.3	Member Data Documentation	98
4.8	BeeGame.Terrain.LandGeneration.TerrainGeneration Class Reference	98
4.8.1	Detailed Description	99
4.8.2	Member Function Documentation	99
4.8.3	Member Data Documentation	102
4.9	BeeGame.Terrain.LandGeneration.Noise.SimplexNoise Class Reference	105
4.9.1	Detailed Description	105
4.9.2	Member Function Documentation	105
4.9.3	Member Data Documentation	110

5	Player	111
5.1	BeeGame.Player.PlayerLook Class Reference	111
5.1.1	Detailed Description	112
5.1.2	Member Function Documentation	112
5.1.3	Member Data Documentation	113
5.2	BeeGame.Player.PlayerMove Class Reference	114
5.2.1	Detailed Description	115
5.2.2	Member Function Documentation	115
5.2.3	Member Data Documentation	117
5.3	BeeGame.Player.Selector Class Reference	119
5.3.1	Detailed Description	119
5.3.2	Member Function Documentation	120
5.3.3	Member Data Documentation	122
6	Resources	123
6.1	BeeGame.LoadResources Class Reference	123
6.1.1	Detailed Description	123
6.1.2	Member Function Documentation	123
6.2	BeeGame.Core.PrefabDictionary Class Reference	124
6.2.1	Detailed Description	124
6.2.2	Member Function Documentation	124
6.2.3	Member Data Documentation	125
6.3	BeeGame.Core.SpriteDictionary Class Reference	125
6.3.1	Detailed Description	126
6.3.2	Member Function Documentation	126
6.3.3	Member Data Documentation	127
6.4	BeeGame.Resources.Resources Class Reference	127
6.4.1	Detailed Description	128
6.4.2	Constructor & Destructor Documentation	128
6.4.3	Member Function Documentation	128
6.4.4	Member Data Documentation	129
6.4.5	Property Documentation	129

7	Unity Type & Method Replacements	130
7.1	BeeGame.Core.THInput Class Reference	130
7.1.1	Detailed Description	131
7.1.2	Member Function Documentation	131
7.1.3	Member Data Documentation	134
7.2	BeeGame.Core.THVector2 Struct Reference	135
7.2.1	Detailed Description	136
7.2.2	Constructor & Destructor Documentation	136
7.2.3	Member Function Documentation	137
7.2.4	Member Data Documentation	142
7.3	BeeGame.Core.THVector3 Struct Reference	142
7.3.1	Detailed Description	143
7.3.2	Constructor & Destructor Documentation	144
7.3.3	Member Function Documentation	145
7.3.4	Member Data Documentation	155
8	Misc	155
8.1	BeeGame.Serialization.Serialization Class Reference	155
8.1.1	Detailed Description	156
8.1.2	Member Function Documentation	156
8.1.3	Member Data Documentation	160
8.2	BeeGame.Core.Extensions Class Reference	161
8.2.1	Detailed Description	161
8.2.2	Member Function Documentation	161
8.3	BeeGame.Test Class Reference	162
8.3.1	Detailed Description	163
8.3.2	Member Function Documentation	163
III	File Documentation	163

1	Items	163
1.1	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/Item.cs File Reference	163
1.2	Item.cs	164
1.3	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/ItemGame↔Object.cs File Reference	167
1.4	ItemGameObject.cs	167
2	Blocks	168
2.1	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Block.cs File Reference	168
2.2	Block.cs	168
2.3	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Air.cs File Reference	169
2.4	Air.cs	170
2.5	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Bedrock.cs File Reference	170
2.6	Bedrock.cs	171
2.7	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Apiary.cs File Reference	171
2.8	Apiary.cs	171
2.9	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Dirt.cs File Reference	172
2.10	Dirt.cs	172
2.11	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Grass.cs File Reference	173
2.12	Grass.cs	173
3	Inventorys	174
3.1	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Inventory.cs File Reference	174
3.2	Inventory.cs	174
3.3	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/ItemsIn↔Inventory.cs File Reference	175
3.4	ItemsInInventory.cs	175
3.5	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Inventory↔Slot.cs File Reference	176
3.6	InventorySlot.cs	176
3.7	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Player Inventory/PlayerInventory.cs File Reference	178
3.8	PlayerInventory.cs	179

4	Chunks	180
4.1	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/↔ Chunk.cs File Reference	180
4.2	Chunk.cs	180
4.3	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/↔ MeshData.cs File Reference	183
4.4	MeshData.cs	183
4.5	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/↔ LoadChunks.cs File Reference	184
4.6	LoadChunks.cs	184
4.7	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/↔ SaveChunk.cs File Reference	187
4.8	SaveChunk.cs	188
4.9	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunk↔ WorldPos.cs File Reference	188
4.10	ChunkWorldPos.cs	188
4.11	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Land↔ Generation/World.cs File Reference	189
4.12	World.cs	189
4.13	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Land↔ Generation/Terrain.cs File Reference	191
4.14	Terrain.cs	192
4.15	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Land↔ Generation/TerrainGeneration.cs File Reference	195
4.16	TerrainGeneration.cs	195
4.17	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Land↔ Generation/Noise/SimplexNoise.cs File Reference	196
4.18	SimplexNoise.cs	197
5	Player	200
5.1	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/PlayerLook.cs File Reference	200
5.2	PlayerLook.cs	200
5.3	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/Player↔ Move.cs File Reference	201
5.4	PlayerMove.cs	201
5.5	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/Selector.cs File Reference	202
5.6	Selector.cs	203

6	Resources	204
6.1	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Prefab↔ Dictionary.cs File Reference	204
6.2	PrefabDictionary.cs	204
6.3	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/SpriteDictionary.cs File Reference	205
6.4	SpriteDictionary.cs	205
6.5	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Resources/Resources.Designer.↔ cs File Reference	205
6.6	Resources.Designer.cs	206
6.7	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/LoadResources.cs File Reference	207
6.8	LoadResources.cs	208
7	Unity Type & Method Replacements	208
7.1	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityType↔ Replacements/THInput.cs File Reference	208
7.2	THInput.cs	208
7.3	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityType↔ Replacements/THVector2.cs File Reference	210
7.4	THVector2.cs	210
7.5	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityType↔ Replacements/THVector3.cs File Reference	212
7.6	THVector3.cs	212
8	Misc	214
8.1	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Serialization/↔ Serialization.cs File Reference	214
8.2	Serialization.cs	215
8.3	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Extensions.cs File Reference	216
8.4	Extensions.cs	217
8.5	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/test.cs File Reference	217
8.6	test.cs	218
8.7	C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Enums/↔ Enums.cs File Reference	218
8.8	Enums.cs	218

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 [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 [Dirt](#)
Dirt Block
- class [Grass](#)
Grass Block

0.3 BeeGame.Core Namespace Reference

Namespaces

- namespace [Enums](#)

Classes

- 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 [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 Direction

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

Direction in the game

Enumerator

NORTH	
EAST	
SOUTH	
WEST	
UP	
DOWN	

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

```
00007    {  
00008        NORTH, EAST, SOUTH, WEST, UP, DOWN  
00009    };
```

0.5 BeeGame.Inventory Namespace Reference

Namespaces

- namespace [Player_Inventory](#)

Classes

- 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 [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 [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

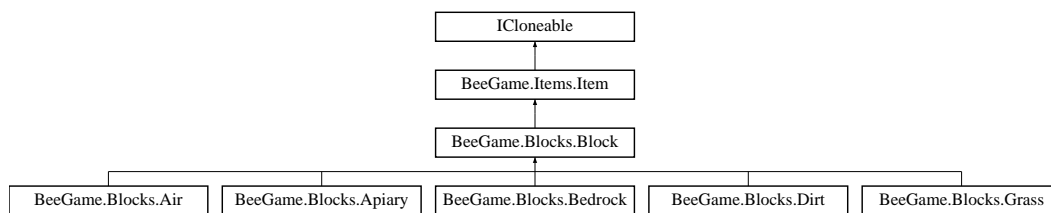
Class Documentation

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.DOWN));
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 -
        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     }

```

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);
00211         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z +
00212         blockSize), addToRenderMesh);
00213         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z +
00214         blockSize), addToRenderMesh);
00215         meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z +
00216         blockSize), addToRenderMesh);
00217         //adds the tirs for the quad
00218         meshData.AddQuadTriangles(addToRenderMesh);
00219         //if the data should be added to the render mesh also add the uvs to the mesh
00220         if (addToRenderMesh)
00221             meshData.uv.AddRange(FaceUVs(Direction.NORTH));
00222         return meshData;
00223     }
```

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 -
00266         blockSize), addToRenderMesh);
00267         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z -
00268         blockSize), addToRenderMesh);
00269         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z -
00270         blockSize), addToRenderMesh);
00271         meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z -
00272         blockSize), addToRenderMesh);
00273         //adds the tirs for the quad
00274         meshData.AddQuadTriangles(addToRenderMesh);
00275         //if the data should be added to the render mesh also add the uvs to the mesh
00276         if (addToRenderMesh)
00277             meshData.uv.AddRange(FaceUVs(Direction.SOUTH));
00278         return meshData;
00279     }
```

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 +
00154         blockSize), addToRenderMesh, Direction.UP);
00155         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z +
00156         blockSize), addToRenderMesh, Direction.UP);
00157         meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z -
00158         blockSize), addToRenderMesh, Direction.UP);
00159         meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z -
00160         blockSize), addToRenderMesh, Direction.UP);
00161         //adds teh tirs for the quad
00162         meshData.AddQuadTriangles(addToRenderMesh);
00163         //if the data should be added to the render mesh also add the uvs to the mesh
00164         if (addToRenderMesh)
00165             meshData.uv.AddRange(FaceUVs(Direction.UP));
00166         return meshData;
00167     }
```

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 +
00294         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 +
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     }

```

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

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

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

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* != *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 position of the items texture

Parameters

<i>direction</i>	Direction for the texture
------------------	---------------------------

Returns

Position of the texture

Reimplemented in [BeeGame.Blocks.Bedrock](#), [BeeGame.Blocks.Grass](#), and [BeeGame.Blocks.Dirt](#).

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 and 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](#).

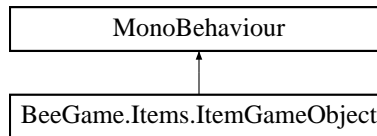
The documentation for this class was generated from the following file:

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

1.2 BeeGame.Items.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 MakeMesh \(\)](#)
Makes the items mesh

1.2.1 Detailed Description

Interface between item and inity gameobjects

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

1.2.2 Member Function Documentation

1.2.2.1 MakeMesh()

```
void BeeGame.Items.ItemGameObject.MakeMesh ( ) [private]
```

Makes the items mesh

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

```
00048     {
00049         MeshData meshData = new MeshData();
00050         if(item != null)
00051             meshData = item.ItemMesh(0, 0, 0, meshData);
00052
00053         Mesh mesh = new Mesh()
00054         {
00055             vertices = meshData.verts.ToArray(),
00056             triangles = meshData.tris.ToArray(),
00057             uv = meshData.uv.ToArray()
00058         };
00059
00060         mesh.RecalculateNormals();
00061
00062         GetComponent<MeshFilter>().mesh = mesh;
00063     }
```

1.2.2.2 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             GetComponent<BoxCollider>().enabled = false;
00040             Instantiate(item.GetGameObject(), transform, false);
00041         }
00042     }
```

1.2.3 Member Data Documentation

1.2.3.1 go

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

GameObject to make

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

1.2.3.2 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](#)

2 Blocks

2.1 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

2.1.1 Detailed Description

Position of the items texture

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

2.1.2 Member Data Documentation

2.1.2.1 x

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

X pos of the texture

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

2.1.2.2 y

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

Y pos of the texture

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

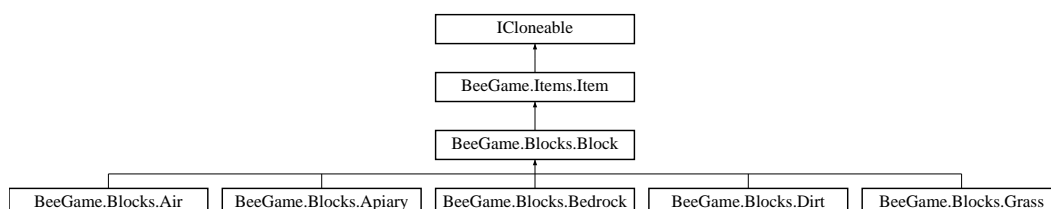
The documentation for this struct was generated from the following file:

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

2.2 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)
- 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 [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

2.2.1 Detailed Description

Base class for blocks

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

2.2.2 Constructor & Destructor Documentation

2.2.2.1 [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     }
```


2.2.2.2 Block() [2/2]

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

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

```
00036                                     : base(name)
00037     {
00038         placeable = true;
00039     }
```

2.2.3 Member Function Documentation

2.2.3.1 BlockData()

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

The data that this block adds to the mesh

Parameters

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

Returns

Given *meshData* with this blocks data added to it

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

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

Definition at line 78 of file [Block.cs](#).

```
00079     {
00080         //Adds the Top face of the block
00081         if (!chunk.GetBlock(x, y + 1, z, false).IsSolid(Direction.DOWN))
00082         {
00083             meshData = FaceDataUp(x, y, z, meshData, addToRenderMesh);
00084         }
```

```

00085
00086         //Adds the Bottom face of the block
00087         if (!chunk.GetBlock(x, y - 1, z, false).IsSolid(Direction.UP))
00088         {
00089             meshData = FaceDataDown(x, y, z, meshData, addToRenderMesh);
00090         }
00091
00092         //Adds the North face of the block
00093         if (!chunk.GetBlock(x, y, z + 1, false).IsSolid(Direction.SOUTH))
00094         {
00095             meshData = FaceDataNorth(x, y, z, meshData, addToRenderMesh);
00096         }
00097
00098         //Adds the South face of the block
00099         if (!chunk.GetBlock(x, y, z - 1, false).IsSolid(Direction.NORTH))
00100         {
00101             meshData = FaceDataSouth(x, y, z, meshData, addToRenderMesh);
00102         }
00103
00104         //Adds the East face of the block
00105         if (!chunk.GetBlock(x + 1, y, z, false).IsSolid(Direction.WEST))
00106         {
00107             meshData = FaceDataEast(x, y, z, meshData, addToRenderMesh);
00108         }
00109
00110         //Adds the West face of the block
00111         if (!chunk.GetBlock(x - 1, y, z, false).IsSolid(Direction.EAST))
00112         {
00113             meshData = FaceDataWest(x, y, z, meshData, addToRenderMesh);
00114         }
00115
00116         return meshData;
00117     }
00118 }

```

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

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

```

00048     {
00049         GameObject go = Object.Instantiate(UnityEngine.Resources.Load("
Prefabs/ItemGameObject") as GameObject, pos, Quaternion.identity) as GameObject;
00050         go.GetComponent<ItemGameObject>().item = this;
00051     }

```

2.2.3.3 GetHashCode()

```

override int BeeGame.Blocks.Block.GetHashCode ( )

```

Hascode for the [Block](#)

Returns

1

Definition at line 136 of file [Block.cs](#).

```
00137     {  
00138         return 1;  
00139     }
```

2.2.3.4 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](#).

Definition at line 125 of file [Block.cs](#).

```
00126     {  
00127         return true;  
00128     }
```

2.2.3.5 ToString()

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

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

Returns

Definition at line 145 of file [Block.cs](#).

```
00146     {  
00147         return $"{itemName} \nID: {GetHashCode()}";  
00148     }
```

2.2.3.6 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 of 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 60 of file [Block.cs](#).

```
00060 { }
```

2.2.4 Member Data Documentation

2.2.4.1 breakable

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

Can this [Block](#) be broken

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

2.2.4.2 changed

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

Has this block been placed by the player

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

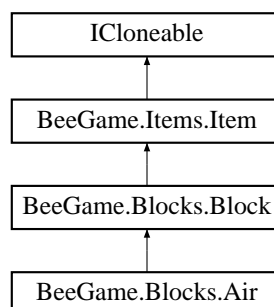
The documentation for this class was generated from the following file:

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

2.3 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      }
```

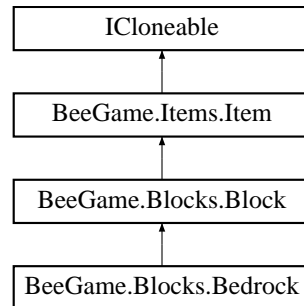
The documentation for this class was generated from the following file:

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

2.4 BeeGame.Blocks.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

2.4.1 Detailed Description

Bedrock Block

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

2.4.2 Constructor & Destructor Documentation

2.4.2.1 Bedrock()

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

Constructor

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

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

2.4.3 Member Function Documentation

2.4.3.1 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     }
```

2.4.3.2 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     }
```

2.4.3.3 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         }
```

2.4.3.4 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         }
```

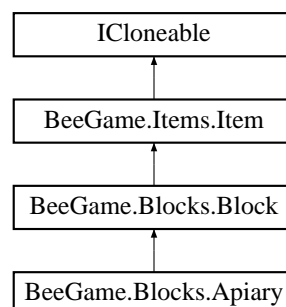
The documentation for this class was generated from the following file:

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

2.5 BeeGame.Blocks.Apiary Class Reference

[Apiary Block](#)

Inheritance diagram for BeeGame.Blocks.Apiary:



Public Member Functions

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

Additional Inherited Members

2.5.1 Detailed Description

[Apiary Block](#)

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

2.5.2 Constructor & Destructor Documentation

2.5.2.1 [Apiary\(\)](#) [1/2]

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

Constructor

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

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

2.5.2.2 [Apiary\(\)](#) [2/2]

```
BeeGame.Blocks.Apiary.Apiary (
    SerializationInfo info,
    StreamingContext context )
```

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

```
00020     {
00021         /*use info.getvalue("valuename", typeof(valueType))
00022         UnityEngine.MonoBehaviour.print("hi");
00023     }
```

2.5.3 Member Function Documentation

2.5.3.1 GetHashCode()

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

ID of the item

Returns

3

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

```
00031      {  
00032          return 3;  
00033      }
```

2.5.3.2 ToString()

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

The item name and ID as a string

Returns

A nicely formatted string

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

```
00040      {  
00041          return $"{itemName} \nID: {GetItemID()}";  
00042      }
```

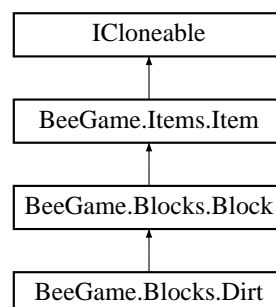
The documentation for this class was generated from the following file:

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

2.6 BeeGame.Blocks.Dirt Class Reference

[Dirt Block](#)

Inheritance diagram for BeeGame.Blocks.Dirt:



Public Member Functions

- [Dirt \(\)](#)
Constructor
- 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

2.6.1 Detailed Description

[Dirt Block](#)

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

2.6.2 Constructor & Destructor Documentation

2.6.2.1 Dirt()

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

Constructor

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

```
00017 : base("Dirt"){}
```

2.6.3 Member Function Documentation

2.6.3.1 GetHashCode()

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

Base ID of the block

Returns

5

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

```
00038     {  
00039         return 5;  
00040     }
```

2.6.3.2 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 26 of file [Dirt.cs](#).

```
00027         {  
00028             return new Tile { x = 2, y = 9 };  
00029         }
```

2.6.3.3 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 46 of file [Dirt.cs](#).

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

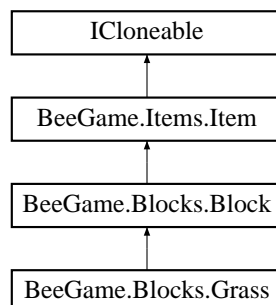
The documentation for this class was generated from the following file:

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

2.7 BeeGame.Blocks.Grass Class Reference

[Grass Block](#)

Inheritance diagram for BeeGame.Blocks.Grass:



Public Member Functions

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

2.7.1 Detailed Description

[Grass Block](#)

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

2.7.2 Constructor & Destructor Documentation

2.7.2.1 [Grass\(\)](#)

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

Constructor also sets teh items name

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

```
00018 : base("Grass") { }
```

2.7.3 Member Function Documentation

2.7.3.1 [GetHashCode\(\)](#)

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

The Base id for the block

Returns

4

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

```
00077         {  
00078             return 4;  
00079         }
```


2.7.3.2 GetItemName()

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

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

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

```
00068     {
00069         return "Grass";
00070     }
```

2.7.3.3 TexturePosition()

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

Texture position of the [Block](#) face

Parameters

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

Returns

Texture position as a [Tile](#)

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

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

```
00041     {
00042         //All textures are on the same Y value for the texture atlas so Y can be set
00043         Tile tile = new Tile()
00044     {
00045         y = 9
00046     };
00047
00048     switch (direction)
00049     {
00050         //if we want the top face return the full grass texture
00051         case Direction.UP:
00052             tile.x = 3;
00053             return tile;
00054         //if we want the bottom face return the dirt texture
00055         case Direction.DOWN:
00056             tile.x = 2;
00057             return tile;
00058         //return the 1/2 grass texture if a side face is wanted
00059         default:
00060             tile.x = 4;
00061             return tile;
00062     }
00063 }
```

2.7.3.4 ToString()

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

REturns the name and value for the block as a string

Returns

A nicely formatted string

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

```
00086         {
00087             return $"{itemName} \nID: {GetItemID()}";
00088         }
```

2.7.3.5 UpdateBlock()

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

Will turn this [Block](#) into a [Dirt](#) block if another block is above it

Parameters

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

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

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

```
00030         {
00031             if (chunk.GetBlock(x, y + 1, z, false).IsSolid(Direction.DOWN))
00032                 chunk.blocks[x, y, z] = new Dirt() { changed =
00033                     changed };
00033         }
```

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

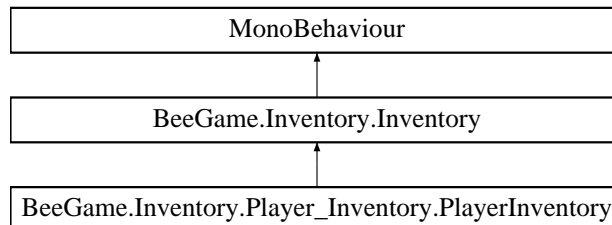
- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/[Grass.cs](#)

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
- [ItemsInInventory](#) [GetAllItems](#) ()
Gets all of the items in the inventory
- 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

- [InventorySlot](#) [] [slots](#)
Slots in the inventory
- string [inventoryName](#) = ""
Name of this inventory

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](#)

Private Attributes

- [ItemsInInventory items](#)
Items in the inventory

3.1.1 Detailed Description

Base class for all inventories 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 was added

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

```
00117         {
00118             return items.AddItem(item);
00119         }
```

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 104 of file [Inventory.cs](#).

```
00105         {
00106             items.AddItem(slotIndex, item);
00107             /* saves the inventory changes
00108             Serialization.Serialization.SerializeInventory(this, inventoryName);
00109         }
```

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 94 of file [Inventory.cs](#).

```
00095         {
00096             return items;
00097         }
```

3.1.2.4 InventorySet()

bool BeeGame.Inventory.Inventory.InventorySet ()

Is the inventory set?

Returns

true if [items](#) == null

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

```
00036         {
00037             if (items == null)
00038                 return true;
00039             return false;
00040         }
```

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 79 of file `Inventory.cs`.

```
00080     {
00081         /* goes through all of the items in the array setting then all to a slot
00082         for (int i = 0; i < slots.Length; i++)
00083         {
00084             slots[i].slotIndex = i;
00085             slots[i].myInventory = this;
00086             slots[i].item = items.itemsInInventory[i];
00087         }
00088     }
```

3.1.2.6 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 59 of file [Inventory.cs](#).

```
00060      {
00061          this.items = items;
00062      }
```

3.1.2.7 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 47 of file [Inventory.cs](#).

```
00048      {
00049          items = new ItemsInInventory(slots.Length);
00050      }
```

3.1.2.8 UpdateBase()

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

Things in the inventory that should be updated

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

```
00070      {
00071          PutItemsInSlots ();
00072      }
```

3.1.3 Member Data Documentation

3.1.3.1 floatingItem

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

Item that is currently 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 [private]

[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](#).

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         return false;
00057     }
00058 }
```

3.2.4 Member Data Documentation

3.2.4.1 itemsInInventory

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

All of the items in the inventory

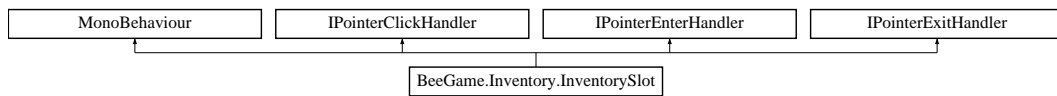
Definition at line 15 of file [ItemsInInventory.cs](#).

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](#) ()
Halves a [Item.itemStackCount](#) between the slot and the [Inventory.floatingItem](#)
- void [SwapItems](#) ()
Swaps the [Item](#) in the [Inventory.floatingItem](#) with the slots [item](#)
- void [CheckFloatingItem](#) ()
Checks if the [Inventory.floatingItem](#) should be null
- void [OnDisable](#) ()
Destroys the item text when the inventory is closed

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 numeToAdd ) [private]
```

Adds a number to items into the slot

Parameters

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

Definition at line 150 of file [InventorySlot.cs](#).

```
00151     {
00152         /** if the item in the slot is null create it
00153         if (item == null)
00154         {
00155             item = myInventory.floatingItem.CloneObject();
00156             item.itemStackCount = 0;
00157         }
00158
00159         /** add to number to add to the stack count
00160         item.itemStackCount += numeToAdd;
00161
00162         /** if the stack count is now larger than it should be dont let it be
00163         if (item.itemStackCount > item.maxStackCount)
00164         {
00165             item.itemStackCount = item.maxStackCount;
00166         }
00167
00168         /** remove the numebr if items form the floating item then check the floating item is not null
00169         myInventory.floatingItem.itemStackCount -= numeToAdd;
00170         CheckFloatingItem();
00171         /** save the inventory changes
00172         myInventory.AddItemToSlots(slotIndex,
00173         item);
00173     }
```

3.3.2.2 CheckFloatingItem()

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

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

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

```
00216     {
00217         if(myInventory.floatingItem.itemStackCount <= 0)
00218         {
00219             myInventory.floatingItem = null;
00220         }
00221     }
```

3.3.2.3 OnDisable()

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

Destroys the item text when the inventory is closed

Definition at line 254 of file [InventorySlot.cs](#).

```
00255     {
00256         Destroy(itemText);
00257     }
```

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 75 of file [InventorySlot.cs](#).

```
00076     {
00077         if (myInventory.floatingItem != null)
00078         {
00079             /* Left click moves whole stacks if items
00080             if (eventData.button == PointerEventData.InputButton.Left)
00081             {
00082                 /* If the item in the slot is empty put the floating item into it then clear it
00083                 if (item == null)
00084                 {
00085                     item = myInventory.floatingItem;
00086                     myInventory.floatingItem = null;
00087                     myInventory.AddItemToSlots(
00088                         slotIndex, item);
00089                     return;
00090                 }
00091                 /* if the items are the same
00092                 if(myInventory.floatingItem == item)
00093                 {
00094                     /* if the item in the inventoys stack count + the floating items stack count is
00095                     less than the max stack count
00096                     if (myInventory.floatingItem.
00097                     itemStackCount + item.itemStackCount <= item.
00098                     maxStackCount)
00099                     {
00100                         AddToSlot(myInventory.
00101                         floatingItem.itemStackCount);
00102                         return;
00103                     }
00104                     /* if the item stack added is larger than the max count add as many as you can and
00105                     move on
00106                     else
00107                     {
00108                         AddToSlot (item.maxStackCount -
00109                         item.itemStackCount);
00110                         return;
00111                     }
00112                 }
00113                 /* If the items were not == swap them
```

```

00107         else
00108         {
00109             SwapItems();
00110             return;
00111         }
00112     }
00113     else if(eventData.button == PointerEventData.InputButton.Right)
00114     {
00115         /* if the item in slot is null add 1 from the floating item to it
00116         if(item == null)
00117         {
00118             AddToSlot(1);
00119             return;
00120         }
00121         /* if the items are the same add 1 from the floating item to this item
00122         else if(item == myInventory.floatingItem)
00123         {
00124             AddToSlot(1);
00125             return;
00126         }
00127     }
00128 }
00129 /* if the floating item is null
00130 else
00131 {
00132     /* add 1/2 of the stack into the floating item if right click was pressed
00133     if(eventData.button == PointerEventData.InputButton.Right)
00134     {
00135         SplitStack();
00136         return;
00137     }
00138
00139     /* otherwise add the items into the floating item slot
00140     SwapItems();
00141     return;
00142 }
00143 }
00144 }

```

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 229 of file [InventorySlot.cs](#).

```

00230     {
00231         /* if the item is null or the floating item has something in it dont display the item text as
00232         it is not necessary
00233         if (item != null && myInventory.floatingItem == null)
00234         {
00235             itemText = Instantiate(PrefabDictionary.
00236             GetPrefab("ItemDetails"));
00237             /* sets the text to the correct position
00238             itemText.transform.GetChild(0).position = Input.mousePosition;
00239             /* puts the correct text in the box
00240             itemText.transform.GetChild(0).GetComponent<Text>().text = $"
00241             {item.GetItemName()}\nStack: {item.itemStackCount}";
00242         }
00243     }
00244 }

```

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 246 of file [InventorySlot.cs](#).

```
00247     {
00248         Destroy(itemText);
00249     }
```

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

Definition at line 181 of file [InventorySlot.cs](#).

```
00182     {
00183         myInventory.floatingItem = item.CloneObject();
00184         int give = (item.itemStackCount + 1) / 2;
00185         myInventory.floatingItem.itemStackCount = give;
00186         item.itemStackCount -= give;
00187
00188         if (item.itemStackCount <= 0)
00189             item = null;
00190
00191         myInventory.AddItemToSlots(slotIndex,
00192             item);
00192         Destroy(itemText);
00193     }
```

3.3.2.8 SwapItems()

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

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

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

```
00199     {
00200         /* temp copy of the item
00201         Item temp = myInventory.floatingItem;
00202         /* sets the floating item
00203         myInventory.floatingItem = item;
00204         /* sets the item that was in the floating item to the item in the slot
00205         item = temp;
00206         /* Saves the changes to the inventory
00207         myInventory.AddItemToSlots(slotIndex,
00208             item);
00208         /* destroys the text as it is not needed anymore
00209         Destroy(itemText);
00210     }
```

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             /* if the slot is selected in the hotbar give the player some indication by colouring it grey
00056             if (selectedSlot)
00057             {
00058                 GetComponent<Image>().color = Color.gray;
00059             }
00060             else
00061             {
00062                 GetComponent<Image>().color = Color.white;
00063             }
00064         }
00065     }
```

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](#).

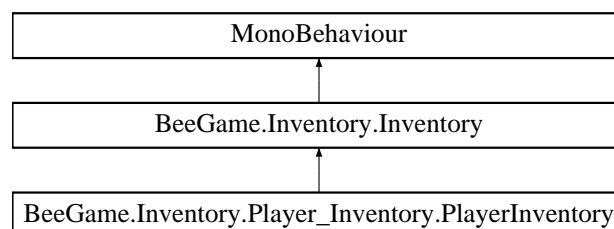
The documentation for this class was generated from the following file:

- 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 [Start](#) ()
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 [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 83 of file [PlayerInventory.cs](#).

```
00084     {
00085         /** get the item
00086         outItem = GetAllItems().itemsInInventory[slotIndex];
00087
00088         if (outItem == null)
00089             return false;
00090
00091         /** 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
00092         if(outItem.placeable)
00093             RemoveItemFromInventory(slotIndex);
00094
00095         return outItem.placeable;
00096     }
```

3.4.2.2 OpenPlayerInventory()

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

Show/Hide the player inventory

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

```
00104     {
00105         playerInventory.SetActive(!playerInventory.activeInHierarchy);
00106         THInput.isAnotherInventoryOpen = !
THInput.isAnotherInventoryOpen;
00107
00108         /** hides/ shows the mouse depending on if the inventory is open or not
00109         if (playerInventory.activeInHierarchy)
00110         {
00111             Cursor.lockState = CursorLockMode.None;
00112             Cursor.visible = true;
00113         }
00114         else
00115         {
00116             Cursor.visible = false;
00117             Cursor.lockState = CursorLockMode.Locked;
00118         }
00119     }
```

3.4.2.3 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 144 of file [PlayerInventory.cs](#).

```
00145         {
00146             /** if the item can be added to the inventory do that
00147             if (AddItemToInventory(item.item))
00148             {
00149                 /** if the item was added destroy its gameobject and save the inventory
00150                 Destroy(item.gameObject);
00151                 Serialization.Serialization.SerializeInventory(this,
inventoryName);
00152             }
00153         }
```

3.4.2.4 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 125 of file [PlayerInventory.cs](#).

```
00126         {
00127             /** if the item is already null nothing needs to be removed
00128             if (GetAllItems().itemsInInventory[index] != null)
00129             {
00130                 /** remove 1 item and if that was the last in the stack remove the item from the inventory
00131                 GetAllItems().itemsInInventory[index].
itemStackCount -= 1;
00132
00133                 if (GetAllItems().itemsInInventory[index].itemStackCount <= 0)
00134                     GetAllItems().itemsInInventory[index] = null;
00135
00136                 Serialization.Serialization.SerializeInventory(this,
inventoryName);
00137             }
00138         }
```

3.4.2.5 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 67 of file [PlayerInventory.cs](#).

```
00068         {
00069             for (int i = 0; i < slots.Length; i++)
00070             {
00071                 slots[i].selectedSlot = false;
00072             }
00073             slots[index].selectedSlot = true;
00074         }
00075     }
```

3.4.2.6 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 (20);
00037         }
```

3.4.2.7 Start()

```
void BeeGame.Inventory.Player_Inventory.PlayerInventory.Start ( ) [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         }
```

3.4.2.8 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 (THInput.GetButtonDown("Player Inventory"))
00049             OpenPlayerInventory();
00050
00051         /** checks if somethig shoul dbe picked up and put into the inventory
00052         RaycastHit[] hit = Physics.SphereCastAll(transform.position, 1f, transform.forward);
00053
00054         for (int i = hit.Length - 1; i >= 0; i--)
00055         {
00056             if (hit[i].collider.GetComponent<ItemGameObject>())
00057                 PickupItem(hit[i].collider.GetComponent<
ItemGameObject>());
00058         }
00059     }
00060 }
```

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:

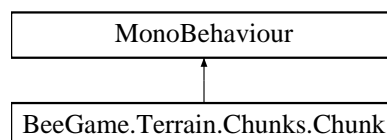
- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Player Inventory/[PlayerInventory.cs](#)

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)
Sets a Block in the given position
- void **SetBlocksUnmodified** ()
*Sets all of the Blocks in the **blocks** array to unmodified so that the whole chunk is not saved when it does not need to be*

Static Public Member Functions

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

Public Attributes

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

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 234 of file [Chunk.cs](#).

```
00235     {
00236         //if the chunk has been told to update the collisions but the chunk has no verts dont do it as
their is no point
00237         if (this.mesh.verts.Count == 0)
00238             return;
00239
00240         //if the render and collision meshes should be shared set the render mesh to the collision mesh
otherwise make a collision mesh
00241         if (this.mesh.shareMeshes)
00242         {
00243             world.chunkHasMadeCollisionMesh = true;
00244             applyCollisionMesh = false;
00245             meshCollider.sharedMesh = filter.mesh;
00246             return;
00247         }
00248
00249         world.chunkHasMadeCollisionMesh = true;
00250         //Applying the mesh takes the longest but nothing can be done with the mesh class in a
secondary thread...thanks Unity
00251
00252         //makes a new mesh setting the name for convenience
00253         Mesh mesh = new Mesh()
00254         {
00255             name = "Collider Mesh",
00256             vertices = this.mesh.colVerts.ToArray(),
00257             triangles = this.mesh.colTris.ToArray()
00258         };
00259
00260         //recalcs the normals and applies the mesh
00261         mesh.RecalculateNormals();
00262
00263         meshCollider.sharedMesh = mesh;
00264
00265         applyCollisionMesh = false;
00266     }
```


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
00130         an air block (empty block)
00131         if (checkNeighbouringChunks)
00132             return world.GetBlock(chunkWorldPos.x + x,
00133                                   chunkWorldPos.y + y, chunkWorldPos.z + z);
00134         return new Air();
00135     }
```

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 160 of file [Chunk.cs](#).

```

00161     {
00162         //if the value is less then 0 or greater than 16 the value is outside the chunk
00163         if (i < 0 || i >= chunkSize)
00164             return false;
00165         return true;
00166     }

```

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 [211](#) of file [Chunk.cs](#).

```

00212     {
00213         //Applying the mesh takes the longest but nothing can be dont with the mesh class in a
secondary thread...thanks unity
00214
00215         mesh.done = false;
00216         //clears the current chunk mesh
00217         filter.mesh.Clear();
00218         //name for convenience
00219         filter.mesh.name = "Render Mesh";
00220         //puts the tris and verts from the meshdata into the chunk mesh
00221         filter.mesh.vertices = meshData.verts.ToArray();
00222         filter.mesh.triangles = meshData.tris.ToArray();
00223
00224         //sets the uvs
00225         filter.mesh.uv = meshData.uv.ToArray();
00226
00227         //redoes the normals incase they got messed up
00228         filter.mesh.RecalculateNormals();
00229     }

```

4.1.2.5 SetBlock()

```

void BeeGame.Terrain.Chunks.Chunk.SetBlock (
    int x,
    int y,
    int z,
    Block block )

```

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             //if the block is not in the chunk find its chunk and set it their
00152             world.SetBlock(chunkWorldPos.x + x,
00153                 chunkWorldPos.y + y, chunkWorldPos.z + z, block);
00153         }
```

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 176 of file [Chunk.cs](#).

```
00177         {
00178             foreach (var block in blocks)
00179             {
00180                 block.changed = false;
00181             }
00182         }
```

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 187 of file [Chunk.cs](#).

```
00188     {
00189         //says that this chunk is rendered and initialtes the mesh
00190         rendered = true;
00191
00192         //goes through every block in the blocks array getting their mesh data
00193         for (int x = 0; x < chunkSize; x++)
00194         {
00195             for (int z = 0; z < chunkSize; z++)
00196             {
00197                 for (int y = 0; y < chunkSize; y++)
00198                 {
00199                     blocks[x, y, z].UpdateBlock(x, y, z, this);
00200                     mesh = blocks[x, y, z].BlockData(this, x, y, z,
mesh);
00201                 }
00202             }
00203         }
00204         mesh.done = true;
00205     }
```

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 thic 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

<code>addToRenderMesh</code>	Should the triangles be added to the render Mesh
------------------------------	--

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

```
00047     {
00048         /*adds the triangles in an anticlockwise order
00049
00050         if (addToRenderMesh)
00051         {
00052             tris.Add(verts.Count - 4);
00053             tris.Add(verts.Count - 3);
00054             tris.Add(verts.Count - 2);
00055             tris.Add(verts.Count - 4);
00056             tris.Add(verts.Count - 2);
00057             tris.Add(verts.Count - 1);
00058         }
00059
00060         colTris.Add(colVerts.Count - 4);
00061         colTris.Add(colVerts.Count - 3);
00062         colTris.Add(colVerts.Count - 2);
00063         colTris.Add(colVerts.Count - 4);
00064         colTris.Add(colVerts.Count - 2);
00065         colTris.Add(colVerts.Count - 1);
00066     }
```

4.2.2.2 AddTriangle()

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

Adds a triangle to both the render and collidson meshes

Parameters

<code>tri</code>	triangle
------------------	----------

not used anymore remove?

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

```
00092     {
00093         tris.Add(tri);
00094
00095         colTris.Add(tri - (verts.Count - colVerts.Count));
00096     }
```

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 thic 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>()
```

Vertices for the [Chunk](#) render Mesh

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

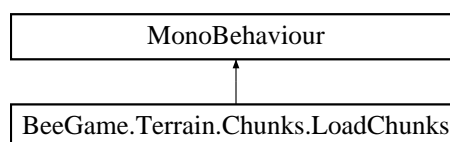
The documentation for this class was generated from the following file:

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

4.3 BeeGame.Terrain.Chunks.LoadChunks Class Reference

Loads the [Chunks](#) around the player

Inheritance diagram for BeeGame.Terrain.Chunks.LoadChunks:



Public Attributes

- [World world](#)

The world the player is in

Private Member Functions

- void [Start](#) ()
Sets the world
- void [Update](#) ()
Builds, Renders, and Remmoves [Chunks](#)
- void [ApplyCollsionMeshToNearbyChunks](#) ()
Makes a 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

4.3.1 Detailed Description

Loads the [Chunks](#) around the player

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

4.3.2 Member Function Documentation

4.3.2.1 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 [109](#) of file [LoadChunks.cs](#).

```
00110     {
00111         //gets the player position in chunk coordinates
00112         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);
00113
00114         for (int i = 0; i < nearbyChunks.Length; i++)
00115         {
00116             ChunkWorldPos chunkPos = new ChunkWorldPos(nearbyChunks[i].x * Chunk.chunkSize
+ playerPos.x, 0, nearbyChunks[i].z * Chunk.chunkSize + playerPos.z);
00117
00118             for (int j = -1; j < 2; j++)
00119             {
00120                 Chunk nearbyChunk = world.GetChunk(chunkPos.x, j * Chunk.chunkSize,
chunkPos.z);
00121
00122                 if (nearbyChunk != null)
00123                     nearbyChunk.applyCollisionMesh = true;
00124             }
00125         }
00126     }
```

4.3.2.2 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 [184](#) of file [LoadChunks.cs](#).

```
00185     {
00186         if (world.GetChunk(pos.x, pos.y, pos.z) == null)
00187             world.CreateChunk(pos.x, pos.y, pos.z);
00188     }
```

4.3.2.3 DeleteChunks()

```
bool BeeGame.Terrain.Chunks.LoadChunks.DeleteChunks ( ) [private]
```

Destroys [Chunks](#) every 10 calls

Returns

true if [Chunks](#) were destroyed

Definition at line 194 of file [LoadChunks.cs](#).

```
00195     {
00196         //destroys every 10 call to reduce load on CPU so that chunks are not destroyed and created at
the same time
00197         if(timer == 10)
00198         {
00199             timer = 0;
00200             var chunksToDelete = new List<ChunkWorldPos>();
00201
00202             //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
00203             foreach (var chunk in world.chunks)
00204             {
00205                 float distance = Vector3.Distance(chunk.Value.transform.position, transform.position);
00206
00207                 if (distance > 256)
00208                     chunksToDelete.Add(chunk.Key);
00209             }
00210
00211             foreach (var chunk in chunksToDelete)
00212             {
00213                 world.DestroyChunk(chunk.x, chunk.y, chunk.z);
00214             }
00215
00216             return true;
00217         }
00218
00219         timer++;
00220
00221         return false;
00222     }
```

4.3.2.4 FindChunksToLoad()

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

Finds the [Chunks](#) that should be rendered

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

```
00149     {
00150         if (buildList.Count == 0)
00151         {
00152             //gets the player position in chunk coordinates
00153             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);
00154
00155             //check all of the chunk positions and if that position does not have a chunk in it make it
00156             for (int i = 0; i < chunkPositions.Length; i++)
00157             {
00158                 ChunkWorldPos newChunkPos = new ChunkWorldPos(chunkPositions[i].x * Chunk
.chunkSize + playerPos.x, 0, chunkPositions[i].z * Chunk.chunkSize + playerPos.z);
00159
00160                 Chunk newChunk = world.GetChunk(newChunkPos.x, newChunkPos.y, newChunkPos.
z);
00161
00162                 if (newChunk != null && (newChunk.rendered || buildList.Contains(newChunkPos))
)
00163                     continue;
00164
00165                 for (int y = -1; y < 2; y++)
00166                 {
00167                     for (int x = newChunkPos.x - Chunk.chunkSize; x < newChunkPos.x + Chunk.chunkSize;
x += Chunk.chunkSize)
00168                     {
00169                         for (int z = newChunkPos.z - Chunk.chunkSize; z < newChunkPos.z + Chunk.
chunkSize; z += Chunk.chunkSize)
```

```

00170             {
00171                 buildList.Add(new ChunkWorldPos(x, y * Chunk.chunkSize, z));
00172             }
00173         }
00174     }
00175     return;
00176 }
00177 }
00178 }

```

4.3.2.5 LoadAndRenderChunks()

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

Gets the chunks that could be built and renders then renders them

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

```

00132     {
00133         //if there is something in the build list new chunks can be made
00134         if (buildList.Count != 0)
00135         {
00136             //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
00137             for (int i = buildList.Count - 1, j = 0; i >= 0 && j < 8; i--, j++)
00138             {
00139                 BuildChunk(buildList[0]);
00140                 buildList.RemoveAt(0);
00141             }
00142         }
00143     }

```

4.3.2.6 Start()

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

Sets the world

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

```

00081     {
00082         LandGeneration.Terrain.world = world;
00083     }

```

4.3.2.7 Update()

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

Builds, Renders, and Removes [Chunks](#)

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

```

00089     {
00090         if (DeleteChunks())
00091             return;
00092         if (!world.chunkHasMadeCollisionMesh)
00093         {
00094             FindChunksToLoad();
00095             LoadAndRenderChunks();
00096             ApplyCollisionMeshToNearbyChunks();
00097         }
00098         //stops chunks being made and collision meshes being made at the same time
00099         world.chunkHasMadeCollisionMesh = false;
00100     }

```

4.3.3 Member Data Documentation

4.3.3.1 buildList

```
List<ChunkWorldPos> BeeGame.Terrain.Chunks.LoadChunks.buildList = new List<ChunkWorldPos>()
[private]
```

List if chunks to build

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

4.3.3.2 chunkPositions

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

Positions to make chunks around the player ///

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

4.3.3.3 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 69 of file [LoadChunks.cs](#).

4.3.3.4 timer

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

Timer for chunk removal

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

4.3.3.5 world

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

The world the player is in

Definition at line 16 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`

4.4 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

4.4.1 Detailed Description

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

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

4.4.2 Constructor & Destructor Documentation

4.4.2.1 SaveChunk()

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

Will search all the the given Blocks for modified blocks

Parameters

<code>blockArray</code>	Chunks blocks (Must be [16, 16, 16])
-------------------------	--

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

```
00024     {
00025         for (int x = 0; x < Chunk.chunkSize; x++)
00026         {
00027             for (int y = 0; y < Chunk.chunkSize; y++)
00028             {
00029                 for (int z = 0; z < Chunk.chunkSize; z++)
00030                 {
00031                     /**if the block has changed save it
00032                     if (blockArray[x, y, z].changed)
00033                         blocks.Add(new ChunkWorldPos(x, y, z), blockArray[x, y, z]);
00034                 }
00035             }
00036         }
00037     }
```

4.4.3 Member Data Documentation

4.4.3.1 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](#)

4.5 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`

4.5.1 Detailed Description

Serializable int version of THVector3

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

4.5.2 Constructor & Destructor Documentation

4.5.2.1 ChunkWorldPos()

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

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

Parameters

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

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

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

4.5.3 Member Function Documentation

4.5.3.1 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     }

```

4.5.3.2 GetHashCode()

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

Makes a unique hascode for the vector

Returns

unique int value for the vector

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

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

```

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

```

4.5.3.3 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);
    }

```

4.5.3.4 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     }

```

4.5.3.5 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     }

```

4.5.4 Member Data Documentation

4.5.4.1 x

```

int BeeGame.Terrain.ChunkWorldPos.x

```

x, y, z values for the vector

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

4.5.4.2 y

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

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

4.5.4.3 z

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

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

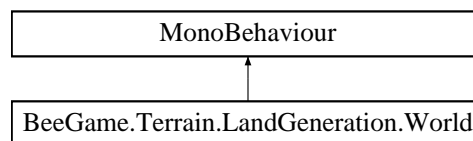
The documentation for this struct was generated from the following file:

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

4.6 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 at the given x, y, z position

Parameters

x	X pos if the chunk
y	Y pos if the chunk
z	Z pos if the chunk

Definition at line 97 of file [World.cs](#).

```

00098     {
00099         /*if teh chnks exists destroy it
00100         if (chunks.TryGetValue(new ChunkWorldPos(x, y, z), out Chunk chunk))
00101         {
00102             /*saves the chunk before destroying it incase any block were changed in it
00103             Serialization.Serialization.SaveChunk(chunk);
00104             Destroy(chunk.gameObject);
00105             chunks.Remove(new ChunkWorldPos(x, y, z));
00106         }
00107     }

```

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.
x, y - chunk.chunkWorldPos.y, z - chunk.chunkWorldPos.
z);
00193         }
00194
00195         /*returns an empty block is the chunk was not found
00196         return new Air();
00197     }
```

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

x	X pos of the chunk
y	Y pos of the chunk
z	Z pos of the chunk

Returns

Chunk at given x, y, z

Definition at line 160 of file [World.cs](#).

```
00161     {
00162         float multiple = Chunk.chunkSize;
00163         /*rounds the given x, y, z to a multiple of 16 as chunks are 16x16x16 in size
00164         ChunkWorldPos pos = new ChunkWorldPos()
00165         {
00166             x = Mathf.FloorToInt(x / multiple) * Chunk.chunkSize,
```



```

00167         y = Mathf.FloorToInt(y / multiple) * Chunk.chunkSize,
00168         z = Mathf.FloorToInt(z / multiple) * Chunk.chunkSize
00169     };
00170
00171     /*gets the chunk if it exists
00172     chunks.TryGetValue(pos, out Chunk chunk);
00173     /*if the chunk does not exist will return null
00174     return chunk;
00175 }

```

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<ChunkWorldPos, 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 15 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 250 of file [Terrain.cs](#).

```
00251     {
00252         if (chunk == null)
00253             return false;
00254
00255         if (chunk.GetBlock((int)pos.x, (int)pos.y, (int)pos.z) != new
00256             Air())
00257             return true;
00258         return false;
00259     }
```

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 224 of file [Terrain.cs](#).

```
00225     {
00226         /*checks that a chunk was hit and if it wasnt return early
00227         Chunk chunk = hit.collider.GetComponent<Chunk>();
```

```

00228
00229         if (chunk == null)
00230             return null;
00231
00232         /**allignes the hit to the block grid and returns the block
00233         ChunkWorldPos pos = GetBlockPos(hit, adjacent);
00234
00235         return chunk.world.GetBlock(pos.x, pos.y, pos.z);
00236     }

```

4.7.2.3 GetBlock() [2/2]

```

static Block BeeGame.Terrain.LandGeneration.Terrain.GetBlock (
    THVector3 pos ) [static]

```

Definition at line 238 of file [Terrain.cs](#).

```

00239     {
00240         Chunk chunk = GetChunk(pos);
00241
00242         if (chunk == null)
00243             return new Air();
00244
00245         chunk.world.GetBlock((int)pos.x, (int)pos.y, (int)pos.z);
00246
00247         return new Block();
00248     }

```

4.7.2.4 GetBlockPos() [1/3]

```

static ChunkWorldPos BeeGame.Terrain.LandGeneration.Terrain.GetBlockPos (
    THVector3 pos ) [static]

```

Gets a block postion from a THVector3

Parameters

<i>pos</i>	Position of the block as a THVector3
------------	--------------------------------------

Returns

[ChunkWorldPos](#) of the Block

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

```

00026     {
00027         return new ChunkWorldPos()
00028         {
00029             x = Mathf.RoundToInt(pos.x),
00030             y = Mathf.RoundToInt(pos.y),
00031             z = Mathf.RoundToInt(pos.z)
00032         };
00033     }

```

4.7.2.5 GetBlockPos() [2/3]

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

Returns the positon of the block hit as a THVector3

Parameters

<i>hit</i>	RaycastHit
<i>adjacent</i>	Do you want the face adjacent to the block hit

Returns

THVector3 of the block you hit in world cordinates

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

```
00042     {
00043         THVector3 vec3 = new THVector3()
00044     {
00045         x = RoundXZ(hit.point.x, hit.normal.x),
00046         y = RoundY(hit.point.y, hit.normal.y),
00047         z = RoundXZ(hit.point.z, hit.normal.z)
00048     };
00049     return (vec3);
00050 }
```

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 207 of file [Terrain.cs](#).

```
00208     {
00209         return GetBlockPos(new THVector3())
```

```

00210         {
00211             /**rounds the hit to the correct position
00212             x = Round(hit.point.x, hit.normal.x, adjacent),
00213             y = Round(hit.point.y, hit.normal.y, adjacent),
00214             z = Round(hit.point.z, hit.normal.z, adjacent)
00215         });
00216     }

```

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 57 of file [Terrain.cs](#).

```

00058     {
00059         return new ChunkWorldPos((int)RoundXZ(hit.point.x, hit.normal.x), (int)
RoundY(hit.point.y, hit.normal.y), (int)RoundXZ(hit.point.z, hit.normal.z));
00060     }

```

4.7.2.8 GetChunk()

```

static Chunk BeeGame.Terrain.LandGeneration.Terrain.GetChunk (
    THVector3 vec3 ) [static]

```

Definition at line 262 of file [Terrain.cs](#).

```

00263     {
00264         return world.GetChunk((int)vec3.x, (int)vec3.y, (int)vec3.
z);
00265     }

```

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 received

Returns

rounded value of *pos* as a float

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

Definition at line 182 of file [Terrain.cs](#).

```
00183     {
00184         if(pos - (int)pos == 0.5f || pos - (int)pos == -0.5f)
00185         {
00186             if(adjacent)
00187             {
00188                 pos += (norm / 2);
00189             }
00190             else
00191             {
00192                 pos -= (norm / 2);
00193             }
00194         }
00195         return pos;
00196     }
00197 }
```

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 71 of file [Terrain.cs](#).

```
00072     {
00073         /**if we are looking at + x/z vlaues
00074         if (pos > 0)
```



```

00075     {
00076         if (normal > 0)
00077         {
00078             pos = (int)pos;
00079             return pos;
00080         }
00081         else if (normal < 0)
00082         {
00083             pos = (int)pos;
00084             return pos - 1;
00085         }
00086         else
00087         {
00088             if ((pos - (int)pos) > 0.5)
00089             {
00090                 return (int)pos + 1;
00091             }
00092             return (int)pos;
00093         }
00094     }
00095     /*if we are looking at - x/z values
00096     else
00097     {
00098         /*if poitive normal
00099         if (normal > 0)
00100         {
00101             pos = (int)pos;
00102             return pos - 1;
00103         }
00104
00105         /*if negative nomrmal
00106         if (normal < 0)
00107         {
00108             pos = (int)pos;
00109             return pos;
00110         }
00111         /*if their is no normal
00112
00113         /*if pos is greater than 0.5 we are in the next block so go to it
00114         if ((-pos - (int)-pos) > 0.5)
00115         {
00116             return (int)pos - 1;
00117         }
00118
00119         return (int)pos;
00120     }
00121 }

```

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 132 of file [Terrain.cs](#).

```

00133     {
00134         pos = (float)Math.Round(pos, 1);
00135         if (pos >= 0)
00136         {
00137             if(normal > 0)
00138             {
00139                 if((int)pos % 2 == 0)
00140                     return Mathf.RoundToInt((float)Math.Round(pos, 1));
00141
00142                 return Mathf.RoundToInt((float)Math.Round(pos, 1)) - normal;
00143             }
00144
00145             if((int)pos % 2 == 0)
00146                 return Mathf.RoundToInt((float)Math.Round(pos, 1)) - normal;
00147
00148             return Mathf.RoundToInt((float)Math.Round(pos, 1));
00149         }
00150
00151         if(pos <= 0)
00152         {
00153             if (normal > 0)
00154             {
00155                 if ((int)pos % 2 == 0)
00156                     /*the Math.Round removes strange rounding errors shown with Mathf.Round eg
00157 sometimes 0.5 would round to 0 not 1
00158                     return Mathf.RoundToInt((float)Math.Round(pos, 1)) - normal;
00159
00160                     return Mathf.RoundToInt((float)Math.Round(pos, 1));// - normal;
00161             }
00162
00163             if ((int)pos % 2 == 0)
00164                 return Mathf.RoundToInt((float)Math.Round(pos, 1));
00165
00166             return Mathf.RoundToInt((float)Math.Round(pos, 1)) - normal;
00167         }
00168
00169         return Mathf.RoundToInt((float)Math.Round(pos, 1));
00170     }

```

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 275 of file [Terrain.cs](#).

```

00276     {
00277         /*checks that a chnk was hit
00278         Chunk chunk = hit.collider.GetComponent<Chunk>();
00279
00280         if (chunk == null)

```

```

00281         return false;
00282
00283         /*aligns the hit to the block grid
00284         ChunkWorldPos pos = GetBlockPosFromRayCast(hit);
00285
00286         /*checks that the block tryign to be replaced can be replaced eg bedrock cannot be replaced
00287         if (GetBlock(hit, adjacent).breakable)
00288         {
00289             /*sets the position of the block and saves the chunk
00290             chunk.world.SetBlock(pos.x, pos.y, pos.z, block);
00291             Serialization.Serialization.SaveChunk(chunk);
00292         }
00293
00294         return true;
00295     }

```

4.7.3 Member Data Documentation

4.7.3.1 world

`World` BeeGame.Terrain.LandGeneration.Terrain.world [static]

Definition at line 17 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 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
Minimun 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 `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

<code>chunk</code>	Chunk to populate with Blocks
--------------------	-------------------------------

Returns

Chunk with Blocks generated

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

```
00071     {
00072         Chunk outChunk = chunk;
00073         lock (chunk)
00074         {
00075             Thread thread = new Thread(() => ChunkGenThread(chunk, out outChunk)) { Name
= $"Generate Chunk Thread @ {chunk.chunkWorldPos}"};
00076
00077             thread.Start();
00078             return outChunk;
00079         }
00080     }
```

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 87 of file [TerrainGeneration.cs](#).

```
00088     {
00089         /**for each x and z position in teh chunk
00090         for (int x = chunk.chunkWorldPos.x; x < chunk.
chunkWorldPos.x + Chunk.chunkSize; x++)
00091         {
00092             for (int z = chunk.chunkWorldPos.z; z < chunk.
chunkWorldPos.z + Chunk.chunkSize; z++)
00093             {
00094                 chunk = GenChunkColumn(chunk, x, z);
00095             }
00096         }
00097
00098         chunk.SetBlocksUnmodified();
00099         outChunk = chunk;
00100     }
```

4.8.2.3 GenChunkColumn()

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

Generates a column of the Chunk

Parameters

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

Returns

Chunk with a new column of blocks generated

Definition at line 109 of file [TerrainGeneration.cs](#).

```
00110     {
00111         /*the height of the mountain
00112         int stoneHeight = Mathf.FloorToInt(stoneBaseHeight);
00113         stoneHeight += GetNoise(-x, 0, z, stoneMountainFrequency, Mathf.
FloorToInt(stoneMountainHeight));
00114
00115         /*if the column is currently too low make it not so low
00116         if (stoneHeight < stoneMinHeight)
00117             stoneHeight = Mathf.FloorToInt(stoneMinHeight);
00118
00119         /*add the height of normal stone on to the mountain
00120         stoneHeight += GetNoise(x, 0, -z, stoneBaseNoise, Mathf.RoundToInt(
stoneBaseNoiseHeight));
00121
00122         /*put dirt on top
00123         int dirtHeight = stoneHeight + Mathf.FloorToInt(dirtBaseHeight);
00124         dirtHeight += GetNoise(x, 100, z, dirtNoise, Mathf.FloorToInt(
dirtNoiseHeight));
00125
00126         /*set the column to the correct blocks
00127         for (int y = chunk.chunkWorldPos.y; y < chunk.
chunkWorldPos.y + Chunk.chunkSize; y++)
00128         {
00129             int caveChance = GetNoise(x + 40, y + 100, z - 50,
caveFrequency, 200);
00130
00131             /*puts a layer of bedrock at the bottom the the world
00132             if (y <= (chunk.chunkWorldPos.y) && chunk.
chunkWorldPos.y == -16)
00133             {
00134                 SetBlock(x, y, z, new Blocks.Bedrock(), chunk);
00135             }
00136             else if (y <= stoneHeight && caveSize < caveChance)
00137             {
00138                 SetBlock(x, y, z, new Blocks.Block(), chunk);
00139             }
00140             else if (y <= dirtHeight && caveSize < caveChance)
00141             {
00142                 SetBlock(x, y, z, new Blocks.Grass(), chunk);
00143             }
00144             else
00145             {
00146                 SetBlock(x, y, z, new Blocks.Air(), chunk);
00147             }
00148         }
00149         return chunk;
00150     }
00151 }
```

4.8.2.4 GetNoise()

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

Get a noise value

Parameters

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

Returns

A noise value as an int

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

```
00163     {
00164         return Mathf.FloorToInt((SimplexNoise.Generate(x * scale, y * scale, z *
00165         scale) + 1f) * (max / 2f));
    }
```

4.8.2.5 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 176 of file [TerrainGeneration.cs](#).

```
00177     {
00178         /*corrects the x, y, z pos of the so that the block is placed in the correct position
00179         x -= chunk.chunkWorldPos.x;
00180         y -= chunk.chunkWorldPos.y;
00181         z -= chunk.chunkWorldPos.z;
00182
00183         /*checks that the block is in the chunk and that no block is already their then sets it
00184         if (Chunk.InRange(x) && Chunk.InRange(y) &&
00185         Chunk.InRange(z))
00186             if (replacesBlocks || chunk.blocks[x, y, z] == null)
00187                 chunk.SetBlock(x, y, z, block);
    }
```

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 58 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 62 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]
```

Minimun 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](#).

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 = 0.3333333333f;
00145         const float G3 = 0.1666666667f;
00146
00147         float n0, n1, n2, n3; /** Noise contributions from the four corners
00148
00149         /** Skew the input space to determine which simplex cell we're in
00150         float s = (x + y + z) * F3; /** Very nice and simple skew factor for 3D
00151         float xs = x + s;
00152         float ys = y + s;
00153         float zs = z + s;
00154         int i = FastFloor(xs);
00155         int j = FastFloor(ys);
00156         int k = FastFloor(zs);
00157
00158         float t = (float)(i + j + k) * G3;
00159         float X0 = i - t; /** Unskew the cell origin back to (x,y,z) space
00160         float Y0 = j - t;
00161         float Z0 = k - t;
00162         float x0 = x - X0; /** The x,y,z distances from the cell origin
00163         float y0 = y - Y0;
00164         float z0 = z - Z0;
00165
00166         /** For the 3D case, the simplex shape is a slightly irregular tetrahedron.
00167         /** Determine which simplex we are in.
00168         int i1, j1, k1; /** Offsets for second corner of simplex in (i,j,k) coords
00169         int i2, j2, k2; /** Offsets for third corner of simplex in (i,j,k) coords
00170
00171         /** This code would benefit from a backport from the GLSL version! */
00172         if (x0 >= y0)
00173         {
00174             if (y0 >= z0)
00175             { i1 = 1; j1 = 0; k1 = 0; i2 = 1; j2 = 1; k2 = 0; } /** X Y Z order
00176             else if (x0 >= z0) { i1 = 1; j1 = 0; k1 = 0; i2 = 1; j2 = 0; k2 = 1; } /** X Z Y order
00177             else { i1 = 0; j1 = 0; k1 = 1; i2 = 1; j2 = 0; k2 = 1; } /** Z X Y order
00178         }
00179         else
00180         { /** x0 < y0
00181             if (y0 < z0) { i1 = 0; j1 = 0; k1 = 1; i2 = 0; j2 = 1; k2 = 1; } /** Z Y X order
00182             else if (x0 < z0) { i1 = 0; j1 = 1; k1 = 0; i2 = 0; j2 = 1; k2 = 1; } /** Y Z X order
00183             else { i1 = 0; j1 = 1; k1 = 0; i2 = 1; j2 = 1; k2 = 0; } /** Y X Z order
00184         }
00185
00186         /** A step of (1,0,0) in (i,j,k) means a step of (1-c,-c,-c) in (x,y,z),
00187         /** a step of (0,1,0) in (i,j,k) means a step of (-c,1-c,-c) in (x,y,z), and
00188         /** a step of (0,0,1) in (i,j,k) means a step of (-c,-c,1-c) in (x,y,z), where
00189         /** c = 1/6.
00190
00191         float x1 = x0 - i1 + G3; /** Offsets for second corner in (x,y,z) coords
00192         float y1 = y0 - j1 + G3;
00193         float z1 = z0 - k1 + G3;
00194         float x2 = x0 - i2 + 2.0f * G3; /** Offsets for third corner in (x,y,z) coords
00195         float y2 = y0 - j2 + 2.0f * G3;
00196         float z2 = z0 - k2 + 2.0f * G3;
00197         float x3 = x0 - 1.0f + 3.0f * G3; /** Offsets for last corner in (x,y,z) coords
00198         float y3 = y0 - 1.0f + 3.0f * G3;
00199         float z3 = z0 - 1.0f + 3.0f * G3;
00200
00201         /** Wrap the integer indices at 256, to avoid indexing perm[] out of bounds
00202         int ii = Mod(i, 256);
00203         int jj = Mod(j, 256);
00204         int kk = Mod(k, 256);
00205
00206         /** Calculate the contribution from the four corners
00207         float t0 = 0.6f - x0 * x0 - y0 * y0 - z0 * z0;
00208         if (t0 < 0.0f) n0 = 0.0f;
00209         else
00210         {
00211             t0 *= t0;
00212             n0 = t0 * t0 * grad(perm[ii + perm[jj + perm[kk]]], x0, y0, z0);
00213         }
00214     }
```

```

00215         float t1 = 0.6f - x1 * x1 - y1 * y1 - z1 * z1;
00216         if (t1 < 0.0f) n1 = 0.0f;
00217         else
00218         {
00219             t1 *= t1;
00220             n1 = t1 * t1 * grad(perm[ii + i1 + perm[jj + j1 +
perm[kk + k1]]], x1, y1, z1);
00221         }
00222
00223         float t2 = 0.6f - x2 * x2 - y2 * y2 - z2 * z2;
00224         if (t2 < 0.0f) n2 = 0.0f;
00225         else
00226         {
00227             t2 *= t2;
00228             n2 = t2 * t2 * grad(perm[ii + i2 + perm[jj + j2 +
perm[kk + k2]]], x2, y2, z2);
00229         }
00230
00231         float t3 = 0.6f - x3 * x3 - y3 * y3 - z3 * z3;
00232         if (t3 < 0.0f) n3 = 0.0f;
00233         else
00234         {
00235             t3 *= t3;
00236             n3 = t3 * t3 * grad(perm[ii + 1 + perm[jj + 1 + perm[kk + 1]]], x3, y3, z3)
;
00237         }
00238
00239         /* Add contributions from each corner to get the final noise value.
00240         /* The result is scaled to stay just inside [-1,1]
00241         return 32.0f * (n0 + n1 + n2 + n3); /* TODO: The scale factor is preliminary!
00242     }

```

4.9.2.5 grad() [1/4]

```

static float BeeGame.Terrain.LandGeneration.Noise.SimplexNoise.grad (
    int hash,
    float x ) [static], [private]

```

Definition at line 283 of file [SimplexNoise.cs](#).

```

00284     {
00285         int h = hash & 15;
00286         float grad = 1.0f + (h & 7); /* Gradient value 1.0, 2.0, ..., 8.0
00287         if ((h & 8) != 0) grad = -grad; /* Set a random sign for the gradient
00288         return (grad * x); /* Multiply the gradient with the distance
00289     }

```

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:

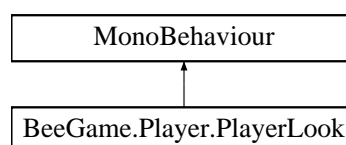
- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/LandGeneration/↔ Noise/[SimplexNoise.cs](#)

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 the cursor and hides it
- void [Update](#) ()
Every fixed update check if the look should 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 the 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](#).

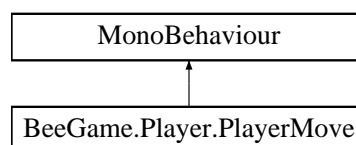
The documentation for this class was generated from the following file:

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

5.2 BeeGame.Player.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

5.2.1 Detailed Description

Moves the player

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

5.2.2 Member Function Documentation

5.2.2.1 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      }
```

5.2.2.2 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      }
```

5.2.2.3 MovePlayer()

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

Moves the player

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

```
00088      {
00089          //Calculate the speed we want to achieve
00090          Vector3 targetVelocity = new Vector3(THInput.GetAxis("Horizontal"), 0,
00091 THInput.GetAxis("Vertical"));
00092          targetVelocity = transform.TransformDirection(targetVelocity);
00093          targetVelocity *= speed;
00094
00095          //Apply a force to reach the target speed
00096          Vector3 velocity = myRigidBody.velocity;
00097          Vector3 velocityChange = (targetVelocity - velocity);
00098
00099          //Clamping the velocity so that the player does not infinitely accelerate
00100          velocityChange.x = Mathf.Clamp(velocityChange.x, -maxVelocity,
00101 maxVelocity);
00102          velocityChange.z = Mathf.Clamp(velocityChange.z, -maxVelocity,
00103 maxVelocity);
00104          velocityChange.y = 0;
00105
00106          //Adds the force to the player so they move in the correct direction
00107          myRigidBody.AddForce(velocityChange, ForceMode.Impulse);
00108
00109          //Jumping
00110          if (canJump && THInput.GetButton("Jump"))
00111          {
00112              canJump = false;
00113              myRigidBody.velocity = new Vector3(velocity.x,
00114 VerticalJumpSpeed(), velocity.z);
00115          }
00116      }
```

5.2.2.4 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         }
```

5.2.2.5 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         }
```

5.2.3 Member Data Documentation

5.2.3.1 canJump

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

Can the player jump?

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

5.2.3.2 gravity

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

Gravity of the player

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

5.2.3.3 jumpHeight

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

How high can the player jump

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

5.2.3.4 maxVelocity

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

Max velocity of the player

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

5.2.3.5 myRigidBody

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

Rigidbody for the player

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

5.2.3.6 speed

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

Speed of the player

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

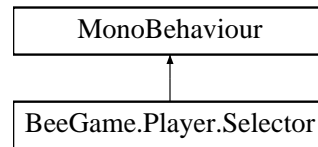
The documentation for this class was generated from the following file:

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

5.3 BeeGame.Player.Selector Class Reference

Moves the Block selector

Inheritance diagram for BeeGame.Player.Selector:



Public Attributes

- GameObject [selector](#)
Selector
- 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

5.3.1 Detailed Description

Moves the Block selector

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

5.3.2 Member Function Documentation

5.3.2.1 Awake()

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

Make the selector

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

```
00042         {
00043             selector = Instantiate(selector);
00044         }
```

5.3.2.2 BreakBlock()

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

Breaks the Block in the selectors postion

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

```
00118         {
00119             Chunk chunk = GetChunk(selector.transform.position);
00120
00121             Block block = chunk.world.GetBlock((int)selector.transform.position.x, (int)
selector.transform.position.y, (int)selector.transform.position.z);
00122
00123             if (!block.breakable)
00124                 return;
00125
00126             chunk.world.SetBlock((int)selector.transform.position.x, (int)
selector.transform.position.y, (int)selector.transform.position.z, new
Air(), true);
00127             /* set to changed so when block is placed down again it will be saved
00128             block.changed = true;
00129             block.BreakBlock(selector.transform.position);
00130         }
```

5.3.2.3 FixedUpdate()

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

Updates the selector if an inventory is not open

Definition at line 49 of file [Selector.cs](#).

```
00050         {
00051             if(!isAnotherInventoryOpen)
00052                 UpdateSelector();
00053         }
```

5.3.2.4 PlaceBlock()

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

Places a Block in the selector position

Definition at line 135 of file [Selector.cs](#).

```
00136         {
00137             Chunk chunk = GetChunk(selector.transform.position);
00138
00139             if (chunk == null)
00140                 return;
00141
00142             /* gets the item in the hotbar and if the item is placeable place it
00143             if(transform.parent.GetComponentInChildren<PlayerInventory>().
00144             GetItemFromHotBar(selectedHotbarSlot, out
00145             Item blockToPlace))
00146                 chunk.world.SetBlock((int)(selector.transform.position.x +
00147             hit.normal.x), (int)(selector.transform.position.y + hit.normal.y), (int)(
00148             selector.transform.position.z + hit.normal.z), (Block)blockToPlace, true);
00149         }
```

5.3.2.5 SelectedSlot()

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

Changes what slot in the hotbar is currently selected by the player

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

```
00093         {
00094             /* adds 1 to the selected slot and if that is out of range set it to the first hotbar slot
00095             if(Input.GetAxis("Mouse ScrollWheel") > 0)
00096             {
00097                 selectedHotbarSlot += 1;
00098                 if (selectedHotbarSlot == 36)
00099                     selectedHotbarSlot = 27;
00100             }
00101             /* removes one from the hotbar selector and if the selector would be inside the inventory set
00102             it to the last slot in the hotbar
00103             else if (Input.GetAxis("Mouse ScrollWheel") < 0)
00104             {
00105                 selectedHotbarSlot -= 1;
00106                 if (selectedHotbarSlot == 26)
00107                     selectedHotbarSlot = 35;
00108             }
00109             transform.parent.GetComponentInChildren<PlayerInventory>().
00110             SelectedSlot(selectedHotbarSlot);
00111         }
```

5.3.2.6 Update()

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

Breaks and places a Block if an inventory is no open

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

```
00059         {
00060             if (!isAnotherInventoryOpen)
00061             {
00062                 if (GetButtonDown("Break Block"))
00063                     BreakBlock();
00064                 if (GetButtonDown("Place"))
00065                     PlaceBlock();
00066             }
00067         }
```

5.3.2.7 UpdateSelector()

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

Updates teh selectors position

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

```
00075         {
00076             if (Physics.Raycast(transform.position, transform.forward, out hit, 15,
00077                 layers))
00078             {
00079                 selector.SetActive(true);
00079                 selector.transform.position = GetBlockPos(hit);
00080                 /*selector.SetActive(BlockInPosition(GetBlockPos(hit),
00081                 hit.collider.GetComponent<Chunk> ( ) ));
00081             }
00082             else
00083             {
00084                 selector.SetActive(false);
00085             }
00086             SelectedSlot();
00087         }
```

5.3.3 Member Data Documentation

5.3.3.1 hit

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

Where the raycast hit

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

5.3.3.2 layers

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

Layers for the selector to look at

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

5.3.3.3 selectedHotbarSlot

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

What slot in the hotbar is selected

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

5.3.3.4 selector

GameObject `BeeGame.Player.Selector.selector`

Selector

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

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

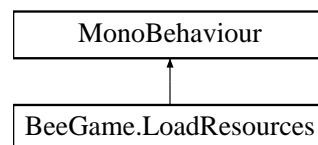
- `C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/Selector.cs`

6 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          SpriteDictionary.LoadSprites();  
00018          PrefabDictionary.LoadPrefabs();  
00019      }
```

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/[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/[SpriteDictionary.cs](#)

6.4 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](#)

6.4.1 Detailed Description

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

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

6.4.2 Constructor & Destructor Documentation

6.4.2.1 Resources()

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

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

```
00033         {  
00034     }
```

6.4.3 Member Function Documentation

6.4.3.1 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]) 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     }
```

6.4.3.2 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]) as Texture2D;
00106                 sprites.Add(splitText[0], Sprite.Create(tex, new UnityEngine.Rect(0, 0, tex.
width, tex.height), Vector2.zero));
00107             }
00108         }
00109
00110         splitText = lineText.Split(splitCharacters, StringSplitOptions.RemoveEmptyEntries);
00111         lineText = "";
00112         tex = UnityEngine.Resources.Load("Sprites/" + splitText[1]) as Texture2D;
00113         sprites.Add(splitText[0], Sprite.Create(tex, new UnityEngine.Rect(0, 0, tex.width,
tex.height), Vector2.zero));
00114
00115         return sprites;
00116     }
```

6.4.4 Member Data Documentation

6.4.4.1 resourceCulture

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

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

6.4.4.2 resourceMan

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

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

6.4.5 Property Documentation

6.4.5.1 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](#).

6.4.5.2 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](#).

6.4.5.3 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](#).

6.4.5.4 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 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 130 of file [THInput.cs](#).

```

00131     {
00132         int returnAxis = 0;
00133
00134         if (axis == "Horizontal")
00135         {
00136             if (GetButton("Right"))
00137             {
00138                 returnAxis += 1;
00139             }
00140
00141             if (GetButton("Left"))
00142             {
00143                 returnAxis -= 1;
00144             }
00145         }
00146         else if (axis == "Vertical")
00147         {
00148             if (GetButton("Forward"))
00149             {
00150                 returnAxis += 1;
00151             }
00152
00153             if (GetButton("Backward"))
00154             {
00155                 returnAxis -= 1;
00156             }
00157         }
00158
00159         return returnAxis;
00160     }

```

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 70 of file [THInput.cs](#).

```

00071     {
00072         if (!inputButtons.ContainsKey(button))
00073         {
00074             throw new Exception("Input Manager: Key button name not defined: " + button);
00075         }
00076
00077         switch (inputButtons[button])
00078         {
00079             case KeyCode[] array:
00080                 /*for each possible key, check if it was pressed and if it was return that it was, if
00081                 none of them was pressed return false
00082                 foreach (var item in array)
00083                 {
00084                     if (Input.GetKey(item))
00085                     {
00086                         return true;
00087                     }
00088                 }
00089                 return false;
00090             default:
00091                 return Input.GetKey((KeyCode)inputButtons[button]);
00092         }
00093     }

```

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 40 of file [THInput.cs](#).

```
00041     {
00042         if (!inputButtons.ContainsKey(button))
00043         {
00044             throw new Exception("Input Manager: Key button name not defined: " + button);
00045         }
00046
00047         switch (inputButtons[button])
00048         {
00049             case KeyCode[] array:
00050                 /*for each possible key, check if it was pressed and if it was return that it was, if
00051                 none of them was pressed return false
00052                 foreach (var item in array)
00053                 {
00054                     if (Input.GetKeyDown(item))
00055                     {
00056                         return true;
00057                     }
00058                 }
00059                 return false;
00060             default:
00061                 return Input.GetKeyDown((KeyCode)inputButtons[button]);
00062         }
00063     }
```

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 100 of file [THInput.cs](#).

```
00101     {
00102         if (!inputButtons.ContainsKey(button))
00103         {
00104             throw new Exception("Input Manager: Key button name not defined: " + button);
00105         }
00106
00107         switch (inputButtons[button])
00108         {
00109             case KeyCode[] array:
00110                 /*for each possible key, check if it was pressed and if it was return that it was, if
00111                 none of them was poressed return false
00112                 foreach (var item in array)
00113                 {
00114                     if (Input.GetKeyUp(item))
00115                     {
00116                         return true;
00117                     }
00118
00119                     return false;
00120                 default:
00121                     return Input.GetKeyUp((KeyCode)inputButtons[button]);
00122             }
00123         }
```

7.1.3 Member Data Documentation

7.1.3.1 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](#).

7.1.3.2 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

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

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

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

7.2.2.2 THVector2() [2/3]

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

Constructor from another [THVector2](#)

Parameters

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

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

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

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

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

<code>vec3</code>	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

<code>vec3</code>	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

<code>a</code>	First Vector
<code>b</code>	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 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         }
```

7.3.3.5 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         }
```

7.3.3.6 operator !=()

```
static bool BeeGame.Core.THVector3.operator!= (  
    THVector3 a,  
    THVector3 b ) [static]
```

Inverse of ==

Parameters

<code>a</code>	First vector
<code>b</code>	Second vector

Returns

true if $a \neq b$

Definition at line 140 of file [THVector3.cs](#).

```

00141      {
00142          return ! (a == b);
00143      }

```

7.3.3.7 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      }

```

7.3.3.8 operator*() [2/3]

```

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

```

Multiplies b to vector a

Parameters

<i>a</i>	Vector a
<i>b</i>	float b

Returns

returns new vector that is the product of a and b

Definition at line 241 of file [THVector3.cs](#).

```

00242     {
00243         a.x *= b;
00244         a.y *= b;
00245         a.z *= b;
00246
00247         return a;
00248     }

```

7.3.3.9 operator*() [3/3]

```

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

```

Multiplies a to vector b

Parameters

<i>a</i>	Vector a
<i>b</i>	float b

Returns

returns new vector that is the product of a and b

Definition at line 255 of file [THVector3.cs](#).

```

00256     {
00257         return new THVector3(a * b.x, a * b.y, a * b.z);
00258     }

```

7.3.3.10 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     }

```

7.3.3.11 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     }

```

7.3.3.12 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      }
```

7.3.3.13 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      }
```

7.3.3.14 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      }
```

7.3.3.15 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      }
```

7.3.3.16 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      }
```

7.3.3.17 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      }
```

7.3.3.18 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         }
```

7.3.3.19 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     }
```

7.3.3.20 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 position

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/THVector3.cs](#)

8 Misc

8.1 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 [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

8.1.1 Detailed Description

Serializes and Deserialises things

Binary serialization is SLOW try to only serialize only what is absolutly necessary

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

8.1.2 Member Function Documentation

8.1.2.1 DeserializeInventory()

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

Deserializesd an [Inventory](#) from its name into a given *inventory*

Parameters

<i>inventory</i>	Inventory to apply the data to
<i>inventoryName</i>	Inventory to deserialize

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

```
00072     {
00073         /* make the path
00074         string inventorySavePath = $"{savePath}/Inventorys/{inventoryName}.dat";
00075
00076         /* checks that the file exists
00077         if (!File.Exists(inventorySavePath))
00078             return;
00079
00080         inventory.SetAllItems((ItemsInInventory) LoadFile($"{inventorySavePath}"
00081     });
00081 }
```

8.1.2.2 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 134 of file [Serialization.cs](#).

```
00135     {
00136         return $"{pos.x}, {pos.y}, {pos.z}";
00137     }
```

8.1.2.3 LoadChunk()

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

Load a Chunk

Parameters

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

Returns

Definition at line 109 of file [Serialization.cs](#).

```
00110      {
00111          /** gets the save file
00112          string saveFile = $"{savePath}/{FileName(chunk.chunkWorldPos)}.dat";
00113
00114          /** if the file does not exist return false
00115          if (!File.Exists(saveFile))
00116              return false;
00117
00118          /** set all of the changed blocks in the chunk
00119          SaveChunk save = (SaveChunk)LoadFile(saveFile);
00120
00121          foreach (var block in save.blocks)
00122          {
00123              chunk.blocks[block.Key.x, block.Key.y, block.Key.z] = block.Value;
00124          }
00125
00126          return true;
00127      }
```

8.1.2.4 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 171 of file [Serialization.cs](#).

```
00172      {
00173          BinaryFormatter bf = new BinaryFormatter();
00174          FileStream fs = new FileStream(file, FileMode.Open);
00175
00176          try
00177          {
00178              return bf.Deserialize(fs);
00179          }
00180          catch (SerializationException e)
00181          {
00182              Debug.Log($"Deserialization Exception {e}");
00183              throw new SerializationException();
00184          }
00185          finally
00186          {
00187              fs.Close();
00188          }
00189      }
```

8.1.2.5 MakeDirectorys()

```
static void BeeGame.Serialization.Serialization.MakeDirectorys ( ) [static]
```

Sets the paths for the save files

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

```
00039     {
00040         savePath = $"{Application.dataPath}/{saveFolderName}/{worldName}";
00041
00042         if (!(Directory.Exists(savePath)))
00043             Directory.CreateDirectory(savePath);
00044     }
```

8.1.2.6 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 89 of file [Serialization.cs](#).

```
00090     {
00091         /* saves the blocks
00092         SaveChunk save = new SaveChunk(chunk.blocks);
00093
00094         /* if no block was changed return early
00095         if (save.blocks.Count == 0)
00096             return;
00097
00098         /* otherwise save the file
00099         string saveFile = $"{savePath}/{FileName(chunk.chunkWorldPos)}.dat";
00100
00101         SaveFile(save, saveFile);
00102     }
```

8.1.2.7 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 146 of file [Serialization.cs](#).

```
00147     {
00148         BinaryFormatter bf = new BinaryFormatter();
00149         FileStream fs = new FileStream(file, FileMode.OpenOrCreate);
00150
00151         try
00152         {
00153             bf.Serialize(fs, obj);
00154         }
00155         catch (SerializationException e)
00156         {
00157             Debug.Log($"Serialization Exception: {e}");
00158             throw new SerializationException();
00159         }
00160         finally
00161         {
00162             fs.Close();
00163         }
00164     }
```

8.1.2.8 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 56 of file [Serialization.cs](#).

```
00057     {
00058         string inventorySavePath = $"{savePath}/Inventorys";
00059
00060         if (!Directory.Exists(inventorySavePath))
00061             Directory.CreateDirectory(inventorySavePath);
00062
00063         SaveFile(inventory.GetAllItems(), $"{inventorySavePath}/{inventoryName}.dat");
00064     }
```

8.1.3 Member Data Documentation

8.1.3.1 saveFolderName

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

Save folder

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

8.1.3.2 savePath

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

Path to save things

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

8.1.3.3 worldName

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

Name if the world. If multiple world are ever added

Definition at line 24 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](#)

8.2 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

8.2.1 Detailed Description

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

8.2.2 Member Function Documentation

8.2.2.1 CloneObject< T >()

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

Allows the copying of a class by value using reflection

Parameters

<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](#).

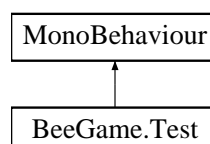
```
00020     {
00021         /**gets the tyoe of the given object
00022         Type typeSource = obj.GetType();
00023
00024         /**makes a new object of type T
00025         T objTarget = (T)Activator.CreateInstance(typeSource);
00026
00027         /**gets the properties in T
00028         PropertyInfo[] 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](#)

8.3 BeeGame.Test Class Reference

Inheritance diagram for BeeGame.Test:



Private Member Functions

- void [Start](#) ()

8.3.1 Detailed Description

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

8.3.2 Member Function Documentation

8.3.2.1 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         {
00077             //only 4 uvs per face
00078             Vector2[] UVs = new Vector2[4];
00079             Tile tilePos = TexturePosition(direction);
00080
00081             //sets the UVs for each vertex
00082             UVs[0] = new THVector2(tileSize * tilePos.x + tileSize - 0.01f, tileSize * tilePos.
00083 y + 0.01f);
00084             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 public override int GetHashCode()
00330 {
00331     return 1;
00332 }
00333
00334 public override bool Equals(object obj)
00335 {

```

```

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/ItemGameObject.cs File Reference

Classes

- class [BeeGame.Items.ItemGameObject](#)
Interface between item and inity gameobjects

Namespaces

- namespace [BeeGame.Items](#)

1.4 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             GetComponent<BoxCollider>().enabled = false;
00040             Instantiate(item.GetGameObject(), transform, false);
00041         }
00042     }
00043
00047 void MakeMesh()
00048 {
00049     MeshData meshData = new MeshData();
00050     if(item != null)
00051         meshData = item.ItemMesh(0, 0, 0, meshData);
00052
00053     Mesh mesh = new Mesh()
00054     {
00055         vertices = meshData.verts.ToArray(),
00056         triangles = meshData.tris.ToArray(),
00057         uv = meshData.uv.ToArray()
00058     };
00059
00060     mesh.RecalculateNormals();
00061
00062     GetComponent<MeshFilter>().mesh = mesh;
00063 }
00064 }
00065 }

```

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
00036         public Block(string name) : base(name)

```

```

00037     {
00038         placeable = true;
00039     }
00040     #endregion
00041
00042     #region Update/Break Block
00043     public virtual void BreakBlock(THVector3 pos)
00044     {
00045         GameObject go = Object.Instantiate(UnityEngine.Resources.Load("
Prefabs/ItemGameObject") as GameObject, pos, Quaternion.identity) as GameObject;
00050         go.GetComponent<ItemGameObject>().item = this;
00051     }
00052
00060     public virtual void UpdateBlock(int x, int y, int z, Chunk chunk) { }
00061     #endregion
00062
00063     #region Mesh
00064     public virtual MeshData BlockData(Chunk chunk, int x, int y, int z,
MeshData meshData, bool addToRenderMesh = true)
00079     {
00080         //Adds the Top face of the block
00081         if (!chunk.GetBlock(x, y + 1, z, false).IsSolid(Direction.DOWN))
00082         {
00083             meshData = FaceDataUp(x, y, z, meshData, addToRenderMesh);
00084         }
00085
00086         //Adds the Bottom face of the block
00087         if (!chunk.GetBlock(x, y - 1, z, false).IsSolid(Direction.UP))
00088         {
00089             meshData = FaceDataDown(x, y, z, meshData, addToRenderMesh);
00090         }
00091
00092         //Adds the North face of the block
00093         if (!chunk.GetBlock(x, y, z + 1, false).IsSolid(Direction.SOUTH))
00094         {
00095             meshData = FaceDataNorth(x, y, z, meshData, addToRenderMesh);
00096         }
00097
00098         //Adds the South face of the block
00099         if (!chunk.GetBlock(x, y, z - 1, false).IsSolid(Direction.NORTH))
00100         {
00101             meshData = FaceDataSouth(x, y, z, meshData, addToRenderMesh);
00102         }
00103
00104         //Adds the East face of the block
00105         if (!chunk.GetBlock(x + 1, y, z, false).IsSolid(Direction.WEST))
00106         {
00107             meshData = FaceDataEast(x, y, z, meshData, addToRenderMesh);
00108         }
00109
00110         //Adds the West face of the block
00111         if (!chunk.GetBlock(x - 1, y, z, false).IsSolid(Direction.EAST))
00112         {
00113             meshData = FaceDataWest(x, y, z, meshData, addToRenderMesh);
00114         }
00115
00116         return meshData;
00117     }
00118
00119     public virtual bool IsSolid(Direction direction)
00125     {
00126     {
00127         return true;
00128     }
00129     #endregion
00130
00131     #region Overrides
00132     public override int GetHashCode()
00137     {
00138         return 1;
00139     }
00140
00145     public override string ToString()
00146     {
00147         return $"{itemName} \nID: {GetHashCode()}";
00148     }
00149     #endregion
00150 }
00151 }

```

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 addRoRenderMesh = 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/Bedrock.cs File Reference

Classes

- class [BeeGame.Blocks.Bedrock](#)
Bedrock Block

Namespaces

- namespace [BeeGame.Blocks](#)

2.6 Bedrock.cs

```
00001 using System;
00002 using BeeGame.Core.Enums;
00003 using BeeGame.Items;
00004 using BeeGame.Core;
00005
00006 namespace BeeGame.Blocks
00007 {
00011     [Serializable]
00012     public class Bedrock : Block
00013     {
00014         #region Constructor
00015         public Bedrock() : base("Bedrock")
00016         {
00019             breakable = false;
00020         }
00021         #endregion
00022
00023         #region Break Block
00024         public override void BreakBlock(THVector3 pos)
00025         {
00030             return;
00031         }
00032         #endregion
00033
00034         #region Mesh
00035         public override Tile TexturePosition(Direction direction)
00036         {
00042             return new Tile() { x = 0, y = 0 };
00043         }
00044         #endregion
00045
00046         #region Overrides
00047         public override int GetHashCode()
00048         {
00053             return -1;
00054         }
00055
00056         public override string ToString()
00057         {
00062             return $"{itemName} \nID: {GetItemID()}";
00063         }
00064         #endregion
00065     }
00066 }
00067 }
```

2.7 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](#)

2.8 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")
00012         {
00015         }
00016     }
00017 }
```

```

00016     }
00017     #endregion
00018
00019     public Apiary(SerializationInfo info, StreamingContext context)
00020     {
00021         /*use info.getvalue("valuenam", typeof(valueType))
00022         UnityEngine.MonoBehaviour.print("hi");
00023     }
00024
00025     #region Overrides
00026     public override int GetHashCode()
00027     {
00028         return 3;
00029     }
00030
00031     public override string ToString()
00032     {
00033         return $"{itemName} \nID: {GetItemID()}";
00034     }
00035     #endregion
00036 }
00037
00038 }
00039
00040 }
00041
00042 }
00043
00044 }
00045

```

2.9 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](#)

2.10 Dirt.cs

```

00001 using System;
00002 using BeeGame.Core.Enums;
00003 using BeeGame.Items;
00004
00005 namespace BeeGame.Blocks
00006 {
00007     [Serializable]
00008     public class Dirt : Block
00009     {
00010         #region Constructor
00011         public Dirt() : base("Dirt"){}
00012         #endregion
00013
00014         #region Mesh
00015         public override Tile TexturePosition(Direction direction)
00016         {
00017             return new Tile { x = 2, y = 9 };
00018         }
00019         #endregion
00020
00021         #region Overrides
00022         public override int GetHashCode()
00023         {
00024             return 5;
00025         }
00026
00027         public override string ToString()
00028         {
00029             return $"{itemName} \nID: {GetItemID()}";
00030         }
00031         #endregion
00032     }
00033 }
00034
00035 }
00036
00037 }
00038
00039 }
00040
00041 }
00042
00043 }
00044
00045 }
00046
00047 }
00048
00049 }
00050
00051 }
00052

```

2.11 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](#)

2.12 Grass.cs

```
00001 using System;
00002 using BeeGame.Core.Enums;
00003 using BeeGame.Terrain.Chunks;
00004 using BeeGame.Items;
00005
00006 namespace BeeGame.Blocks
00007 {
00011     [Serializable]
00012     public class Grass : Block
00013     {
00014         #region Constructor
00015         public Grass() : base("Grass"){ }
00019         #endregion
00021         #region Mesh
00022         public override void UpdateBlock(int x, int y, int z, Chunk chunk)
00030         {
00031             if (chunk.GetBlock(x, y + 1, z, false).IsSolid(Direction.DOWN))
00032                 chunk.blocks[x, y, z] = new Dirt() { changed = changed };
00033         }
00034
00040         public override Tile TexturePosition(Direction direction)
00041         {
00042             //All textures are on the same Y value for the texture atlas so Y can be set
00043             Tile tile = new Tile()
00044             {
00045                 y = 9
00046             };
00047
00048             switch (direction)
00049             {
00050                 //if we want the top face return the full grass texture
00051                 case Direction.UP:
00052                     tile.x = 3;
00053                     return tile;
00054                 //if we want the bottom face return the dirt texture
00055                 case Direction.DOWN:
00056                     tile.x = 2;
00057                     return tile;
00058                 //return the 1/2 grass texture if a side face is wanted
00059                 default:
00060                     tile.x = 4;
00061                     return tile;
00062             }
00063         }
00064         #endregion
00066         #region Overrides
00067         public override string GetItemName()
00068         {
00069             return "Grass";
00070         }
00071
00076         public override int GetHashCode()
00077         {
00078             return 4;
00079         }
00080
00085         public override string ToString()
00086         {
00087             return $"{itemName} \nID: {GetItemID()}";
00088         }
00089         #endregion
00090     }
00091 }
```

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         private ItemsInInventory items;
00019         public InventorySlot[] slots;
00023         internal Item floatingItem;
00027         public string inventoryName = "";
00028         #endregion
00029
00030         #region Init
00031         public bool InventorySet()
00036         {
00037             if (items == null)
00038                 return true;
00039
00040             return false;
00041         }
00042
00047         public void SetInventorySize(int inventorySize)
00048         {
00049             items = new ItemsInInventory(slots.Length);
00050         }
00051
00059         public void SetAllItems(ItemsInInventory items)
00060         {
00061             this.items = items;
00062         }
00063         #endregion
00064
00065         #region Update
00066         public void UpdateBase()
00070         {
00071             PutItemsInSlots();
00072         }
00073         #endregion
00074
00075         #region Edit Inventory
00076         void PutItemsInSlots()
00080         {
00081             /* goes through all of the items in the array setting them all to a slot
00082             for (int i = 0; i < slots.Length; i++)
00083             {
00084                 slots[i].slotIndex = i;
00085                 slots[i].myInventory = this;
00086                 slots[i].item = items.itemsInInventory[i];
00087             }
00088         }
00089
00094         public ItemsInInventory GetAllItems()
00095         {
00096             return items;
```

```

00097     }
00098
00104     public void AddItemToSlots(int slotIndex, Item item)
00105     {
00106         items.AddItem(slotIndex, item);
00107         /* saves the inventory changes
00108         Serialization.Serialization.SerializeInventory(this, inventoryName);
00109     }
00110
00116     public bool AddItemToInventory(Item item)
00117     {
00118         return items.AddItem(item);
00119     }
00120     #endregion
00121 }
00122 }

```

3.3 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/ItemsInInventory.cs File Reference

Classes

- class [BeeGame.Inventory.ItemsInInventory](#)
Class that holds all of the items in the inventory. Can be serialized so inventory may be saved

Namespaces

- namespace [BeeGame.Inventory](#)

3.4 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 }

```


3.5 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/InventorySlot.cs File Reference

Classes

- class [BeeGame.Inventory.InventorySlot](#)

Namespaces

- namespace [BeeGame.Inventory](#)

3.6 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,
00010         IPointerExitHandler
00011     {
00012         #region Data
00013         internal int slotIndex;
00014         public Item item;
00015         public Inventory myInventory;
00016         public GameObject itemText;
00017         public bool selectedSlot = false;
00018         #endregion
00019
00020         private void Update()
00021         {
00022             UpdateIcon();
00023         }
00024
00025         void UpdateIcon()
00026         {
00027             if(item == null)
00028             {
00029                 GetComponent<Image>().sprite = null;
00030             }
00031             else
00032             {
00033                 GetComponent<Image>().sprite = item.GetItemSprite();
00034             }
00035
00036             /* if the slot is selected in the hotbar give the player some indication by colouring it grey
00037             if (selectedSlot)
00038             {
00039                 GetComponent<Image>().color = Color.gray;
00040             }
00041             else
00042             {
00043                 GetComponent<Image>().color = Color.white;
00044             }
00045         }
00046
00047         #region Interact With Slot
00048         public void OnPointerClick(PointerEventData eventData)
00049         {
00050             if (myInventory.floatingItem != null)
00051             {
00052                 /* Left click moves whole stacks if items
00053                 if (eventData.button == PointerEventData.InputButton.Left)
00054                 {
00055                     /* If the item in the slot is empty put the floating item into it then clear it
00056                     if (item == null)
00057                     {
00058                         item = myInventory.floatingItem;
00059                         myInventory.floatingItem = null;
00060                         myInventory.AddItemToSlots(slotIndex, item);
00061                         return;
00062                     }
00063                 }
00064             }
00065         }
```

```

00090             /** if the items are the same
00091             if(myInventory.floatingItem == item)
00092             {
00093                 /** if the item in the inventoys stack count + the floating items stack count is
00094                 less than the max stack count
00094                 if (myInventory.floatingItem.itemStackCount + item.
itemStackCount <= item.maxStackCount)
00095                 {
00096                     AddToSlot(myInventory.floatingItem.
itemStackCount);
00097                     return;
00098                 }
00099                 /** if the item stack added is larger than the max count add as many as you can and
00100                 move on
00100                 else
00101                 {
00102                     AddToSlot(item.maxStackCount - item.
itemStackCount);
00103                     return;
00104                 }
00105             }
00106             /** If the items were not == swap them
00107             else
00108             {
00109                 SwapItems();
00110                 return;
00111             }
00112         }
00113         else if(eventData.button == PointerEventData.InputButton.Right)
00114         {
00115             /** if the item in slot is null add 1 from the floating item to it
00116             if(item == null)
00117             {
00118                 AddToSlot(1);
00119                 return;
00120             }
00121             /** if the items are the same add 1 from the floating item to this item
00122             else if(item == myInventory.floatingItem)
00123             {
00124                 AddToSlot(1);
00125                 return;
00126             }
00127         }
00128     }
00129     /** if the floating item is null
00130     else
00131     {
00132         /** add 1/2 of the stack into the floating item if right click was pressed
00133         if(eventData.button == PointerEventData.InputButton.Right)
00134         {
00135             SplitStack();
00136             return;
00137         }
00138     }
00139     /** otherwie add the items into the floating item slot
00140     SwapItems();
00141     return;
00142 }
00143 }
00144 }
00145
00150 void AddToSlot(int numerToAdd)
00151 {
00152     /** if the item in the slot is null create it
00153     if (item == null)
00154     {
00155         item = myInventory.floatingItem.CloneObject();
00156         item.itemStackCount = 0;
00157     }
00158
00159     /** add to number to add to the stack count
00160     item.itemStackCount += numerToAdd;
00161
00162     /** if the stack count is now larger than it should be dont let it be
00163     if (item.itemStackCount > item.maxStackCount)
00164     {
00165         item.itemStackCount = item.maxStackCount;
00166     }
00167
00168     /** remove the numebr if items form the floating item then check the floating item is not null
00169     myInventory.floatingItem.itemStackCount -= numerToAdd;
00170     CheckFloatingItem();
00171     /** save the inventory changes
00172     myInventory.AddItemToSlots(slotIndex, item);
00173 }
00174
00181 void SplitStack()

```

```

00182     {
00183         myInventory.floatingItem = item.CloneObject();
00184         int give = (item.itemStackCount + 1) / 2;
00185         myInventory.floatingItem.itemStackCount = give;
00186         item.itemStackCount -= give;
00187
00188         if (item.itemStackCount <= 0)
00189             item = null;
00190
00191         myInventory.AddItemToSlots(slotIndex, item);
00192         Destroy(itemText);
00193     }
00194
00198     void SwapItems()
00199     {
00200         /** temp copy of the item
00201         Item temp = myInventory.floatingItem;
00202         /** sets the floating item
00203         myInventory.floatingItem = item;
00204         /** sets the item that was in the floating item to the item in the the slot
00205         item = temp;
00206         /** Saves the changes to the inventory
00207         myInventory.AddItemToSlots(slotIndex, item);
00208         /** destroys the text as it is not needed anymore
00209         Destroy(itemText);
00210     }
00211
00215     void CheckFloatingItem()
00216     {
00217         if(myInventory.floatingItem.itemStackCount <= 0)
00218         {
00219             myInventory.floatingItem = null;
00220         }
00221     }
00222     #endregion
00223
00224     #region Display Item On Hover
00225     public void OnPointerEnter(PointerEventData eventData)
00226     {
00231         /** if the item is null or the floating item has something in it dont display the item text as
00232         it is not neccessary
00233         if (item != null && myInventory.floatingItem == null)
00234         {
00235             itemText = Instantiate(PrefabDictionary.
00236             GetPrefab("ItemDetails"));
00237             /** sets the text to the correct postion
00238             itemText.transform.GetChild(0).position = Input.mousePosition;
00239             /** puts the correct text in the box
00240             itemText.transform.GetChild(0).GetChild(0).GetComponent<Text>().text = $"
00241             {item.GetItemName()}\nStack: {item.itemStackCount}";
00242         }
00243     }
00244
00246     public void OnPointerExit(PointerEventData eventData)
00247     {
00248         Destroy(itemText);
00249     }
00250
00254     void OnDisable()
00255     {
00256         Destroy(itemText);
00257     }
00258     #endregion
00259 }
00260 }

```

3.7 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](#)

3.8 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 Start()
00024         {
00025             SetPlayerInventory();
00026             inventoryName = "PlayerInventory";
00027             Serialization.Serialization.DeSerializeInventory(this, inventoryName);
00028         }
00029
00033         void SetPlayerInventory()
00034         {
00035             if (InventorySet())
00036                 SetInventorySize(20);
00037         }
00038         #endregion
00039
00043         void Update()
00044         {
00045             UpdateBase();
00046
00047             /* checks if the inventory should be opened/closed
00048             if (THInput.GetButtonDown("Player Inventory"))
00049                 OpenPlayerInventory();
00050
00051             /* checks if something should be picked up and put into the inventory
00052             RaycastHit[] hit = Physics.SphereCastAll(transform.position, 1f, transform.forward);
00053
00054             for (int i = hit.Length - 1; i >= 0; i--)
00055             {
00056                 if (hit[i].collider.GetComponent<ItemGameObject>())
00057                     PickupItem(hit[i].collider.GetComponent<ItemGameObject>());
00058             }
00059
00060         }
00061
00062         #region Hotbar
00063         public void SelectedSlot(int index)
00068         {
00069             for (int i = 0; i < slots.Length; i++)
00070             {
00071                 slots[i].selectedSlot = false;
00072             }
00073
00074             slots[index].selectedSlot = true;
00075         }
00076
00083         public bool GetItemFromHotBar(int slotIndex, out Item outItem)
00084         {
00085             /* get the item
00086             outItem = GetAllItems().itemsInInventory[slotIndex];
00087
00088             if (outItem == null)
00089                 return false;
00090
00091             /* 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
00092             if(outItem.placeable)
00093                 RemoveItemFromInventory(slotIndex);
00094
00095             return outItem.placeable;
00096         }
00097         #endregion
00098
00099         #region Interact With Inventory
00100         void OpenPlayerInventory()
00104         {
00105             playerInventory.SetActive(!playerInventory.activeInHierarchy);
00106             THInput.isAnotherInventoryOpen = !
THInput.isAnotherInventoryOpen;
00107
00108             /* hides/ shows the mouse depending on if the inventory is open or not
00109             if (playerInventory.activeInHierarchy)
00110             {
```

```

00111         Cursor.lockState = CursorLockMode.None;
00112         Cursor.visible = true;
00113     }
00114     else
00115     {
00116         Cursor.visible = false;
00117         Cursor.lockState = CursorLockMode.Locked;
00118     }
00119 }
00120
00121 public void RemoveItemFromInventory(int index)
00122 {
00123     /** if the item is already null nothign needs to be removed
00124     if (GetAllItems().itemsInInventory[index] != null)
00125     {
00126         /** remove 1 item and if that was the last in the stack remove the item from the inventory
00127         GetAllItems().itemsInInventory[index].itemStackCount -= 1;
00128
00129         if (GetAllItems().itemsInInventory[index].itemStackCount <= 0)
00130             GetAllItems().itemsInInventory[index] = null;
00131
00132         Serialization.Serialization.SerializeInventory(this, inventoryName);
00133     }
00134 }
00135
00136 void PickupItem(ItemGameObject item)
00137 {
00138     /** if the item can be added to the inventory do that
00139     if (AddItemToInventory(item.item))
00140     {
00141         /** if the item was added destroy its gameobject and save the inventory
00142         Destroy(item.gameObject);
00143         Serialization.Serialization.SerializeInventory(this, inventoryName);
00144     }
00145 }
00146 #endregion
00147 }
00148 }

```

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()
00075     {
00079         filter = GetComponent<MeshFilter>();
00080         meshCollider = GetComponent<MeshCollider>();
00081     }
00082
00086     void Update()
00087     {
00088         lock(mesh)
00089         {
00090             if (update)
00091             {
00092                 update = false;
00093                 updateCollisionMesh = true;
00094                 mesh = new MeshData();
00095                 //Enabling threading here works in editor but not in build?
00096                 //ok whatever...
00097                 //Thread thread = new Thread(UpdateChunk);
00098
00099                 //thread.Start();
00100                 UpdateChunk();
00101             }
00102
00103             if (mesh.done && mesh != new MeshData())
00104             {
00105                 RenderMesh(mesh);
00106             }
00107
00108             if (applyCollisionMesh)
00109                 ColliderMesh();
00110         }
00111     }
00112     #endregion
00113
00114     #region Get/Set Blocks
00115     public Block GetBlock(int x, int y, int z, bool checkNeighbouringChunks = true)
00116     {
00125         //checks that block is in the chunk
00126         if (InRange(x) && InRange(y) && InRange(z))
00127             return blocks[x, y, z];
00128
00129         //if the block is not in the chunk and we should check other chunks do that, otherwise return
00130         an air block (empty block)
00131         if(checkNeighbouringChunks)
00132             return world.GetBlock(chunkWorldPos.x + x, chunkWorldPos.
00133 y + y, chunkWorldPos.z + z);
00134
00135         return new Air();
00136     }
00137
00143     public void SetBlock(int x, int y, int z, Block block)
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         //if the block is not in the chunk find its chunk and set it their
00152         world.SetBlock(chunkWorldPos.x + x, chunkWorldPos.y + y, chunkWorldPos.
00153 z + z, block);
00154     }
00155
00160     public static bool InRange(int i)
00161     {
00162         //if the value is less then 0 or greater than 16 the value is outside the chunk
00163         if (i < 0 || i >= chunkSize)

```

```

00164         return false;
00165     return true;
00166 }
00167 #endregion
00168
00169 #region Mesh
00170 public void SetBlocksUnmodified()
00171 {
00172     foreach (var block in blocks)
00173     {
00174         block.changed = false;
00175     }
00176 }
00177
00187 void UpdateChunk()
00188 {
00189     //says that this chunk is rendered and initialtes the mesh
00190     rendered = true;
00191
00192     //goes through every block in the blocks array getting their mesh data
00193     for (int x = 0; x < chunkSize; x++)
00194     {
00195         for (int z = 0; z < chunkSize; z++)
00196         {
00197             for (int y = 0; y < chunkSize; y++)
00198             {
00199                 blocks[x, y, z].UpdateBlock(x, y, z, this);
00200                 mesh = blocks[x, y, z].BlockData(this, x, y, z, mesh);
00201             }
00202         }
00203     }
00204     mesh.done = true;
00205 }
00206
00211 void RenderMesh(MeshData meshData)
00212 {
00213     //Applying the mesh takes the longest but nothing can be dont with the mesh class in a
secondary thread...thanks unity
00214
00215     mesh.done = false;
00216     //clears the current chunk mesh
00217     filter.mesh.Clear();
00218     //name for convenience
00219     filter.mesh.name = "Render Mesh";
00220     //puts the tris and verts from the meshdata into the chunk mesh
00221     filter.mesh.vertices = meshData.verts.ToArray();
00222     filter.mesh.triangles = meshData.tris.ToArray();
00223
00224     //sets the uvs
00225     filter.mesh.uv = meshData.uv.ToArray();
00226
00227     //redoes the normals incase they got messed up
00228     filter.mesh.RecalculateNormals();
00229 }
00230
00234 void ColliderMesh()
00235 {
00236     //if the chunk has been told to update the collsions but the chunk has ne verts dont do it as
their is no point
00237     if (this.mesh.verts.Count == 0)
00238     {
00239         return;
00240     }
00241     //if the render and collision meshes should be shared set the render mesh to the collision mesh
otherwise make a collision mesh
00242     if (this.mesh.shareMeshes)
00243     {
00244         world.chunkHasMadeCollisionMesh = true;
00245         applyCollisionMesh = false;
00246         meshCollider.sharedMesh = filter.mesh;
00247         return;
00248     }
00249     world.chunkHasMadeCollisionMesh = true;
00250     //Applying the mesh takes the longest but nothing can be dont with the mesh class in a
secondary thread...thanks Unity
00251
00252     //makes a new mesh setting the name for convenience
00253     Mesh mesh = new Mesh()
00254     {
00255         name = "Collider Mesh",
00256         vertices = this.mesh.colVerts.ToArray(),
00257         triangles = this.mesh.colTris.ToArray()
00258     };
00259
00260     //recalcs the normals and applies the mesh
00261     mesh.RecalculateNormals();
00262

```

```

00263         meshCollider.sharedMesh = mesh;
00264
00265         applyCollisionMesh = false;
00266     }
00267     #endregion
00268 }
00269 }

```

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/Chunks/LoadChunks.cs File Reference

Classes

- class [BeeGame.Terrain.Chunks.LoadChunks](#)
Loads the [Chunks](#) around the player

Namespaces

- namespace [BeeGame.Terrain.Chunks](#)

4.6 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     {
00016         public World world;
00017
00021         private List<ChunkWorldPos> buildList = new List<ChunkWorldPos>();
00022
00026         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),
00027             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),
00028             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),
00029             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),
00030             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),
00031             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),
00032             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),
00033             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),
00034             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),
00035             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),
00036             new ChunkWorldPos(-4, 0, 1), new
ChunkWorldPos(-1, 0, -4), new ChunkWorldPos(-1, 0, 4), new

```

```

00037     ChunkWorldPos( 1, 0, -4), new ChunkWorldPos( 1, 0, 4),
        new ChunkWorldPos( 4, 0, -1), new
00038     ChunkWorldPos( 4, 0, 1), new ChunkWorldPos(-3, 0, -3), new
        ChunkWorldPos(-3, 0, 3), new ChunkWorldPos( 3, 0, -3), new
        new ChunkWorldPos( 3, 0, 3), new
00039     ChunkWorldPos(-4, 0, -2), new ChunkWorldPos(-4, 0, 2), new
        ChunkWorldPos(-2, 0, -4), new ChunkWorldPos(-2, 0, 4), new
        new ChunkWorldPos( 2, 0, -4), new
00040     ChunkWorldPos( 2, 0, 4), new ChunkWorldPos( 4, 0, -2), new
        ChunkWorldPos( 4, 0, 2), new ChunkWorldPos(-5, 0, 0), new
        new ChunkWorldPos(-4, 0, -3), new
00041     ChunkWorldPos(-4, 0, 3), new ChunkWorldPos(-3, 0, -4), new
        ChunkWorldPos(-3, 0, 4), new ChunkWorldPos( 0, 0, -5), new
        new ChunkWorldPos( 0, 0, 5), new
00042     ChunkWorldPos( 3, 0, -4), new ChunkWorldPos( 3, 0, 4), new
        ChunkWorldPos( 4, 0, -3), new ChunkWorldPos( 4, 0, 3), new
        new ChunkWorldPos( 5, 0, 0), new
00043     ChunkWorldPos(-5, 0, -1), new ChunkWorldPos(-5, 0, 1), new
        ChunkWorldPos(-1, 0, -5), new ChunkWorldPos(-1, 0, 5), new
        new ChunkWorldPos( 1, 0, -5), new
00044     ChunkWorldPos( 1, 0, 5), new ChunkWorldPos( 5, 0, -1), new
        ChunkWorldPos( 5, 0, 1), new ChunkWorldPos(-5, 0, -2), new
        new ChunkWorldPos(-5, 0, 2), new
00045     ChunkWorldPos(-2, 0, -5), new ChunkWorldPos(-2, 0, 5), new
        ChunkWorldPos( 2, 0, -5), new ChunkWorldPos( 2, 0, 5), new
        new ChunkWorldPos( 5, 0, -2), new
00046     ChunkWorldPos( 5, 0, 2), new ChunkWorldPos(-4, 0, -4), new
        ChunkWorldPos(-4, 0, 4), new ChunkWorldPos( 4, 0, -4), new
        new ChunkWorldPos( 4, 0, 4), new
00047     ChunkWorldPos(-5, 0, -3), new ChunkWorldPos(-5, 0, 3), new
        ChunkWorldPos(-3, 0, -5), new ChunkWorldPos(-3, 0, 5), new
        new ChunkWorldPos( 3, 0, -5), new
00048     ChunkWorldPos( 3, 0, 5), new ChunkWorldPos( 5, 0, -3), new
        ChunkWorldPos( 5, 0, 3), new ChunkWorldPos(-6, 0, 0), new
        new ChunkWorldPos( 0, 0, -6), new
00049     ChunkWorldPos( 0, 0, 6), new ChunkWorldPos( 6, 0, 0), new
        ChunkWorldPos(-6, 0, -1), new ChunkWorldPos(-6, 0, 1), new
        new ChunkWorldPos(-1, 0, -6), new
00050     ChunkWorldPos(-1, 0, 6), new ChunkWorldPos( 1, 0, -6), new
        ChunkWorldPos( 1, 0, 6), new ChunkWorldPos( 6, 0, -1), new
        new ChunkWorldPos( 6, 0, 1), new
00051     ChunkWorldPos(-6, 0, -2), new ChunkWorldPos(-6, 0, 2), new
        ChunkWorldPos(-2, 0, -6), new ChunkWorldPos(-2, 0, 6), new
        new ChunkWorldPos( 2, 0, -6), new
00052     ChunkWorldPos( 2, 0, 6), new ChunkWorldPos( 6, 0, -2), new
        ChunkWorldPos( 6, 0, 2), new ChunkWorldPos(-5, 0, -4), new
        new ChunkWorldPos(-5, 0, 4), new
00053     ChunkWorldPos(-4, 0, -5), new ChunkWorldPos(-4, 0, 5), new
        ChunkWorldPos( 4, 0, -5), new ChunkWorldPos( 4, 0, 5), new
        new ChunkWorldPos( 5, 0, -4), new
00054     ChunkWorldPos( 5, 0, 4), new ChunkWorldPos(-6, 0, -3), new
        ChunkWorldPos(-6, 0, 3), new ChunkWorldPos(-3, 0, -6), new
        new ChunkWorldPos(-3, 0, 6), new
00055     ChunkWorldPos( 3, 0, -6), new ChunkWorldPos( 3, 0, 6), new
        ChunkWorldPos( 6, 0, -3), new ChunkWorldPos( 6, 0, 3), new
        new ChunkWorldPos(-7, 0, 0), new
00056     ChunkWorldPos( 0, 0, -7), new ChunkWorldPos( 0, 0, 7), new
        ChunkWorldPos( 7, 0, 0), new ChunkWorldPos(-7, 0, -1), new
        new ChunkWorldPos(-7, 0, 1), new
00057     ChunkWorldPos(-5, 0, -5), new ChunkWorldPos(-5, 0, 5), new
        ChunkWorldPos(-1, 0, -7), new ChunkWorldPos(-1, 0, 7), new
        new ChunkWorldPos( 1, 0, -7), new
00058     ChunkWorldPos( 1, 0, 7), new ChunkWorldPos( 5, 0, -5), new
        ChunkWorldPos( 5, 0, 5), new ChunkWorldPos( 7, 0, -1), new
        new ChunkWorldPos( 7, 0, 1), new
00059     ChunkWorldPos(-6, 0, -4), new ChunkWorldPos(-6, 0, 4), new
        ChunkWorldPos(-4, 0, -6), new ChunkWorldPos(-4, 0, 6), new
        new ChunkWorldPos( 4, 0, -6), new
00060     ChunkWorldPos( 4, 0, 6), new ChunkWorldPos( 6, 0, -4), new
        ChunkWorldPos( 6, 0, 4), new ChunkWorldPos(-7, 0, -2), new
        new ChunkWorldPos(-7, 0, 2), new
00061     ChunkWorldPos(-2, 0, -7), new ChunkWorldPos(-2, 0, 7), new
        ChunkWorldPos( 2, 0, -7), new ChunkWorldPos( 2, 0, 7), new
        new ChunkWorldPos( 7, 0, -2), new
00062     ChunkWorldPos( 7, 0, 2), new ChunkWorldPos(-7, 0, -3), new
        ChunkWorldPos(-7, 0, 3), new ChunkWorldPos(-3, 0, -7), new
        new ChunkWorldPos(-3, 0, 7), new
00063     ChunkWorldPos( 3, 0, -7), new ChunkWorldPos( 3, 0, 7), new
        ChunkWorldPos( 7, 0, -3), new ChunkWorldPos( 7, 0, 3), new
        new ChunkWorldPos(-6, 0, -5), new
00064     ChunkWorldPos(-6, 0, 5), new ChunkWorldPos(-5, 0, -6), new
        ChunkWorldPos(-5, 0, 6), new ChunkWorldPos( 5, 0, -6), new
        new ChunkWorldPos( 5, 0, 6), new
00065     ChunkWorldPos( 6, 0, -5), new ChunkWorldPos( 6, 0, 5) };
00069     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),
00070
        new
        ChunkWorldPos(1, 0, 1), new ChunkWorldPos(1, 0, -1), new
        ChunkWorldPos(-1, 0, 1), new ChunkWorldPos(-1, 0, -1));
00071
00075     private static int timer = 0;
00076
00080     private void Start()
00081     {
00082         LandGeneration.Terrain.world = world;
00083     }
00084
00088     void Update()
00089     {
00090         if (DeleteChunks())
00091             return;
00092         if (!world.chunkHasMadeCollisionMesh)
00093         {
00094             FindChunksToLoad();
00095             LoadAndRenderChunks();
00096             ApplyCollisionMeshToNearbyChunks();
00097         }
00098         //stops chunks being made and collision meshes being made at the same time
00099         world.chunkHasMadeCollisionMesh = false;
00100     }
00101
00109     void ApplyCollisionMeshToNearbyChunks()
00110     {
00111         //gets the player position in chunk coordinates
00112         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);
00113
00114         for (int i = 0; i < nearbyChunks.Length; i++)
00115         {
00116             ChunkWorldPos chunkPos = new ChunkWorldPos(nearbyChunks[i].x *
        Chunk.chunkSize + playerPos.x, 0, nearbyChunks[i].z * Chunk.
        chunkSize + playerPos.z);
00117
00118             for (int j = -1; j < 2; j++)
00119             {
00120                 Chunk nearbyChunk = world.GetChunk(chunkPos.x, j *
        Chunk.chunkSize, chunkPos.z);
00121
00122                 if (nearbyChunk != null)
00123                     nearbyChunk.applyCollisionMesh = true;
00124             }
00125         }
00126
00127
00131     void LoadAndRenderChunks()
00132     {
00133         //if there is somethign in the build list new chunks can be made
00134         if (buildList.Count != 0)
00135         {
00136             //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
00137             for (int i = buildList.Count - 1, j = 0; i >= 0 && j < 8; i--, j++)
00138             {
00139                 BuildChunk(buildList[0]);
00140                 buildList.RemoveAt(0);
00141             }
00142         }
00143
00144
00148     void FindChunksToLoad()
00149     {
00150         if (buildList.Count == 0)
00151         {
00152             //gets the player position in chunk coordinates
00153             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);
00154
00155             //check all of the chunk positions and if that position does not have a chunk in it make it
00156             for (int i = 0; i < chunkPositions.Length; i++)
00157             {
00158                 ChunkWorldPos newChunkPos = new ChunkWorldPos(chunkPositions[
        i].x * Chunk.chunkSize + playerPos.x, 0, chunkPositions[i].z *
        Chunk.chunkSize + playerPos.z);
00159
00160                 Chunk newChunk = world.GetChunk(newChunkPos.x, newChunkPos.
        y, newChunkPos.z);
00161

```

```

00162         if (newChunk != null && (newChunk.rendered || buildList.Contains(newChunkPos)))
00163             continue;
00164
00165         for (int y = -1; y < 2; y++)
00166         {
00167             for (int x = newChunkPos.x - Chunk.chunkSize; x < newChunkPos.
x + Chunk.chunkSize; x += Chunk.chunkSize)
00168             {
00169                 for (int z = newChunkPos.z - Chunk.chunkSize; z < newChunkPos.
z + Chunk.chunkSize; z += Chunk.chunkSize)
00170                 {
00171                     buildList.Add(new ChunkWorldPos(x, y *
Chunk.chunkSize, z));
00172                 }
00173             }
00174         }
00175         return;
00176     }
00177 }
00178 }
00179
00184 void BuildChunk(ChunkWorldPos pos)
00185 {
00186     if (world.GetChunk(pos.x, pos.y, pos.z) == null)
00187         world.CreateChunk(pos.x, pos.y, pos.z);
00188 }
00189
00194 bool DeleteChunks()
00195 {
00196     //destroys every 10 call to reduce load on CPU so that chunks are not destroyed and created at
the same time
00197     if(timer == 10)
00198     {
00199         timer = 0;
00200         var chunksToDelete = new List<ChunkWorldPos>();
00201
00202         //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
00203         foreach (var chunk in world.chunks)
00204         {
00205             float distance = Vector3.Distance(chunk.Value.transform.position, transform.position);
00206
00207             if (distance > 256)
00208                 chunksToDelete.Add(chunk.Key);
00209         }
00210
00211         foreach (var chunk in chunksToDelete)
00212         {
00213             world.DestroyChunk(chunk.x, chunk.y, chunk.z);
00214         }
00215
00216         return true;
00217     }
00218
00219     timer++;
00220
00221     return false;
00222 }
00223 }
00224 }

```

4.7 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/SaveChunk.cs File Reference

Classes

- class [BeeGame.Terrain.Chunks.SaveChunk](#)
Saves a *Chunks* modified *Blocks* for save optimisation

Namespaces

- namespace [BeeGame.Terrain.Chunks](#)

4.8 SaveChunk.cs

```
00001 using System;
00002 using System.Collections.Generic;
00003 using BeeGame.Blocks;
00004
00005
00006 namespace BeeGame.Terrain.Chunks
00007 {
00011     [Serializable]
00012     public class SaveChunk
00013     {
00017         public Dictionary<ChunkWorldPos, Block> blocks = new Dictionary<ChunkWorldPos, Block>();
00018
00023         public SaveChunk(Block[, ,] blockArray)
00024         {
00025             for (int x = 0; x < Chunk.chunkSize; x++)
00026             {
00027                 for (int y = 0; y < Chunk.chunkSize; y++)
00028                 {
00029                     for (int z = 0; z < Chunk.chunkSize; z++)
00030                     {
00031                         /*if the block has changed save it
00032                         if (blockArray[x, y, z].changed)
00033                             blocks.Add(new ChunkWorldPos(x, y, z), blockArray[x, y, z]);
00034                     }
00035                 }
00036             }
00037         }
00038     }
00039 }
```

4.9 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](#)

4.10 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 }

```

4.11 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 {
00014     public class World : MonoBehaviour
00015     {
00016         #region Data
00017         public Dictionary<ChunkWorldPos, Chunk> chunks = new Dictionary<ChunkWorldPos, Chunk>();
00021

```

```

00025     public GameObject chunkPrefab;
00026
00030     public bool chunkHasMadeCollisionMesh = false;
00031     #endregion
00032
00033     #region Creation and Destruction
00034     #region Chunk
00035     public void CreateChunk(int x, int y, int z)
00042     {
00043         /**pos of the chunk
00044         ChunkWorldPos pos = new ChunkWorldPos(x, y, z);
00045
00046         /**makes the chunk at the given position
00047         GameObject newChunk = Instantiate(chunkPrefab, new Vector3(x, y, z), Quaternion.identity);
00048
00049         Chunk chunk = newChunk.GetComponent<Chunk>();
00050
00051         /**setting the chunks pos and a reference to this
00052         chunk.chunkWorldPos = pos;
00053         chunk.world = this;
00054
00055         /**adds the 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)
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 changed 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 changed 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 changed 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  }
00149  #endregion
00150  #endregion
00151
00152  #region Get Things
00153  public Chunk GetChunk(int x, int y, int z)
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  }
00176
00184  public Block GetBlock(int x, int y, int z)
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.
x, y - chunk.chunkWorldPos.y, z - chunk.chunkWorldPos.
z);
00193      }
00194
00195      /*returns an empty block if 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 System.Collections.Generic;
00003 using System.Linq;
00004 using System.Text;
00005 using UnityEngine;
00006 using BeeGame.Terrain.Chunks;
00007 using BeeGame.Blocks;
00008 using BeeGame.Core;
00009
00010 namespace BeeGame.Terrain.LandGeneration
00011 {
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)
00018         {
00019             return new ChunkWorldPos()
00020             {
00021                 x = Mathf.RoundToInt(pos.x),
00022                 y = Mathf.RoundToInt(pos.y),
00023                 z = Mathf.RoundToInt(pos.z)
00024             };
00025         }
00026
00027         public static THVector3 GetBlockPos(RaycastHit hit)
00028         {
00029             THVector3 vec3 = new THVector3()
00030             {
00031                 x = RoundXZ(hit.point.x, hit.normal.x),
00032                 y = RoundY(hit.point.y, hit.normal.y),
00033                 z = RoundXZ(hit.point.z, hit.normal.z)
00034             };
00035             return (vec3);
00036         }
00037
00038         public static ChunkWorldPos GetBlockPosFromRayCast(RaycastHit
00039 hit)
00040         {
00041             return new ChunkWorldPos((int)RoundXZ(hit.point.x, hit.normal.x), (int)RoundY(hit.
00042 point.y, hit.normal.y), (int)RoundXZ(hit.point.z, hit.normal.z));
00043         }
00044
00045         static float RoundXZ(float pos, float normal)
00046         {
00047             /*if we are looking at + x/z vlaues
00048             if (pos > 0)
00049             {
00050                 if (normal > 0)
00051                 {
00052                     pos = (int)pos;
00053                     return pos;
00054                 }
00055                 else if (normal < 0)
00056                 {
00057                     pos = (int)pos;
00058                     return pos - 1;
00059                 }
00060                 else
00061                 {
00062                     if ((pos - (int)pos) > 0.5)
00063                     {
00064                         return (int)pos + 1;
00065                     }
00066                     return (int)pos;
00067                 }
00068             }
00069             else
00070             {
00071                 if (normal > 0)
00072                 {
00073                     pos = (int)pos + 1;
00074                     return pos;
00075                 }
00076                 else if (normal < 0)
00077                 {
00078                     pos = (int)pos + 1;
00079                     return pos;
00080                 }
00081                 else
00082                 {
00083                     if ((pos - (int)pos) > 0.5)
00084                     {
00085                         return (int)pos + 1;
00086                     }
00087                     return (int)pos;
00088                 }
00089             }
00090             return (int)pos;
00091         }
00092     }
```

```

00093     }
00094 }
00095 /**if we are looking at - x/z values
00096 else
00097 {
00098     /**if poitive normal
00099     if (normal > 0)
00100     {
00101         pos = (int)pos;
00102         return pos - 1;
00103     }
00104
00105     /**if negative nomrmal
00106     if (normal < 0)
00107     {
00108         pos = (int)pos;
00109         return pos;
00110     }
00111     /**if their is no normal
00112
00113     /**if pos is greater than 0.5 we are in the next block so go to it
00114     if ((-pos - (int)-pos) > 0.5)
00115     {
00116         return (int)pos - 1;
00117     }
00118
00119     return (int)pos;
00120 }
00121 }
00122
00132 static float RoundY(float pos, float normal)
00133 {
00134     pos = (float)Math.Round(pos, 1);
00135     if (pos >= 0)
00136     {
00137         if(normal > 0)
00138         {
00139             if((int)pos % 2 == 0)
00140                 return Mathf.RoundToInt((float)Math.Round(pos, 1));
00141
00142             return Mathf.RoundToInt((float)Math.Round(pos, 1)) - normal;
00143         }
00144
00145         if((int)pos % 2 == 0)
00146             return Mathf.RoundToInt((float)Math.Round(pos, 1)) - normal;
00147
00148         return Mathf.RoundToInt((float)Math.Round(pos, 1));
00149     }
00150
00151     if(pos <= 0)
00152     {
00153         if (normal > 0)
00154         {
00155             if ((int)pos % 2 == 0)
00156                 /**the Math.Round removes strange rounding errors shown with Mathf.Round eg
00157                 sometimes 0.5 would round to 0 not 1
00158                 return Mathf.RoundToInt((float)Math.Round(pos, 1)) - normal;
00159
00160                 return Mathf.RoundToInt((float)Math.Round(pos, 1)); // - normal;
00161         }
00162
00163         if ((int)pos % 2 == 0)
00164             return Mathf.RoundToInt((float)Math.Round(pos, 1));
00165
00166         return Mathf.RoundToInt((float)Math.Round(pos, 1)) - normal;
00167     }
00168
00169     return Mathf.RoundToInt((float)Math.Round(pos, 1));
00170 }
00171
00182 public static float Round(float pos, float norm, bool adjacent = false)
00183 {
00184     if(pos - (int)pos == 0.5f || pos - (int)pos == -0.5f)
00185     {
00186         if(adjacent)
00187         {
00188             pos += (norm / 2);
00189         }
00190         else
00191         {
00192             pos -= (norm / 2);
00193         }
00194     }
00195
00196     return pos;
00197 }

```

```

00198         #endregion
00199
00200     #region Get Block
00201     public static ChunkWorldPos GetBlockPos(RaycastHit hit, bool adjacent = false)
00202     {
00203         return GetBlockPos(new THVector3()
00204         {
00205             /*rounds the hit to the correct position
00206             x = Round(hit.point.x, hit.normal.x, adjacent),
00207             y = Round(hit.point.y, hit.normal.y, adjacent),
00208             z = Round(hit.point.z, hit.normal.z, adjacent)
00209             */);
00210         });
00211     }
00212
00213     public static Block GetBlock(RaycastHit hit, bool adjacent = false)
00214     {
00215         /*checks that a chunk was hit and if it wasnt return early
00216         Chunk chunk = hit.collider.GetComponent<Chunk>();
00217
00218         if (chunk == null)
00219             return null;
00220
00221         /*allignes the hit to the block grid and returns the block
00222         ChunkWorldPos pos = GetBlockPos(hit, adjacent);
00223
00224         return chunk.world.GetBlock(pos.x, pos.y, pos.z);
00225     }
00226
00227     public static Block GetBlock(THVector3 pos)
00228     {
00229         Chunk chunk = GetChunk(pos);
00230
00231         if (chunk == null)
00232             return new Air();
00233
00234         chunk.world.GetBlock((int)pos.x, (int)pos.y, (int)pos.z);
00235
00236         return new Block();
00237     }
00238
00239     public static bool BlockInPosition(THVector3 pos,
00240     Chunk chunk)
00241     {
00242         if (chunk == null)
00243             return false;
00244
00245         if (chunk.GetBlock((int)pos.x, (int)pos.y, (int)pos.z) != new
00246         Air())
00247             return true;
00248
00249             return false;
00250     }
00251     #endregion
00252
00253     public static Chunk GetChunk(THVector3 vec3)
00254     {
00255         return world.GetChunk((int)vec3.x, (int)vec3.y, (int)vec3.
00256         z);
00257     }
00258
00259     #region Set Block
00260     public static bool SetBlock(RaycastHit hit, Block block, bool adjacent = false)
00261     {
00262         /*checks that a chnk was hit
00263         Chunk chunk = hit.collider.GetComponent<Chunk>();
00264
00265         if (chunk == null)
00266             return false;
00267
00268         /*allignes the hit to the block grid
00269         ChunkWorldPos pos = GetBlockPosFromRayCast(hit);
00270
00271         /*checks that the block tryign to be replaced can be replaced eg bedrock cannot be replaced
00272         if (GetBlock(hit, adjacent).breakable)
00273         {
00274             /*sets the position of the block and saves the chunk
00275             chunk.world.SetBlock(pos.x, pos.y, pos.z, block);
00276             Serialization.Serialization.SaveChunk(chunk);
00277         }
00278
00279         return true;
00280     }
00281     #endregion
00282 }
00283
00284 }

```

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 {
00010     public class TerrainGeneration
00011     {
00012         #region Data
00013         private float stoneBaseHeight = -24;
00014         private float stoneBaseNoise = 0.05f;
00015         private float stoneBaseNoiseHeight = 4;
00016
00017         private float stoneMountainHeight = 48;
00018         private float stoneMountainFrequency = 0.008f;
00019         private float stoneMinHeight = -12;
00020
00021         private float dirtBaseHeight = 1;
00022         private float dirtNoise = 0.04f;
00023         private float dirtNoiseHeight = 3;
00024
00025         private float caveFrequency = 0.025f;
00026         private int caveSize = 8;
00027         #endregion
00028
00029         public Chunk ChunkGen(Chunk chunk)
00030         {
00031             Chunk outChunk = chunk;
00032             lock (chunk)
00033             {
00034                 Thread thread = new Thread(() => ChunkGenThread(chunk, out outChunk)) { Name = $"Generate
00035 Chunk Thread @ {chunk.chunkWorldPos}"};
00036
00037                 thread.Start();
00038                 return outChunk;
00039             }
00040         }
00041
00042         public void ChunkGenThread(Chunk chunk, out Chunk outChunk)
00043         {
00044             /*for each x and z position in teh chunk
00045             for (int x = chunk.chunkWorldPos.x; x < chunk.
00046 chunkWorldPos.x + Chunk.chunkSize; x++)
00047             {
00048                 for (int z = chunk.chunkWorldPos.z; z < chunk.
00049 chunkWorldPos.z + Chunk.chunkSize; z++)
00050                 {
00051                     chunk = GenChunkColum(chunk, x, z);
00052                 }
00053             }
00054
00055             chunk.SetBlocksUnmodified();
00056             outChunk = chunk;
00057         }
00058
00059         public Chunk GenChunkColum(Chunk chunk, int x, int z)
00060         {
00061             /*the height of the mountain
```

```

00112         int stoneHeight = Mathf.FloorToInt(stoneBaseHeight);
00113         stoneHeight += GetNoise(-x, 0, z, stoneMountainFrequency, Mathf.FloorToInt(stoneMountainHeight)
);
00114
00115         /**if the colum is currently to low make it not so low
00116         if (stoneHeight < stoneMinHeight)
00117             stoneHeight = Mathf.FloorToInt(stoneMinHeight);
00118
00119         /**add the height of normal stone on to the mountain
00120         stoneHeight += GetNoise(x, 0, -z, stoneBaseNoise, Mathf.RoundToInt(stoneBaseNoiseHeight));
00121
00122         /**put dirt on top
00123         int dirtHeight = stoneHeight + Mathf.FloorToInt(dirtBaseHeight);
00124         dirtHeight += GetNoise(x, 100, z, dirtNoise, Mathf.FloorToInt(dirtNoiseHeight));
00125
00126         /**set the colum to the correct blocks
00127         for (int y = chunk.chunkWorldPos.y; y < chunk.
chunkWorldPos.y + Chunk.chunkSize; y++)
00128         {
00129             int caveChance = GetNoise(x + 40, y + 100, z - 50, caveFrequency, 200);
00130
00131             /**puts a layer of bedrock at the bottom the the world
00132             if (y <= (chunk.chunkWorldPos.y) && chunk.
chunkWorldPos.y == -16)
00133             {
00134                 SetBlock(x, y, z, new Blocks.Bedrock(), chunk);
00135             }
00136             else if (y <= stoneHeight && caveSize < caveChance)
00137             {
00138                 SetBlock(x, y, z, new Blocks.Block(), chunk);
00139             }
00140             else if (y <= dirtHeight && caveSize < caveChance)
00141             {
00142                 SetBlock(x, y, z, new Blocks.Grass(), chunk);
00143             }
00144             else
00145             {
00146                 SetBlock(x, y, z, new Blocks.Air(), chunk);
00147             }
00148         }
00149
00150         return chunk;
00151     }
00152
00162     public static int GetNoise(int x, int y, int z, float scale, int max)
00163     {
00164         return Mathf.FloorToInt((SimplexNoise.Generate(x * scale, y * scale, z *
scale) + 1f) * (max / 2f));
00165     }
00166
00176     public static void SetBlock(int x, int y, int z, Blocks.Block block,
Chunk chunk, bool replacesBlocks = false)
00177     {
00178         /**corrects the x, y, z pos of the so that the block is placed in the correct position
00179         x -= chunk.chunkWorldPos.x;
00180         y -= chunk.chunkWorldPos.y;
00181         z -= chunk.chunkWorldPos.z;
00182
00183         /**chechs that the block is in the chunk and that no block is already their then sets it
00184         if (Chunk.InRange(x) && Chunk.InRange(y) &&
Chunk.InRange(z))
00185             if (replacesBlocks || chunk.blocks[x, y, z] == null)
00186                 chunk.SetBlock(x, y, z, block);
00187     }
00188 }
00189 }

```

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 n0, n1, n2;
00086              float x1 = x0 - (i1 - i) * 0.5f;
00087              float y1 = y0 - (j1 - j) * 0.5f;
00088              float x2 = x0 - i1 + j1;
00089              float y2 = y0 - j1 + i1;
00090              float t0 = 1.0f - x1 * x1 - y1 * y1;
00091              t0 *= t0;
00092              float t1 = 1.0f - x2 * x2 - y2 * y2;
00093              t1 *= t1;
00094              float n0 = t0 * t0 * grad(perm[i & 0xff], x0, y0);
00095              float n1 = t1 * t1 * grad(perm[(i + i1) & 0xff], x1, y1);
00096              float n2 = t1 * t1 * grad(perm[(j + j1) & 0xff], x2, y2);
00097              return 0.395f * (n0 + n1 + n2);
00098          }
00099      }
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 }
00139
00140
00141 public static float Generate(float x, float y, float z)
00142 {
00143     /* Simple skewing factors for the 3D case
00144     const float F3 = 0.333333333f;
00145     const float G3 = 0.166666667f;
00146
00147     float n0, n1, n2, n3; /* Noise contributions from the four corners
00148
00149     /* Skew the input space to determine which simplex cell we're in
00150     float s = (x + y + z) * F3; /* Very nice and simple skew factor for 3D
00151     float xs = x + s;
00152     float ys = y + s;
00153     float zs = z + s;
00154     int i = FastFloor(xs);
00155     int j = FastFloor(ys);
00156     int k = FastFloor(zs);
00157
00158     float t = (float)(i + j + k) * G3;
00159     float X0 = i - t; /* Unskew the cell origin back to (x,y,z) space
00160     float Y0 = j - t;
00161     float Z0 = k - t;
00162     float x0 = x - X0; /* The x,y,z distances from the cell origin
00163     float y0 = y - Y0;
00164     float z0 = z - Z0;
00165
00166     /* For the 3D case, the simplex shape is a slightly irregular tetrahedron.
00167     /* Determine which simplex we are in.
00168     int i1, j1, k1; /* Offsets for second corner of simplex in (i,j,k) coords
00169     int i2, j2, k2; /* Offsets for third corner of simplex in (i,j,k) coords
00170
00171     /* This code would benefit from a backport from the GLSL version! */
00172     if (x0 >= y0)
00173     {
00174         if (y0 >= z0)
00175         { i1 = 1; j1 = 0; k1 = 0; i2 = 1; j2 = 1; k2 = 0; } /* X Y Z order
00176         else if (x0 >= z0) { i1 = 1; j1 = 0; k1 = 0; i2 = 1; j2 = 0; k2 = 1; } /* X Z Y order
00177         else { i1 = 0; j1 = 0; k1 = 1; i2 = 1; j2 = 0; k2 = 1; } /* Z X Y order
00178     }
00179     else
00180     { /* x0 < y0
00181         if (y0 < z0) { i1 = 0; j1 = 0; k1 = 1; i2 = 0; j2 = 1; k2 = 1; } /* Z Y X order
00182         else if (x0 < z0) { i1 = 0; j1 = 1; k1 = 0; i2 = 0; j2 = 1; k2 = 1; } /* Y Z X order
00183         else { i1 = 0; j1 = 1; k1 = 0; i2 = 1; j2 = 1; k2 = 0; } /* Y X Z order
00184     }
00185
00186     /* A step of (1,0,0) in (i,j,k) means a step of (1-c,-c,-c) in (x,y,z),
00187     /* a step of (0,1,0) in (i,j,k) means a step of (-c,1-c,-c) in (x,y,z), and

```

```

00188     /** a step of (0,0,1) in (i,j,k) means a step of (-c,-c,1-c) in (x,y,z), where
00189     /** c = 1/6.
00190
00191     float x1 = x0 - i1 + G3; /** Offsets for second corner in (x,y,z) coords
00192     float y1 = y0 - j1 + G3;
00193     float z1 = z0 - k1 + G3;
00194     float x2 = x0 - i2 + 2.0f * G3; /** Offsets for third corner in (x,y,z) coords
00195     float y2 = y0 - j2 + 2.0f * G3;
00196     float z2 = z0 - k2 + 2.0f * G3;
00197     float x3 = x0 - 1.0f + 3.0f * G3; /** Offsets for last corner in (x,y,z) coords
00198     float y3 = y0 - 1.0f + 3.0f * G3;
00199     float z3 = z0 - 1.0f + 3.0f * G3;
00200
00201     /** Wrap the integer indices at 256, to avoid indexing perm[] out of bounds
00202     int ii = Mod(i, 256);
00203     int jj = Mod(j, 256);
00204     int kk = Mod(k, 256);
00205
00206     /** Calculate the contribution from the four corners
00207     float t0 = 0.6f - x0 * x0 - y0 * y0 - z0 * z0;
00208     if (t0 < 0.0f) n0 = 0.0f;
00209     else
00210     {
00211         t0 *= t0;
00212         n0 = t0 * t0 * grad(perm[ii + perm[jj + perm[kk]]], x0, y0, z0);
00213     }
00214
00215     float t1 = 0.6f - x1 * x1 - y1 * y1 - z1 * z1;
00216     if (t1 < 0.0f) n1 = 0.0f;
00217     else
00218     {
00219         t1 *= t1;
00220         n1 = t1 * t1 * grad(perm[ii + i1 + perm[jj + j1 + perm[kk + k1]]], x1, y1, z1);
00221     }
00222
00223     float t2 = 0.6f - x2 * x2 - y2 * y2 - z2 * z2;
00224     if (t2 < 0.0f) n2 = 0.0f;
00225     else
00226     {
00227         t2 *= t2;
00228         n2 = t2 * t2 * grad(perm[ii + i2 + perm[jj + j2 + perm[kk + k2]]], x2, y2, z2);
00229     }
00230
00231     float t3 = 0.6f - x3 * x3 - y3 * y3 - z3 * z3;
00232     if (t3 < 0.0f) n3 = 0.0f;
00233     else
00234     {
00235         t3 *= t3;
00236         n3 = t3 * t3 * grad(perm[ii + 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/PlayerMove.cs File Reference

Classes

- class [BeeGame.Player.PlayerMove](#)
Moves the player

Namespaces

- namespace [BeeGame.Player](#)

5.4 PlayerMove.cs

```

00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using System.Text;
00005 using UnityEngine;
00006 using BeeGame.Core;
00007
00008 namespace BeeGame.Player
00009 {
00013     [RequireComponent(typeof(Rigidbody))]
00014     public class PlayerMove : MonoBehaviour
00015     {
00016         #region Data
00017         public float speed = 10f;
00024         public float gravity = 9.81f;
00028         public float maxVelocity = 10f;

```

```

00029
00033     private bool canJump = false;
00037     public float jumpHeight = 2f;
00038
00042     private Rigidbody myRigidBody;
00043     #endregion
00044
00045     #region Unity Methods
00046     private void Awake()
00050     {
00051         myRigidBody = GetComponent<Rigidbody>();
00052
00053         //i want to use myown gravity and rotation
00054         myRigidBody.useGravity = false;
00055         myRigidBody.freezeRotation = true;
00056     }
00057
00061     void FixedUpdate()
00062     {
00063         //If the player is grounded it can move
00064         if (canJump)
00065         {
00066             MovePlayer();
00067         }
00068
00069         //adds the downward force
00070         myRigidBody.AddForce(new Vector3(0, myRigidBody.mass * -gravity, 0));
00071     }
00072
00077     private void OnCollisionStay(Collision collision)
00078     {
00079         canJump = true;
00080     }
00081     #endregion
00082
00083     #region Movement Methods
00084     void MovePlayer()
00088     {
00089         //Calculate the speed we want to achive
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     }
00113
00118     float VerticalJumpSpeed()
00119     {
00120         //Gets the correct of fore required for the player to reach the desired apex
00121         //Can this be done without Square Root as that take alot of work?
00122         return Mathf.Sqrt(2 * jumpHeight * gravity);
00123     }
00124     #endregion
00125 }
00126 }

```

5.5 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](#)

5.6 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 static BeeGame.Terrain.LandGeneration.Terrain;
00007 using static BeeGame.Core.THInput;
00008
00009 namespace BeeGame.Player
00010 {
00011     public class Selector : MonoBehaviour
00012     {
00013         #region Data
00014         public GameObject selector;
00021
00025         public LayerMask layers;
00029         private RaycastHit hit;
00030
00034         public int selectedHotbarSlot = 27;
00035         #endregion
00036
00037         #region Unity Methods
00038         void Awake()
00042         {
00043             selector = Instantiate(selector);
00044         }
00045
00049         void FixedUpdate()
00050         {
00051             if (!isAnotherInventoryOpen)
00052                 UpdateSelector();
00053         }
00054
00058         void Update()
00059         {
00060             if (!isAnotherInventoryOpen)
00061             {
00062                 if (GetButtonDown("Break Block"))
00063                     BreakBlock();
00064                 if (GetButtonDown("Place"))
00065                     PlaceBlock();
00066             }
00067         }
00068         #endregion
00069
00070         #region Update
00071         void UpdateSelector()
00075         {
00076             if (Physics.Raycast(transform.position, transform.forward, out hit, 15, layers))
00077             {
00078                 selector.SetActive(true);
00079                 selector.transform.position = GetBlockPos(hit);
00080                 /*selector.SetActive(BlockInPosition(GetBlockPos(hit),
00081 hit.collider.GetComponent<Chunk>()));
00082             }
00083             else
00084             {
00085                 selector.SetActive(false);
00086             }
00087             SelectedSlot();
00088         }
00092         void SelectedSlot()
00093         {
00094             /* adds 1 to the selected slot and if that is out of range set it to the first hotbar slot
00095             if (Input.GetAxis("Mouse ScrollWheel") > 0)
00096             {
00097                 selectedHotbarSlot += 1;
00098                 if (selectedHotbarSlot == 36)
00099                     selectedHotbarSlot = 27;
00100             }
00101             /* removes one from the hotbar selector and if the selector would be inside the inventory set
00102             it to the last slot in the hotbar
00103             else if (Input.GetAxis("Mouse ScrollWheel") < 0)
00104             {
00105                 selectedHotbarSlot -= 1;
```

```

00105             if (selectedHotbarSlot == 26)
00106                 selectedHotbarSlot = 35;
00107         }
00108
00109         transform.parent.GetComponentInChildren<PlayerInventory>().SelectedSlot (
selectedHotbarSlot);
00110     }
00111     #endregion
00112
00113     #region Break/Place
00114     void BreakBlock()
00115     {
00116         Chunk chunk = GetChunk(selector.transform.position);
00117
00118         Block block = chunk.world.GetBlock((int)selector.transform.position.x, (int)selector.
transform.position.y, (int)selector.transform.position.z);
00119
00120         if (!block.breakable)
00121             return;
00122
00123         chunk.world.SetBlock((int)selector.transform.position.x, (int)selector.transform.position.
y, (int)selector.transform.position.z, new Air(), true);
00124         /* set to changed so when block is placed down again it will be saved
00125         block.changed = true;
00126         block.BreakBlock(selector.transform.position);
00127     }
00128
00129     void PlaceBlock()
00130     {
00131         Chunk chunk = GetChunk(selector.transform.position);
00132
00133         if (chunk == null)
00134             return;
00135
00136         /* gets the item in the hotbar and if the item is placeable place it
00137         if(transform.parent.GetComponentInChildren<PlayerInventory>().
GetItemFromHotBar(selectedHotbarSlot, out Item blockToPlace))
00138             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, true);
00139     }
00140     #endregion
00141 }
00142
00143 }
00144 }
00145 }
00146 }
00147 }
00148 }

```

6 Resources

6.1 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/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 {
00006     public static class PrefabDictionary
00007     {

```

```

00014     private static Dictionary<string, GameObject> prefabDictionary = new Dictionary<string, GameObject>
00015     ();
00016
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/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/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`

6.6 Resources.Designer.cs

```
00001 /**-----
00002 /** <auto-generated>
00003 /** This code was generated by a tool.
00004 /** Runtime Version:4.0.30319.42000
00005 /**
00006 /** Changes to this file may cause incorrect behavior and will be lost if
00007 /** the code is regenerated.
00008 /** </auto-generated>
00009 /**-----
00010
00011 namespace BeeGame.Resources {
00012     using System;
00013     using System.Collections.Generic;
00014     using UnityEngine;
00015
00019     /** This class was auto-generated by the StronglyTypedResourceBuilder
00020     /** class via a tool like ResGen or Visual Studio.
00021     /** To add or remove a member, edit your .ResX file then rerun ResGen
00022     /** with the /str option, or rebuild your VS project.
00023     [global::System.CodeDom.Compiler.GeneratedCodeAttribute("
System.Resources.Tools.StronglyTypedResourceBuilder", "4.0.0.0")]
00024     [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
00025     [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
00026     internal class Resources {
00027
00028         private static global::System.Resources.ResourceManager
resourceMan;
00029
00030         private static global::System.Globalization.CultureInfo resourceCulture;
00031
00032         [global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance", "
CA1811:AvoidUncalledPrivateCode")]
00033         internal Resources() {
00034         }
00035
00039         [global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.
EditorBrowsableState.Advanced)]
00040         internal static global::System.Resources.ResourceManager ResourceManager {
00041             get {
00042                 if (object.ReferenceEquals(resourceMan, null)) {
00043                     global::System.Resources.ResourceManager temp = new global::System.Resources.
ResourceManager("BeeGame.Resources.Resources", typeof(Resources).Assembly);
00044                     resourceMan = temp;
00045                 }
00046                 return resourceMan;
00047             }
00048         }
00049
00054         [global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.
EditorBrowsableState.Advanced)]
00055         internal static global::System.Globalization.CultureInfo Culture {
00056             get {
00057                 return resourceCulture;
00058             }
00059             set {
00060                 resourceCulture = value;
00061             }
00062         }
00063
00067         internal static byte[] Prefabs {
00068             get {
00069                 object obj = ResourceManager.GetObject("Prefabs", resourceCulture);
00070                 return ((byte[])(obj));
00071             }
00072         }
00073
00077         internal static byte[] Sprites {
00078             get {
00079                 object obj = ResourceManager.GetObject("Sprites", resourceCulture);
00080                 return ((byte[])(obj));
00081             }
00082         }
00083
00084         internal static Dictionary<string, Sprite> GetSprites()
00085         {
00086             string[] splitCharacters = new string[] { ",", " " };
00087         }
00088     }
00089 }
```

```

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]) as Texture2D;
00106                 sprites.Add(splitText[0], Sprite.Create(tex, new UnityEngine.Rect(0, 0, tex.
width, tex.height), Vector2.zero));
00107             }
00108         }
00109
00110         splitText = lineText.Split(splitCharacters, StringSplitOptions.RemoveEmptyEntries);
00111         lineText = "";
00112         tex = UnityEngine.Resources.Load("Sprites/" + splitText[1]) as Texture2D;
00113         sprites.Add(splitText[0], Sprite.Create(tex, new UnityEngine.Rect(0, 0, tex.width,
tex.height), Vector2.zero));
00114
00115         return sprites;
00116     }
00117
00118     internal static Dictionary<string, GameObject> GetPrefabs()
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]) 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     }
00149 }
00150 }

```

6.7 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](#)

6.8 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             SpriteDictionary.LoadSprites();
00018             PrefabDictionary.LoadPrefabs();
00019         }
00020     }
00021 }
```

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
00040         public static bool GetButtonDown(string button)
00041         {
00042             if (!inputButtons.ContainsKey(button))
00043             {
00044                 throw new Exception("Input Manager: Key button name not defined: " + button);
00045             }
00046         }
00047     }
00048 }
```

```

00047         switch (inputButtons[button])
00048         {
00049             case KeyCode[] array:
00050                 /*for each possible key, check if it was pressed and if it was return that it was, if
00051                 none of them was poressed return false
00052                 foreach (var item in array)
00053                 {
00054                     if (Input.GetKeyDown(item))
00055                     {
00056                         return true;
00057                     }
00058                 }
00059                 return false;
00060             default:
00061                 return Input.GetKeyDown((KeyCode)inputButtons[button]);
00062         }
00063     }
00064
00070     public static bool GetButton(string button)
00071     {
00072         if (!inputButtons.ContainsKey(button))
00073         {
00074             throw new Exception("Input Manager: Key button name not defined: " + button);
00075         }
00076
00077         switch (inputButtons[button])
00078         {
00079             case KeyCode[] array:
00080                 /*for each possible key, check if it was pressed and if it was return that it was, if
00081                 none of them was poressed return false
00082                 foreach (var item in array)
00083                 {
00084                     if (Input.GetKey(item))
00085                     {
00086                         return true;
00087                     }
00088                 }
00089                 return false;
00090             default:
00091                 return Input.GetKey((KeyCode)inputButtons[button]);
00092         }
00093     }
00094
00100     public static bool GetButtonUp(string button)
00101     {
00102         if (!inputButtons.ContainsKey(button))
00103         {
00104             throw new Exception("Input Manager: Key button name not defined: " + button);
00105         }
00106
00107         switch (inputButtons[button])
00108         {
00109             case KeyCode[] array:
00110                 /*for each possible key, check if it was pressed and if it was return that it was, if
00111                 none of them was poressed return false
00112                 foreach (var item in array)
00113                 {
00114                     if (Input.GetKeyUp(item))
00115                     {
00116                         return true;
00117                     }
00118                 }
00119                 return false;
00120             default:
00121                 return Input.GetKeyUp((KeyCode)inputButtons[button]);
00122         }
00123     }
00124
00130     public static int GetAxis(string axis)
00131     {
00132         int returnAxis = 0;
00133
00134         if (axis == "Horizontal")
00135         {
00136             if (GetButton("Right"))
00137             {
00138                 returnAxis += 1;
00139             }
00140
00141             if (GetButton("Left"))
00142             {
00143                 returnAxis -= 1;
00144             }
00145         }

```

```

00146         else if (axis == "Vertical")
00147         {
00148             if (GetButton("Forward"))
00149             {
00150                 returnAxis += 1;
00151             }
00152             if (GetButton("Backward"))
00153             {
00154                 returnAxis -= 1;
00155             }
00156         }
00157     }
00158     return returnAxis;
00159 }
00160 }
00161 }
00162 }

```

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 {
00009     [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)
00071     {
00072         return (float)Math.Sqrt(Math.Pow((a.x - b.x), 2) + Math.Pow((a.y - b.
00073 y), 2) + Math.Pow((a.z - b.z), 2));
00074     }
00075     #endregion
00076
00077     #region Overrides
00078     public override bool Equals(object obj)
00079     {
00080         if (!(obj is THVector3))
00081             return false;
00082         if (obj.GetHashCode() == GetHashCode())
00083             return true;
00084         return false;
00085     }
00086
00087     public override int GetHashCode()
00088     {
00089         unchecked
00090         {
00091             int hash = 13;
00092
00093             hash *= 443 * x.GetHashCode();
00094             hash *= 373 * y.GetHashCode();
00095             hash *= 127 * z.GetHashCode();
00096
00097             return hash;
00098         }
00099     }
00100
00101     public override string ToString()
00102     {
00103         return $"{x}, {y}, {z}";
00104     }
00105
00106     public static bool operator ==(THVector3 a, THVector3 b)
00107     {
00108         return a.Equals(b);
00109     }
00110
00111     public static bool operator !=(THVector3 a, THVector3 b)
00112     {
00113         return !(a == b);
00114     }
00115
00116     public static THVector3 operator +(THVector3 a,
00117 THVector3 b)
00118     {
00119         {
00120             a.x += b.x;
00121             a.y += b.y;
00122             a.z += b.z;
00123
00124             return a;
00125         }
00126     }
00127
00128     public static THVector3 operator +(THVector3 a, float b)
00129     {
00130         {
00131             a.x += b;
00132             a.y += b;
00133             a.z += b;
00134
00135             return a;
00136         }
00137     }
00138
00139     public static THVector3 operator +(float a, THVector3 b)
00140     {
00141         {
00142             return new THVector3(a + b.x, a + b.y, a + b.z);
00143         }
00144     }
00145
00146     public static THVector3 operator -(THVector3 a,
00147 THVector3 b)
00148     {
00149         {
00150             a.x -= b.x;
00151             a.y -= b.y;
00152             a.z -= b.z;
00153         }
00154     }

```

```

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 }

```

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.Terrain;
00006 using BeeGame.Terrain.Chunks;
00007 using BeeGame.Inventory;
00008 using BeeGame.Blocks;
00009
00010 namespace BeeGame.Serialization
00011 {
00012     public static class Serialization
00013     {
00014         #region Data
00015         public static string worldName = "World";
00016         public static string saveFolderName = "Saves";
00017         private static string savePath;
00018         #endregion
00019
00020         public static void MakeDirectories()
00021         {
00022             savePath = $"{Application.dataPath}/{saveFolderName}/{worldName}";
00023
00024             if (!Directory.Exists(savePath))
00025                 Directory.CreateDirectory(savePath);
00026         }
00027
00028         #region Inventories
00029         public static void SerializeInventory(Inventory inventory, string inventoryName)
00030         {
00031             string inventorySavePath = $"{savePath}/Inventories";
00032
00033             if (!Directory.Exists(inventorySavePath))
00034                 Directory.CreateDirectory(inventorySavePath);
00035
00036             SaveFile(inventory.GetAllItems(), $"{inventorySavePath}/{inventoryName}.dat");
00037         }
00038
00039         public static void DeSerializeInventory(Inventory inventory,
00040 string inventoryName)
00041         {
00042             /** make the path
00043             string inventorySavePath = $"{savePath}/Inventories/{inventoryName}.dat";
00044
00045             /** checks that the file exists
00046             if (!File.Exists(inventorySavePath))
00047                 return;
00048
00049             inventory.SetAllItems((ItemsInInventory)LoadFile($"{inventorySavePath}"));
00050         }
00051         #endregion
00052
00053         #region Chunk
00054         public static void SaveChunk(Chunk chunk)
00055         {
00056             /** saves the blocks
00057             SaveChunk save = new SaveChunk(chunk.blocks);
00058
00059             /** if no block was changed return early
00060             if (save.blocks.Count == 0)
00061                 return;
00062
00063             /** otherwise save the file
00064             string saveFile = $"{savePath}/{FileName(chunk.chunkWorldPos)}.dat";
00065
00066             SaveFile(save, saveFile);
00067         }
00068
00069         public static bool LoadChunk(Chunk chunk)
00070         {
00071             /** gets the save file
```



```

00112         string saveFile = $"{savePath}/{FileName(chunk.chunkWorldPos)}.dat";
00113
00114         /* if the file does not exist return false
00115         if (!File.Exists(saveFile))
00116             return false;
00117
00118         /* set all of the changed blocks in the chunk
00119         SaveChunk save = (SaveChunk)LoadFile(saveFile);
00120
00121         foreach (var block in save.blocks)
00122         {
00123             chunk.blocks[block.Key.x, block.Key.y, block.Key.z] = block.Value;
00124         }
00125
00126         return true;
00127     }
00128
00134     public static string FileName(ChunkWorldPos pos)
00135     {
00136         return $"{pos.x}, {pos.y}, {pos.z}";
00137     }
00138     #endregion
00139
00140     #region Save/Load Files
00141     private static void SaveFile(object obj, string file)
00142     {
00143         BinaryFormatter bf = new BinaryFormatter();
00144         FileStream fs = new FileStream(file, FileMode.OpenOrCreate);
00145
00146         try
00147         {
00148             bf.Serialize(fs, obj);
00149         }
00150         catch (SerializationException e)
00151         {
00152             Debug.Log($"Serialization Exception: {e}");
00153             throw new SerializationException();
00154         }
00155         finally
00156         {
00157             fs.Close();
00158         }
00159     }
00160
00161     private static object LoadFile(string file)
00162     {
00163         BinaryFormatter bf = new BinaryFormatter();
00164         FileStream fs = new FileStream(file, FileMode.Open);
00165
00166         try
00167         {
00168             return bf.Deserialize(fs);
00169         }
00170         catch (SerializationException e)
00171         {
00172             Debug.Log($"Deserialization Exception {e}");
00173             throw new SerializationException();
00174         }
00175         finally
00176         {
00177             fs.Close();
00178         }
00179     }
00180     #endregion
00181 }
00182 }

```

8.3 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Extensions.cs File Reference

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         {
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
recursively 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             return objTarget;
00057         }
00058     }
00059 }
00060 }
```

8.5 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/test.cs File Reference

Classes

- class [BeeGame.Test](#)

Namespaces

- namespace [BeeGame](#)

8.6 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 }
```

8.7 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.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

8.8 Enums.cs

```
00001 namespace BeeGame.Core.Enums
00002 {
00006     public enum Direction
00007     {
00008         NORTH, EAST, SOUTH, WEST, UP, DOWN
00009     };
00010
00011 }
```

Index

- AddItem
 - BeeGame::Inventory::ItemsInInventory, [48](#), [49](#)
- AddItemToInventory
 - BeeGame::Inventory::Inventory, [43](#)
- AddItemToSlots
 - BeeGame::Inventory::Inventory, [43](#)
- AddQuadTriangles
 - BeeGame::Terrain::Chunks::MeshData, [70](#)
- AddToSlot
 - BeeGame::Inventory::InventorySlot, [51](#)
- AddTriangle
 - BeeGame::Terrain::Chunks::MeshData, [71](#)
- AddVertices
 - BeeGame::Terrain::Chunks::MeshData, [71](#)
- Air
 - BeeGame::Blocks::Air, [29](#)
- Apiary
 - BeeGame::Blocks::Apiary, [35](#)
- applyCollisionMesh
 - BeeGame::Terrain::Chunks::Chunk, [67](#)
- ApplyCollisionMeshToNearbyChunks
 - BeeGame::Terrain::Chunks::LoadChunks, [74](#)
- Awake
 - BeeGame::LoadResources, [123](#)
 - BeeGame::Player::PlayerMove, [115](#)
 - BeeGame::Player::Selector, [120](#)
- Bedrock
 - BeeGame::Blocks::Bedrock, [32](#)
- BeeGame, [2](#)
- BeeGame.Blocks, [2](#)
- BeeGame.Blocks.Air, [28](#)
- BeeGame.Blocks.Apiary, [34](#)
- BeeGame.Blocks.Bedrock, [32](#)
- BeeGame.Blocks.Block, [22](#)
- BeeGame.Blocks.Dirt, [36](#)
- BeeGame.Blocks.Grass, [38](#)
- BeeGame.Core, [2](#)
- BeeGame.Core.Enums, [3](#)
- BeeGame.Core.Extensions, [161](#)
- BeeGame.Core.PrefabDictionary, [124](#)
- BeeGame.Core.SpriteDictionary, [125](#)
- BeeGame.Core.THInput, [130](#)
- BeeGame.Core.THVector2, [135](#)
- BeeGame.Core.THVector3, [142](#)
- BeeGame.Inventory, [4](#)
- BeeGame.Inventory.Inventory, [42](#)
- BeeGame.Inventory.InventorySlot, [50](#)
- BeeGame.Inventory.ItemsInInventory, [47](#)
- BeeGame.Inventory.Player_Inventory, [4](#)
- BeeGame.Inventory.Player_Inventory.PlayerInventory,
[56](#)
- BeeGame.Items, [4](#)
- BeeGame.Items.Item, [6](#)
- BeeGame.Items.ItemGameObject, [20](#)
- BeeGame.Items.Tile, [21](#)
- BeeGame.LoadResources, [123](#)
- BeeGame.Player, [4](#)
- BeeGame.Player.PlayerLook, [111](#)
- BeeGame.Player.PlayerMove, [114](#)
- BeeGame.Player.Selector, [119](#)
- BeeGame.Resources, [4](#)
- BeeGame.Resources.Resources, [127](#)
- BeeGame.Serialization, [5](#)
- BeeGame.Serialization.Serialization, [155](#)
- BeeGame.Terrain, [5](#)
- BeeGame.Terrain.ChunkWorldPos, [80](#)
- BeeGame.Terrain.Chunks, [5](#)
- BeeGame.Terrain.Chunks.Chunk, [61](#)
- BeeGame.Terrain.Chunks.LoadChunks, [73](#)
- BeeGame.Terrain.Chunks.MeshData, [70](#)
- BeeGame.Terrain.Chunks.SaveChunk, [79](#)
- BeeGame.Terrain.LandGeneration, [5](#)
- BeeGame.Terrain.LandGeneration.Noise, [6](#)
- BeeGame.Terrain.LandGeneration.Noise.SimplexNoise,
[105](#)
- BeeGame.Terrain.LandGeneration.Terrain, [90](#)
- BeeGame.Terrain.LandGeneration.TerrainGeneration,
[98](#)
- BeeGame.Terrain.LandGeneration.World, [84](#)
- BeeGame.Test, [162](#)
- BeeGame::Blocks::Air
 - Air, [29](#)
 - BlockData, [29](#)
 - BreakBlock, [30](#)
 - GetHashCode, [30](#)
 - IsSolid, [31](#)
 - ToString, [31](#)
- BeeGame::Blocks::Apiary
 - Apiary, [35](#)
 - GetHashCode, [36](#)
 - ToString, [36](#)
- BeeGame::Blocks::Bedrock
 - Bedrock, [32](#)
 - BreakBlock, [33](#)
 - GetHashCode, [33](#)
 - TexturePosition, [33](#)
 - ToString, [34](#)
- BeeGame::Blocks::Block
 - Block, [23](#)
 - BlockData, [24](#)
 - BreakBlock, [25](#)
 - breakable, [28](#)
 - changed, [28](#)
 - GetHashCode, [25](#)
 - IsSolid, [26](#)
 - ToString, [26](#)
 - UpdateBlock, [26](#)
- BeeGame::Blocks::Dirt
 - Dirt, [37](#)

- GetHashCode, 37
- TexturePosition, 37
- ToString, 38
- BeeGame::Blocks::Grass
 - GetHashCode, 39
 - GetItemName, 39
 - Grass, 39
 - TexturePosition, 40
 - ToString, 40
 - UpdateBlock, 41
- BeeGame::Core::Enums
 - Direction, 3
- BeeGame::Core::Extensions
 - CloneObject< T >, 161
- BeeGame::Core::PrefabDictionary
 - GetPrefab, 124
 - LoadPrefabs, 125
 - prefabDictionary, 125
- BeeGame::Core::SpriteDictionary
 - GetSprite, 126
 - itemSpriteDictionary, 127
 - LoadSprites, 126
- BeeGame::Core::THInput
 - GetAxis, 131
 - GetButton, 132
 - GetButtonDown, 133
 - GetButtonUp, 133
 - inputButtons, 134
 - isAnotherInventoryOpen, 134
- BeeGame::Core::THVector2
 - Equals, 137
 - GetHashCode, 137
 - operator THVector2, 137
 - operator Vector2, 138
 - operator!=, 138
 - operator*, 138, 139
 - operator+, 139
 - operator-, 140
 - operator/, 140, 141
 - operator==, 141
 - THVector2, 136, 137
 - ToString, 141
 - x, 142
 - y, 142
- BeeGame::Core::THVector3
 - Distance, 145
 - Equals, 146
 - GetHashCode, 146
 - operator THVector3, 146
 - operator Vector3, 147
 - operator!=, 147
 - operator*, 148, 149
 - operator+, 149, 150
 - operator-, 151, 152
 - operator/, 152, 153
 - operator==, 154
 - THVector3, 144, 145
 - ToString, 154
- x, 155
- y, 155
- z, 155
- BeeGame::Inventory::Inventory
 - AddItemToInventory, 43
 - AddItemToSlots, 43
 - floatingItem, 46
 - GetAllItems, 44
 - inventoryName, 47
 - InventorySet, 44
 - items, 47
 - PutItemsInSlots, 44
 - SetAllItems, 45
 - SetInventorySize, 46
 - slots, 47
 - UpdateBase, 46
- BeeGame::Inventory::InventorySlot
 - AddToSlot, 51
 - CheckFloatingItem, 51
 - item, 55
 - itemText, 55
 - myInventory, 56
 - OnDisable, 51
 - OnPointerClick, 52
 - OnPointerEnter, 53
 - OnPointerExit, 53
 - selectedSlot, 56
 - slotIndex, 56
 - SplitStack, 54
 - SwapItems, 54
 - Update, 54
 - UpdateIcon, 55
- BeeGame::Inventory::ItemsInInventory
 - AddItem, 48, 49
 - ItemsInInventory, 48
 - itemsInInventory, 49
- BeeGame::Inventory::Player_Inventory::PlayerInventory
 - GetItemFromHotBar, 57
 - OpenPlayerInventory, 58
 - PickupItem, 58
 - playerInventory, 61
 - RemoveItemFromInventory, 59
 - SelectedSlot, 59
 - SetPlayerInventory, 60
 - Start, 60
 - Update, 60
- BeeGame::Items::Item
 - Clone, 8
 - Equals, 8
 - FaceDataDown, 9
 - FaceDataEast, 10
 - FaceDataNorth, 10
 - FaceDataSouth, 11
 - FaceDataUp, 12
 - FaceDataWest, 13
 - FaceUVs, 14
 - GetGameObject, 14
 - GetHashCode, 14

- GetItemID, 15
- GetItemName, 15
- GetItemSprite, 15
- Item, 8
- ItemMesh, 16
- itemName, 18
- itemStackCount, 18
- maxStackCount, 19
- operator!=, 16
- operator==, 17
- placeable, 19
- TexturePosition, 17
- tileSize, 19
- ToString, 18
- usesGameObject, 19
- BeeGame::Items::ItemGameObject
 - go, 21
 - item, 21
 - MakeMesh, 20
 - Start, 20
- BeeGame::Items::Tile
 - x, 22
 - y, 22
- BeeGame::LoadResources
 - Awake, 123
- BeeGame::Player::PlayerLook
 - cameraTransform, 113
 - Look, 112
 - myTransform, 113
 - rotationLock, 113
 - speed, 114
 - Start, 112
 - Update, 113
 - xRot, 114
 - yRot, 114
- BeeGame::Player::PlayerMove
 - Awake, 115
 - canJump, 117
 - FixedUpdate, 116
 - gravity, 117
 - jumpHeight, 118
 - maxVelocity, 118
 - MovePlayer, 116
 - myRigidBody, 118
 - OnCollisionStay, 116
 - speed, 118
 - VerticalJumpSpeed, 117
- BeeGame::Player::Selector
 - Awake, 120
 - BreakBlock, 120
 - FixedUpdate, 120
 - hit, 122
 - layers, 122
 - PlaceBlock, 120
 - selectedHotbarSlot, 122
 - SelectedSlot, 121
 - selector, 122
 - Update, 121
- UpdateSelector, 121
- BeeGame::Resources::Resources
 - Culture, 129
 - GetPrefabs, 128
 - GetSprites, 128
 - Prefabs, 130
 - resourceCulture, 129
 - resourceMan, 129
 - ResourceManager, 130
 - Resources, 128
 - Sprites, 130
- BeeGame::Serialization::Serialization
 - DeSerializeInventory, 156
 - FileName, 157
 - LoadChunk, 157
 - LoadFile, 158
 - MakeDirectorys, 158
 - SaveChunk, 159
 - SaveFile, 159
 - saveFolderName, 160
 - savePath, 160
 - SerializeInventory, 160
 - worldName, 161
- BeeGame::Terrain::ChunkWorldPos
 - ChunkWorldPos, 81
 - Equals, 81
 - GetHashCode, 82
 - operator ChunkWorldPos, 82
 - operator THVector3, 83
 - ToString, 83
 - x, 83
 - y, 83
 - z, 84
- BeeGame::Terrain::Chunks::Chunk
 - applyCollisionMesh, 67
 - blocks, 68
 - chunkSize, 68
 - chunkWorldPos, 68
 - ColliderMesh, 63
 - filter, 68
 - GetBlock, 63
 - InRange, 64
 - mesh, 68
 - meshCollider, 69
 - RenderMesh, 65
 - rendered, 69
 - SetBlock, 65
 - SetBlocksUnmodified, 66
 - Start, 66
 - Update, 66
 - update, 69
 - UpdateChunk, 67
 - updateCollisionMesh, 69
 - world, 69
- BeeGame::Terrain::Chunks::LoadChunks
 - ApplyCollisionMeshToNearbyChunks, 74
 - BuildChunk, 75
 - buildList, 78

- chunkPositions, 78
- DeleteChunks, 75
- FindChunksToLoad, 76
- LoadAndRenderChunks, 77
- nearbyChunks, 78
- Start, 77
- timer, 78
- Update, 77
- world, 78
- BeeGame::Terrain::Chunks::MeshData
 - AddQuadTriangles, 70
 - AddTriangle, 71
 - AddVertices, 71
 - colTris, 72
 - colVerts, 72
 - done, 72
 - shareMeshes, 72
 - tris, 72
 - uv, 73
 - verts, 73
- BeeGame::Terrain::Chunks::SaveChunk
 - blocks, 80
 - SaveChunk, 79
- BeeGame::Terrain::LandGeneration::Noise::Simplex↔
 - Noise
 - FastFloor, 105
 - Generate, 106, 107
 - grad, 109, 110
 - Mod, 110
 - perm, 110
- BeeGame::Terrain::LandGeneration::Terrain
 - BlockInPosition, 91
 - GetBlock, 91, 92
 - GetBlockPos, 92, 93
 - GetBlockPosFromRayCast, 94
 - GetChunk, 94
 - Round, 94
 - RoundXZ, 95
 - RoundY, 96
 - SetBlock, 97
 - world, 98
- BeeGame::Terrain::LandGeneration::TerrainGeneration
 - caveFrequency, 102
 - caveSize, 103
 - ChunkGen, 99
 - ChunkGenThread, 100
 - dirtBaseHeight, 103
 - dirtNoise, 103
 - dirtNoiseHeight, 103
 - GenChunkColumn, 100
 - GetNoise, 101
 - SetBlock, 102
 - stoneBaseHeight, 103
 - stoneBaseNoise, 104
 - stoneBaseNoiseHeight, 104
 - stoneMinHeight, 104
 - stoneMountainFrequency, 104
 - stoneMountainHeight, 104
- BeeGame::Terrain::LandGeneration::World
 - chunkHasMadeCollisionMesh, 89
 - chunkPrefab, 89
 - chunks, 89
 - CreateChunk, 85
 - DestroyChunk, 86
 - GetBlock, 86
 - GetChunk, 87
 - SetBlock, 88
 - UpdateIfEqual, 89
- BeeGame::Test
 - Start, 163
- Block
 - BeeGame::Blocks::Block, 23
- BlockData
 - BeeGame::Blocks::Air, 29
 - BeeGame::Blocks::Block, 24
- BlockInPosition
 - BeeGame::Terrain::LandGeneration::Terrain, 91
- blocks
 - BeeGame::Terrain::Chunks::Chunk, 68
 - BeeGame::Terrain::Chunks::SaveChunk, 80
- BreakBlock
 - BeeGame::Blocks::Air, 30
 - BeeGame::Blocks::Bedrock, 33
 - BeeGame::Blocks::Block, 25
 - BeeGame::Player::Selector, 120
- breakable
 - BeeGame::Blocks::Block, 28
- BuildChunk
 - BeeGame::Terrain::Chunks::LoadChunks, 75
- buildList
 - BeeGame::Terrain::Chunks::LoadChunks, 78
- C:/Users/Toothless/Documents/GitHub/BeeGame/↔
 - Code/BeeGame/BeeGame/Blocks/Air.cs, 169, 170
- C:/Users/Toothless/Documents/GitHub/BeeGame/↔
 - Code/BeeGame/BeeGame/Blocks/Apiary.cs, 171
- C:/Users/Toothless/Documents/GitHub/BeeGame/↔
 - Code/BeeGame/BeeGame/Blocks/Bedrock.↔cs, 170, 171
- C:/Users/Toothless/Documents/GitHub/BeeGame/↔
 - Code/BeeGame/BeeGame/Blocks/Block.cs, 168
- C:/Users/Toothless/Documents/GitHub/BeeGame/↔
 - Code/BeeGame/BeeGame/Blocks/Dirt.cs, 172
- C:/Users/Toothless/Documents/GitHub/BeeGame/↔
 - Code/BeeGame/BeeGame/Blocks/Grass.cs, 173
- C:/Users/Toothless/Documents/GitHub/BeeGame/↔
 - Code/BeeGame/BeeGame/Core/Enums/↔Enums.cs, 218
- C:/Users/Toothless/Documents/GitHub/BeeGame/↔
 - Code/BeeGame/BeeGame/Core/Extensions.↔cs, 216, 217

C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Core/Prefab↵
 Dictionary.cs, 204
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Core/Sprite↵
 Dictionary.cs, 205
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Core/UnityType↵
 Replacements/THInput.cs, 208
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Core/UnityType↵
 Replacements/THVector2.cs, 210
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Core/UnityType↵
 Replacements/THVector3.cs, 212
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Inventory/↵
 Inventory.cs, 174
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Inventory/↵
 InventorySlot.cs, 176
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Inventory/Items↵
 InInventory.cs, 175
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Inventory/Player
 Inventory/PlayerInventory.cs, 178, 179
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Items/Item.cs,
 163, 164
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Items/Item↵
 GameObject.cs, 167
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/LoadResources.↵
 cs, 207, 208
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Player/Player↵
 Look.cs, 200
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Player/Player↵
 Move.cs, 201
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Player/Selector.↵
 cs, 202, 203
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Resources/↵
 Resources.Designer.cs, 205, 206
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Serialization/↵
 Serialization.cs, 214, 215
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Terrain/Chunk↵
 WorldPos.cs, 188
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Terrain/Chunks/↵
 LoadChunks.cs, 184
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Terrain/Chunks/↵
 MeshData.cs, 183
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Terrain/Chunks/↵
 SaveChunk.cs, 187, 188
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Terrain/Land↵
 Generation/Noise/SimplexNoise.cs, 196, 197
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Terrain/Land↵
 Generation/Terrain.cs, 191, 192
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Terrain/Land↵
 Generation/TerrainGeneration.cs, 195
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/Terrain/Land↵
 Generation/World.cs, 189
 C:/Users/Toothless/Documents/GitHub/BeeGame/↵
 Code/BeeGame/BeeGame/test.cs, 217, 218
 cameraTransform
 BeeGame::Player::PlayerLook, 113
 canJump
 BeeGame::Player::PlayerMove, 117
 caveFrequency
 BeeGame::Terrain::LandGeneration::Terrain↵
 Generation, 102
 caveSize
 BeeGame::Terrain::LandGeneration::Terrain↵
 Generation, 103
 changed
 BeeGame::Blocks::Block, 28
 CheckFloatingItem
 BeeGame::Inventory::InventorySlot, 51
 ChunkGen
 BeeGame::Terrain::LandGeneration::Terrain↵
 Generation, 99
 ChunkGenThread
 BeeGame::Terrain::LandGeneration::Terrain↵
 Generation, 100
 chunkHasMadeCollisionMesh
 BeeGame::Terrain::LandGeneration::World, 89
 chunkPositions
 BeeGame::Terrain::Chunks::LoadChunks, 78
 chunkPrefab
 BeeGame::Terrain::LandGeneration::World, 89
 chunkSize
 BeeGame::Terrain::Chunks::Chunk, 68
 ChunkWorldPos
 BeeGame::Terrain::ChunkWorldPos, 81
 chunkWorldPos
 BeeGame::Terrain::Chunks::Chunk, 68
 chunks
 BeeGame::Terrain::LandGeneration::World, 89
 Clone
 BeeGame::Items::Item, 8

- CloneObject< T >
 - BeeGame::Core::Extensions, [161](#)
- colTris
 - BeeGame::Terrain::Chunks::MeshData, [72](#)
- colVerts
 - BeeGame::Terrain::Chunks::MeshData, [72](#)
- ColliderMesh
 - BeeGame::Terrain::Chunks::Chunk, [63](#)
- CreateChunk
 - BeeGame::Terrain::LandGeneration::World, [85](#)
- Culture
 - BeeGame::Resources::Resources, [129](#)
- DeSerializeInventory
 - BeeGame::Serialization::Serialization, [156](#)
- DeleteChunks
 - BeeGame::Terrain::Chunks::LoadChunks, [75](#)
- DestroyChunk
 - BeeGame::Terrain::LandGeneration::World, [86](#)
- Direction
 - BeeGame::Core::Enums, [3](#)
- Dirt
 - BeeGame::Blocks::Dirt, [37](#)
- dirtBaseHeight
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, [103](#)
- dirtNoise
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, [103](#)
- dirtNoiseHeight
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, [103](#)
- Distance
 - BeeGame::Core::THVector3, [145](#)
- done
 - BeeGame::Terrain::Chunks::MeshData, [72](#)
- Equals
 - BeeGame::Core::THVector2, [137](#)
 - BeeGame::Core::THVector3, [146](#)
 - BeeGame::Items::Item, [8](#)
 - BeeGame::Terrain::ChunkWorldPos, [81](#)
- FaceDataDown
 - BeeGame::Items::Item, [9](#)
- FaceDataEast
 - BeeGame::Items::Item, [10](#)
- FaceDataNorth
 - BeeGame::Items::Item, [10](#)
- FaceDataSouth
 - BeeGame::Items::Item, [11](#)
- FaceDataUp
 - BeeGame::Items::Item, [12](#)
- FaceDataWest
 - BeeGame::Items::Item, [13](#)
- FaceUVs
 - BeeGame::Items::Item, [14](#)
- FastFloor
 - BeeGame::Terrain::LandGeneration::Noise::↔
SimplexNoise, [105](#)
- FileName
 - BeeGame::Serialization::Serialization, [157](#)
- filter
 - BeeGame::Terrain::Chunks::Chunk, [68](#)
- FindChunksToLoad
 - BeeGame::Terrain::Chunks::LoadChunks, [76](#)
- FixedUpdate
 - BeeGame::Player::PlayerMove, [116](#)
 - BeeGame::Player::Selector, [120](#)
- floatingItem
 - BeeGame::Inventory::Inventory, [46](#)
- GenChunkColumn
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, [100](#)
- Generate
 - BeeGame::Terrain::LandGeneration::Noise::↔
SimplexNoise, [106](#), [107](#)
- GetAllItems
 - BeeGame::Inventory::Inventory, [44](#)
- GetAxis
 - BeeGame::Core::THInput, [131](#)
- GetBlock
 - BeeGame::Terrain::Chunks::Chunk, [63](#)
 - BeeGame::Terrain::LandGeneration::Terrain, [91](#),
[92](#)
 - BeeGame::Terrain::LandGeneration::World, [86](#)
- GetBlockPos
 - BeeGame::Terrain::LandGeneration::Terrain, [92](#),
[93](#)
- GetBlockPosFromRayCast
 - BeeGame::Terrain::LandGeneration::Terrain, [94](#)
- GetButton
 - BeeGame::Core::THInput, [132](#)
- GetButtonDown
 - BeeGame::Core::THInput, [133](#)
- GetButtonUp
 - BeeGame::Core::THInput, [133](#)
- GetChunk
 - BeeGame::Terrain::LandGeneration::Terrain, [94](#)
 - BeeGame::Terrain::LandGeneration::World, [87](#)
- GetGameObject
 - BeeGame::Items::Item, [14](#)
- GetHashCode
 - BeeGame::Blocks::Air, [30](#)
 - BeeGame::Blocks::Apiary, [36](#)
 - BeeGame::Blocks::Bedrock, [33](#)
 - BeeGame::Blocks::Block, [25](#)
 - BeeGame::Blocks::Dirt, [37](#)
 - BeeGame::Blocks::Grass, [39](#)
 - BeeGame::Core::THVector2, [137](#)
 - BeeGame::Core::THVector3, [146](#)
 - BeeGame::Items::Item, [14](#)
 - BeeGame::Terrain::ChunkWorldPos, [82](#)
- GetItemFromHotBar
 - BeeGame::Inventory::Player_Inventory::Player↔
Inventory, [57](#)

- GetItemID
 - BeeGame::Items::Item, 15
- GetItemName
 - BeeGame::Blocks::Grass, 39
 - BeeGame::Items::Item, 15
- GetItemSprite
 - BeeGame::Items::Item, 15
- GetNoise
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, 101
- GetPrefab
 - BeeGame::Core::PrefabDictionary, 124
- GetPrefabs
 - BeeGame::Resources::Resources, 128
- GetSprite
 - BeeGame::Core::SpriteDictionary, 126
- GetSprites
 - BeeGame::Resources::Resources, 128
- go
 - BeeGame::Items::ItemGameObject, 21
- grad
 - BeeGame::Terrain::LandGeneration::Noise::↔
SimplexNoise, 109, 110
- Grass
 - BeeGame::Blocks::Grass, 39
- gravity
 - BeeGame::Player::PlayerMove, 117
- hit
 - BeeGame::Player::Selector, 122
- InRange
 - BeeGame::Terrain::Chunks::Chunk, 64
- inputButtons
 - BeeGame::Core::THInput, 134
- inventoryName
 - BeeGame::Inventory::Inventory, 47
- InventorySet
 - BeeGame::Inventory::Inventory, 44
- isAnotherInventoryOpen
 - BeeGame::Core::THInput, 134
- IsSolid
 - BeeGame::Blocks::Air, 31
 - BeeGame::Blocks::Block, 26
- Item
 - BeeGame::Items::Item, 8
- item
 - BeeGame::Inventory::InventorySlot, 55
 - BeeGame::Items::ItemGameObject, 21
- ItemMesh
 - BeeGame::Items::Item, 16
- itemName
 - BeeGame::Items::Item, 18
- itemSpriteDictionary
 - BeeGame::Core::SpriteDictionary, 127
- itemStackCount
 - BeeGame::Items::Item, 18
- itemText
 - BeeGame::Inventory::InventorySlot, 55
- items
 - BeeGame::Inventory::Inventory, 47
- ItemsInInventory
 - BeeGame::Inventory::ItemsInInventory, 48
- itemsInInventory
 - BeeGame::Inventory::ItemsInInventory, 49
- jumpHeight
 - BeeGame::Player::PlayerMove, 118
- layers
 - BeeGame::Player::Selector, 122
- LoadAndRenderChunks
 - BeeGame::Terrain::Chunks::LoadChunks, 77
- LoadChunk
 - BeeGame::Serialization::Serialization, 157
- LoadFile
 - BeeGame::Serialization::Serialization, 158
- LoadPrefabs
 - BeeGame::Core::PrefabDictionary, 125
- LoadSprites
 - BeeGame::Core::SpriteDictionary, 126
- Look
 - BeeGame::Player::PlayerLook, 112
- MakeDirectorys
 - BeeGame::Serialization::Serialization, 158
- MakeMesh
 - BeeGame::Items::ItemGameObject, 20
- maxStackCount
 - BeeGame::Items::Item, 19
- maxVelocity
 - BeeGame::Player::PlayerMove, 118
- mesh
 - BeeGame::Terrain::Chunks::Chunk, 68
- meshCollider
 - BeeGame::Terrain::Chunks::Chunk, 69
- Mod
 - BeeGame::Terrain::LandGeneration::Noise::↔
SimplexNoise, 110
- MovePlayer
 - BeeGame::Player::PlayerMove, 116
- myInventory
 - BeeGame::Inventory::InventorySlot, 56
- myRigidBody
 - BeeGame::Player::PlayerMove, 118
- myTransform
 - BeeGame::Player::PlayerLook, 113
- nearbyChunks
 - BeeGame::Terrain::Chunks::LoadChunks, 78
- OnCollisionStay
 - BeeGame::Player::PlayerMove, 116
- OnDisable
 - BeeGame::Inventory::InventorySlot, 51
- OnPointerClick
 - BeeGame::Inventory::InventorySlot, 52
- OnPointerEnter

- BeeGame::Inventory::InventorySlot, [53](#)
- OnPointerExit
 - BeeGame::Inventory::InventorySlot, [53](#)
- OpenPlayerInventory
 - BeeGame::Inventory::Player_Inventory::Player↔
Inventory, [58](#)
- operator ChunkWorldPos
 - BeeGame::Terrain::ChunkWorldPos, [82](#)
- operator THVector2
 - BeeGame::Core::THVector2, [137](#)
- operator THVector3
 - BeeGame::Core::THVector3, [146](#)
 - BeeGame::Terrain::ChunkWorldPos, [83](#)
- operator Vector2
 - BeeGame::Core::THVector2, [138](#)
- operator Vector3
 - BeeGame::Core::THVector3, [147](#)
- operator!=
 - BeeGame::Core::THVector2, [138](#)
 - BeeGame::Core::THVector3, [147](#)
 - BeeGame::Items::Item, [16](#)
- operator*
 - BeeGame::Core::THVector2, [138](#), [139](#)
 - BeeGame::Core::THVector3, [148](#), [149](#)
- operator+
 - BeeGame::Core::THVector2, [139](#)
 - BeeGame::Core::THVector3, [149](#), [150](#)
- operator-
 - BeeGame::Core::THVector2, [140](#)
 - BeeGame::Core::THVector3, [151](#), [152](#)
- operator/
 - BeeGame::Core::THVector2, [140](#), [141](#)
 - BeeGame::Core::THVector3, [152](#), [153](#)
- operator==
 - BeeGame::Core::THVector2, [141](#)
 - BeeGame::Core::THVector3, [154](#)
 - BeeGame::Items::Item, [17](#)
- perm
 - BeeGame::Terrain::LandGeneration::Noise::↔
SimplexNoise, [110](#)
- PickupItem
 - BeeGame::Inventory::Player_Inventory::Player↔
Inventory, [58](#)
- PlaceBlock
 - BeeGame::Player::Selector, [120](#)
- placeable
 - BeeGame::Items::Item, [19](#)
- playerInventory
 - BeeGame::Inventory::Player_Inventory::Player↔
Inventory, [61](#)
- prefabDictionary
 - BeeGame::Core::PrefabDictionary, [125](#)
- Prefabs
 - BeeGame::Resources::Resources, [130](#)
- PutItemsInSlots
 - BeeGame::Inventory::Inventory, [44](#)
- RemoveItemFromInventory
 - BeeGame::Inventory::Player_Inventory::Player↔
Inventory, [59](#)
- RenderMesh
 - BeeGame::Terrain::Chunks::Chunk, [65](#)
- rendered
 - BeeGame::Terrain::Chunks::Chunk, [69](#)
- resourceCulture
 - BeeGame::Resources::Resources, [129](#)
- resourceMan
 - BeeGame::Resources::Resources, [129](#)
- ResourceManager
 - BeeGame::Resources::Resources, [130](#)
- Resources
 - BeeGame::Resources::Resources, [128](#)
- rotationLock
 - BeeGame::Player::PlayerLook, [113](#)
- Round
 - BeeGame::Terrain::LandGeneration::Terrain, [94](#)
- RoundXZ
 - BeeGame::Terrain::LandGeneration::Terrain, [95](#)
- RoundY
 - BeeGame::Terrain::LandGeneration::Terrain, [96](#)
- SaveChunk
 - BeeGame::Serialization::Serialization, [159](#)
 - BeeGame::Terrain::Chunks::SaveChunk, [79](#)
- SaveFile
 - BeeGame::Serialization::Serialization, [159](#)
- saveFolderName
 - BeeGame::Serialization::Serialization, [160](#)
- savePath
 - BeeGame::Serialization::Serialization, [160](#)
- selectedHotbarSlot
 - BeeGame::Player::Selector, [122](#)
- SelectedSlot
 - BeeGame::Inventory::Player_Inventory::Player↔
Inventory, [59](#)
 - BeeGame::Player::Selector, [121](#)
- selectedSlot
 - BeeGame::Inventory::InventorySlot, [56](#)
- selector
 - BeeGame::Player::Selector, [122](#)
- SerializeInventory
 - BeeGame::Serialization::Serialization, [160](#)
- SetAllItems
 - BeeGame::Inventory::Inventory, [45](#)
- SetBlock
 - BeeGame::Terrain::Chunks::Chunk, [65](#)
 - BeeGame::Terrain::LandGeneration::Terrain, [97](#)
 - BeeGame::Terrain::LandGeneration::Terrain↔
Generation, [102](#)
 - BeeGame::Terrain::LandGeneration::World, [88](#)
- SetBlocksUnmodified
 - BeeGame::Terrain::Chunks::Chunk, [66](#)
- SetInventorySize
 - BeeGame::Inventory::Inventory, [46](#)
- SetPlayerInventory
 - BeeGame::Inventory::Player_Inventory::Player↔
Inventory, [60](#)

- shareMeshes
 - BeeGame::Terrain::Chunks::MeshData, 72
- slotIndex
 - BeeGame::Inventory::InventorySlot, 56
- slots
 - BeeGame::Inventory::Inventory, 47
- speed
 - BeeGame::Player::PlayerLook, 114
 - BeeGame::Player::PlayerMove, 118
- SplitStack
 - BeeGame::Inventory::InventorySlot, 54
- Sprites
 - BeeGame::Resources::Resources, 130
- Start
 - BeeGame::Inventory::Player_Inventory::Player↔Inventory, 60
 - BeeGame::Items::ItemGameObject, 20
 - BeeGame::Player::PlayerLook, 112
 - BeeGame::Terrain::Chunks::Chunk, 66
 - BeeGame::Terrain::Chunks::LoadChunks, 77
 - BeeGame::Test, 163
- stoneBaseHeight
 - BeeGame::Terrain::LandGeneration::Terrain↔Generation, 103
- stoneBaseNoise
 - BeeGame::Terrain::LandGeneration::Terrain↔Generation, 104
- stoneBaseNoiseHeight
 - BeeGame::Terrain::LandGeneration::Terrain↔Generation, 104
- stoneMinHeight
 - BeeGame::Terrain::LandGeneration::Terrain↔Generation, 104
- stoneMountainFrequency
 - BeeGame::Terrain::LandGeneration::Terrain↔Generation, 104
- stoneMountainHeight
 - BeeGame::Terrain::LandGeneration::Terrain↔Generation, 104
- SwapItems
 - BeeGame::Inventory::InventorySlot, 54
- THVector2
 - BeeGame::Core::THVector2, 136, 137
- THVector3
 - BeeGame::Core::THVector3, 144, 145
- TexturePosition
 - BeeGame::Blocks::Bedrock, 33
 - BeeGame::Blocks::Dirt, 37
 - BeeGame::Blocks::Grass, 40
 - BeeGame::Items::Item, 17
- tileSize
 - BeeGame::Items::Item, 19
- timer
 - BeeGame::Terrain::Chunks::LoadChunks, 78
- ToString
 - BeeGame::Blocks::Air, 31
 - BeeGame::Blocks::Apiary, 36
 - BeeGame::Blocks::Bedrock, 34
 - BeeGame::Blocks::Block, 26
 - BeeGame::Blocks::Dirt, 38
 - BeeGame::Blocks::Grass, 40
 - BeeGame::Core::THVector2, 141
 - BeeGame::Core::THVector3, 154
 - BeeGame::Items::Item, 18
 - BeeGame::Terrain::ChunkWorldPos, 83
- tris
 - BeeGame::Terrain::Chunks::MeshData, 72
- Update
 - BeeGame::Inventory::InventorySlot, 54
 - BeeGame::Inventory::Player_Inventory::Player↔Inventory, 60
 - BeeGame::Player::PlayerLook, 113
 - BeeGame::Player::Selector, 121
 - BeeGame::Terrain::Chunks::Chunk, 66
 - BeeGame::Terrain::Chunks::LoadChunks, 77
- update
 - BeeGame::Terrain::Chunks::Chunk, 69
- UpdateBase
 - BeeGame::Inventory::Inventory, 46
- UpdateBlock
 - BeeGame::Blocks::Block, 26
 - BeeGame::Blocks::Grass, 41
- UpdateChunk
 - BeeGame::Terrain::Chunks::Chunk, 67
- updateCollisionMesh
 - BeeGame::Terrain::Chunks::Chunk, 69
- Updatelcon
 - BeeGame::Inventory::InventorySlot, 55
- UpdateIfEqual
 - BeeGame::Terrain::LandGeneration::World, 89
- UpdateSelector
 - BeeGame::Player::Selector, 121
- usesGameObject
 - BeeGame::Items::Item, 19
- uv
 - BeeGame::Terrain::Chunks::MeshData, 73
- VerticalJumpSpeed
 - BeeGame::Player::PlayerMove, 117
- verts
 - BeeGame::Terrain::Chunks::MeshData, 73
- world
 - BeeGame::Terrain::Chunks::Chunk, 69
 - BeeGame::Terrain::Chunks::LoadChunks, 78
 - BeeGame::Terrain::LandGeneration::Terrain, 98
- worldName
 - BeeGame::Serialization::Serialization, 161
- x
 - BeeGame::Core::THVector2, 142
 - BeeGame::Core::THVector3, 155
 - BeeGame::Items::Tile, 22
 - BeeGame::Terrain::ChunkWorldPos, 83
- xRot
 - BeeGame::Player::PlayerLook, 114

y

BeeGame::Core::THVector2, [142](#)

BeeGame::Core::THVector3, [155](#)

BeeGame::Items::Tile, [22](#)

BeeGame::Terrain::ChunkWorldPos, [83](#)

yRot

BeeGame::Player::PlayerLook, [114](#)

z

BeeGame::Core::THVector3, [155](#)

BeeGame::Terrain::ChunkWorldPos, [84](#)