Bee Game

Generated by Doxygen 1.8.13

Contents

1 Namespace Index

1.1 Packages

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

2 Hierarchical Index

2.1 Class Hierarchy

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

3 Class Index

3.1 Class List

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

4 File Index

4.1 File List

Here is a list of all files with brief descriptions:

5 Namespace Documentation

5.1 BeeGame Namespace Reference

Namespaces

- namespace Blocks
- namespace Core
- namespace Exceptipns
- namespace Inventory
- namespace Items
- namespace Player
- namespace Quest
- namespace Resources
- namespace Serialization
- namespace Terrain

Classes

- class LoadResources

  Loads all of the resources in the game
- class SpawnItem
- class Test

5.2 BeeGame.Blocks Namespace Reference

Classes

- class Air

  Air Block is an empty block that does not render and has no collider
- class Apiary

  Apiary Block
- class Bedrock

  Bedrock Block
- class Block

  Base class for blocks
- class Chest

  Chest Block
- class CraftingTable

  The Workbanch Block class
- class Dirt

  Dirt Block
- class Grass

  Grass Block
- class Leaves
- class Wood

5.3 BeeGame.Core Namespace Reference

Namespaces

- namespace Dictionarys
- namespace Enums
- namespace UnityTypeReplacements

Classes

- class Events
- class Extensions
- class THInput

  My implementation of the unity input system. Acts as a buffer layer to the unity system so that the input keys can be changed at runtime
- struct THVector2

  Serilializable version of Vector2
- struct THVector3

  Serializable version of Vector3

5.4 BeeGame.Core.Dictionarys Namespace Reference

Classes

- class BeeCombinationDictionaryEqualityComparer
- class BeeDictionarys
- class CraftingRecipies
- class PrefabDictionary

  The prefabs avaliable to the game
- class SpriteDictionary

  All of the sprites avaliable to the game

5.5 BeeGame.Core.Enums Namespace Reference

Enumerations

- enum HoneyCombType { HoneyCombType.HONEY, HoneyCombType.ICEY }

  Honey Comb Types
- enum BeeSpecies {
  BeeSpecies.FOREST, BeeSpecies.MEADOWS, BeeSpecies.TROPICAL, BeeSpecies.WINTRY,
  BeeSpecies.MODEST, BeeSpecies.MARSHY, BeeSpecies.ENDER, BeeSpecies.MONASTIC,
  BeeSpecies.STEADFAST, BeeSpecies.VALIANT, BeeSpecies.COMMON, BeeSpecies.CULTIVATED,
  BeeSpecies.DILIGENT, BeeSpecies.RURAL, BeeSpecies.FARMERLY, BeeSpecies.AGRARIAN,
  BeeSpecies.UNWEARY, BeeSpecies.INDUSTRIOUS, BeeSpecies.ICY, BeeSpecies.GLACIAL,
  BeeSpecies.NOBLE, BeeSpecies.IMPERIAL, BeeSpecies.MAJESTIC, BeeSpecies.MIRY,
  BeeSpecies.BOGGY, BeeSpecies.HERIOC, BeeSpecies.PHANTASMAL, BeeSpecies.SPECTRAL,
  BeeSpecies.HERMETIC, BeeSpecies.SECLUDED, BeeSpecies.SINISTER, BeeSpecies.FIENDISH,
  BeeSpecies.DEMONIC, BeeSpecies.FRUGAL, BeeSpecies.AUSTER, BeeSpecies.VINDICTIVE,
  BeeSpecies.EXOTIC, BeeSpecies.ENDEMIC, BeeSpecies.VENGEFUL, BeeSpecies.AVENGING,
  BeeSpecies.SETADFAST, BeeSpecies.HEROIC }

  The different possible bee Species
- enum BeeType { BeeType.QUEEN, BeeType.DRONE, BeeType.PRINCESS }

  The different bee types
- enum BeeTempPreferance {
  BeeTempPreferance.FROZEN, BeeTempPreferance.COLD, BeeTempPreferance.TEMPERATE, BeeTemp↩
  Preferance.HOT,
  BeeTempPreferance.HELL }

  The different bee temp preferences
- enum BeeLifeSpan {
  BeeLifeSpan.HUMMINGBIRD, BeeLifeSpan.SHORTEST, BeeLifeSpan.SHORT, BeeLifeSpan.NORMAL,
  BeeLifeSpan.LONG, BeeLifeSpan.LONGEST, BeeLifeSpan.SEATURTLE }

  The lifespan of the bee
- enum BeeProductionSpeed { BeeProductionSpeed.SLOW, BeeProductionSpeed.NORMAL, Bee↩
  ProductionSpeed.FAST }

  How fast the bee produces items
- enum BeeEffect { BeeEffect.NONE, BeeEffect.POSION }

  Any effects of the bee
- enum BeeHumidityPreferance {
  BeeHumidityPreferance.ARID, BeeHumidityPreferance.DRY, BeeHumidityPreferance.TEMPERATE, Bee↩
  HumidityPreferance.MOIST,
  BeeHumidityPreferance.HUMID }

  Humidity preferences of the bee
- enum Direction {
  Direction.NORTH, Direction.EAST, Direction.SOUTH, Direction.WEST,
  Direction.UP, Direction.DOWN }

  Direction in the game

5.5.1 Enumeration Type Documentation

5.5.1.1 BeeEffect

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

Any effects of the bee

Enumerator

NONE	
POSION	

Definition at line 55 of file Enums.cs.

```
        {
            NONE, POSION
        }
```

5.5.1.2 BeeHumidityPreferance

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

Humidity preferences of the bee

Enumerator

ARID	
DRY	
TEMPERATE	
MOIST	
HUMID	

Definition at line 63 of file Enums.cs.

```
        {
            ARID, DRY, TEMPERATE, MOIST, HUMID
        };
```

5.5.1.3 BeeLifeSpan

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

The lifespan of the bee

Enumerator

HUMMINGBIRD	
SHORTEST	
SHORT	
NORMAL	
LONG	
LONGEST	
SEATURTLE	

Definition at line 39 of file Enums.cs.

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

5.5.1.4 BeeProductionSpeed

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

How fast the bee produces items

Enumerator

SLOW	
NORMAL	
FAST	

Definition at line 47 of file Enums.cs.

```
    {
        SLOW, NORMAL, FAST
    };
```

5.5.1.5 BeeSpecies

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

The different possible bee Species

Enumerator

FOREST	
MEADOWS	
TROPICAL	
WINTRY	
MODEST	
MARSHY	
ENDER	
MONASTIC	
STEADFAST	
VALIANT	
COMMON	
CULTIVATED	
DILIGENT	
RURAL	
FARMERLY	
AGRARIAN	

Enumerator

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

Definition at line 15 of file Enums.cs.

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

5.5.1.6 BeeTempPreferance

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

The different bee temp preferences

Enumerator

FROZEN	
COLD	
TEMPERATE	
HOT	
HELL	

Definition at line 31 of file Enums.cs.

```
{
    FROZEN, COLD, TEMPERATE, HOT, HELL
};
```

5.5.1.7 BeeType

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

The different bee types

Enumerator

QUEEN	
DRONE	
PRINCESS	

Definition at line 23 of file Enums.cs.

```
{
    QUEEN, DRONE, PRINCESS
};
```

5.5.1.8 Direction

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

Direction in the game

Enumerator

NORTH	
EAST	
SOUTH	
WEST	
UP	
DOWN	

Definition at line 72 of file Enums.cs.

```
{
    NORTH, EAST, SOUTH, WEST, UP, DOWN
};
```

5.5.1.9 HoneyCombType

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

Honey Comb Types

Enumerator

HONEY	
ICEY	

Definition at line 6 of file Enums.cs.

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

5.6 BeeGame.Core.UnityTypeReplacements Namespace Reference

Classes

- struct THQuaternion

5.7 BeeGame.Exceptipns Namespace Reference

Classes

- class CraftingRecipieAdditionException
- class InputException

5.8 BeeGame.Inventory Namespace Reference

Namespaces

- namespace BlockInventory
- namespace Player_Inventory

Classes

- class ApiaryInventory

  Inventory for Apiarys Apiary
- class ChestInventory

  Incentory for the chests
- class Inventory

  Base class for all inventorys in the game
- class InventorySlot
- class ItemsInInventory

  Class that holds all of the items in the inventory. Can be serialized so inventory may be saved

5.9 BeeGame.Inventory.BlockInventory Namespace Reference

Classes

- class CraftingTableInventory

  Invnetory for the CraftingTable Block

5.10 BeeGame.Inventory.Player_Inventory Namespace Reference

Classes

- class PlayerInventory

  Controlls the player inventory

5.11 BeeGame.Items Namespace Reference

Classes

- class AbstractItem

  Does this need to exist?
- class ApplyColour

  Applies a given colour to a gameobject
- class Bee

  The bee item
- class HoneyComb

  Honey comb item produced by bees
- class Item

  Base class for all Items and Blocks in the game
- class ItemGameObject

  Interface between item and inity gameobjects
- class NormalBee
- class QueenBee
- struct Tile

  Position of the items texture

5.12 BeeGame.Player Namespace Reference

Classes

- class PlayerLook

  The look for the player
- class PlayerMove

  Moves the player
- class SavePlayerPosition

  Saves the player postion
- class Selector

  Moves the Block selector

5.13 BeeGame.Quest Namespace Reference

Classes

- class QuestBook

5.14 BeeGame.Resources Namespace Reference

Classes

- class Resources

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

5.15 BeeGame.Serialization Namespace Reference

Classes

- class Serialization

  Serializes and Deserialises things

5.16 BeeGame.Terrain Namespace Reference

Namespaces

- namespace Chunks
- namespace LandGeneration

Classes

- struct ChunkWorldPos

  Serializable int version of THVector3

5.17 BeeGame.Terrain.Chunks Namespace Reference

Classes

- class Chunk

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

  Loads the Chunks around the player
- class MeshData

  The data for a Chunks's Mesh
- class SaveChunk

  Saves a Chunks modified Blocks for save optimisation

5.18 BeeGame.Terrain.LandGeneration Namespace Reference

Namespaces

- namespace Noise

Classes

- class Terrain

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

  Generates the terrain for the game
- class World

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

5.19 BeeGame.Terrain.LandGeneration.Noise Namespace Reference

Classes

- class SimplexNoise

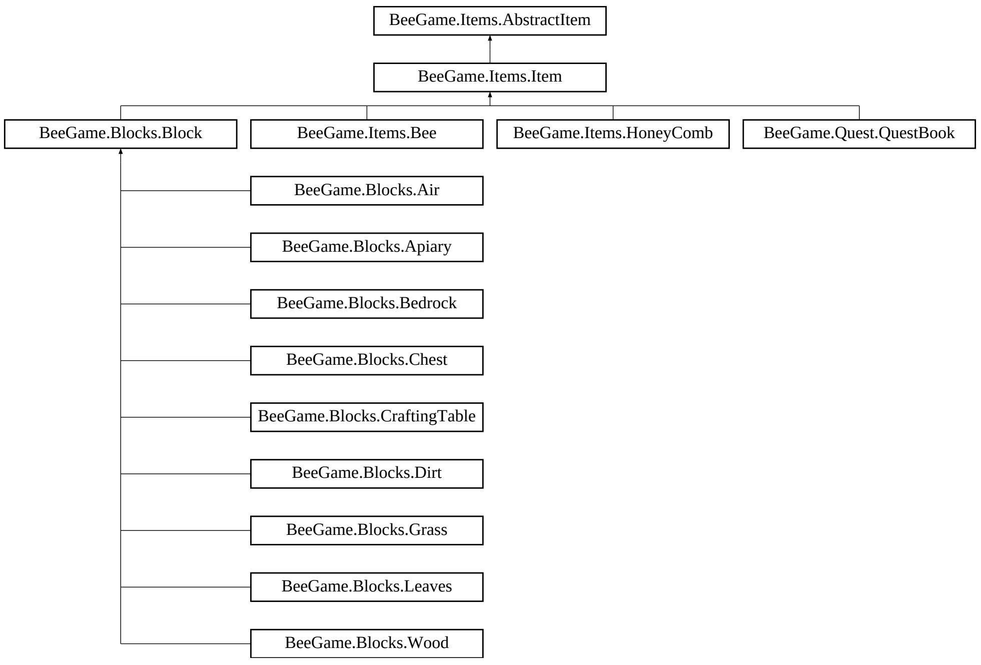
  Implementation of the Perlin simplex noise, an improved Perlin noise algorithm. Based loosely on SimplexNoise1234 by Stefan Gustavson `http://staffwww.itn.liu.se/~stegu/aqsis/aqsis-newnoise/`

6 Class Documentation

6.1 BeeGame.Items.AbstractItem Class Reference

Does this need to exist?

Inheritance diagram for BeeGame.Items.AbstractItem:



Public Member Functions

- abstract string GetItemName ()
- abstract string GetItemID ()
- abstract override int GetHashCode ()

6.1.1 Detailed Description

Does this need to exist?

Definition at line 12 of file AbstractItem.cs.

6.1.2 Member Function Documentation

6.1.2.1 GetHashCode()

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

Implemented in BeeGame.Items.Item, BeeGame.Items.Bee, BeeGame.Blocks.CraftingTable, BeeGame.Blocks.←
Block, BeeGame.Blocks.Chest, BeeGame.Blocks.Apiary, BeeGame.Items.HoneyComb, BeeGame.Blocks.Grass,
BeeGame.Blocks.Bedrock, BeeGame.Blocks.Air, BeeGame.Blocks.Dirt, BeeGame.Blocks.Leaves, BeeGame.←
Blocks.Wood, and BeeGame.Quest.QuestBook.

6.1.2.2 GetItemID()

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

Implemented in BeeGame.Items.Bee, BeeGame.Items.HoneyComb, and BeeGame.Items.Item.

6.1.2.3 GetItemName()

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

Implemented in BeeGame.Items.Item.

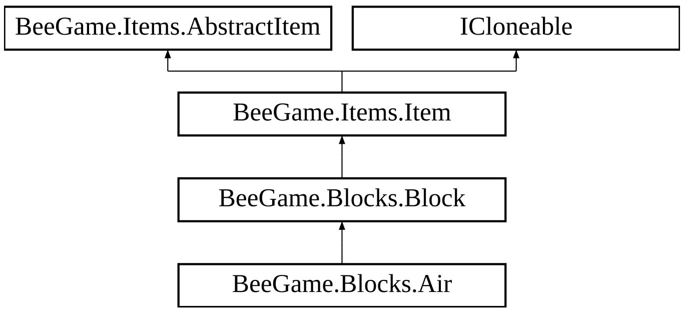
The documentation for this class was generated from the following file:

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

6.2 BeeGame.Blocks.Air Class Reference

Air Block is an empty block that does not render and has no collider

Inheritance diagram for BeeGame.Blocks.Air:



Public Member Functions

- Air ()
- override void BreakBlock (THVector3 pos)

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

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

  Hashcode acts as the base ID for an item
- override string ToString ()

  Gets the item name and ID in a nice format

Static Public Attributes

- static new int ID => 0

Additional Inherited Members

6.2.1 Detailed Description

Air Block is an empty block that does not render and has no collider

Definition at line 12 of file Air.cs.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 Air()

```
BeeGame.Blocks.Air.Air ( )
```

Definition at line 16 of file Air.cs.

```
                        : base("Air")
            {
            }
```

6.2.3 Member Function Documentation

6.2.3.1 BlockData()

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

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

Returns

Given MeshData

Reimplemented from BeeGame.Blocks.Block.

Definition at line 33 of file Air.cs.

```
            {
                return meshData;
            }
```

6.2.3.2 BreakBlock()

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

No item should be made when air is broken

Parameters

pos	position to spawn the Item

Reimplemented from BeeGame.Blocks.Block.

Definition at line 24 of file Air.cs.

```
        {
            return;
        }
```

6.2.3.3 GetHashCode()

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

Hashcode acts as the base ID for an item

Returns

2

Implements BeeGame.Items.AbstractItem.

Definition at line 52 of file Air.cs.

```
        {
            return ID;
        }
```

6.2.3.4 IsSolid()

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

Parameters

direction	Direction wanted to chesk solid

Returns

false

Reimplemented from BeeGame.Blocks.Block.

Definition at line 43 of file Air.cs.

```
        {
            return false;
        }
```

6.2.3.5 ToString()

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

Gets the item name and ID in a nice format

Returns

Definition at line 61 of file Air.cs.

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

6.2.4 Member Data Documentation

6.2.4.1 ID

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

Definition at line 14 of file Air.cs.

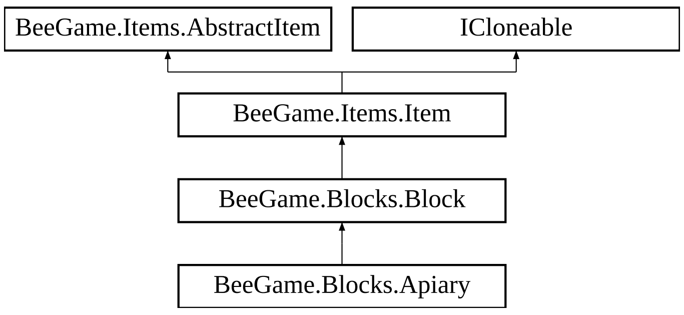
The documentation for this class was generated from the following file:

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

6.3 BeeGame.Blocks.Apiary Class Reference

Apiary Block

Inheritance diagram for BeeGame.Blocks.Apiary:



Public Member Functions

- Apiary ()

  Constructor
- override GameObject GetGameObject ()

  Gets the game object for this apiary
- override Tile TexturePosition (Direction direction)

  Returns the texture for the apiary Block
- override MeshData BlockData (Chunk chunk, int x, int y, int z, MeshData meshData, bool addToRender↵ Mesh=true)

  The data that this block adds to the mesh
- override void BreakBlock (THVector3 pos)

  Breaks the block
- override int GetHashCode ()

  ID of the item
- override string ToString ()

  The item name and ID as a string
- override bool InteractWithBlock (Inventory.Inventory inv)

  Toggles the ApiaryInventory for the block
- void MakeBees (Bee queen, ref Item[ ] inventory)

  Will make new Bee/Items from the given BeeType.QUEEN Bee
- Bee MakeBee (BeeType beeType, QueenBee queen)

  Nakes a new Bee
- BeeProductionSpeed CombineProductionSpeed (BeeProductionSpeed b1, BeeProductionSpeed b2)

  Combines the BeeProductionSpeed of the given BeeProductionSpeed

Public Attributes

- int mutationMultiplyer

Static Public Attributes

- static new int ID => 10

Private Member Functions

- BeeSpecies CombineSpecies (BeeSpecies s1, BeeSpecies s2)

  Returns a BeeSpecies depending on the given BeeSpecies
- float Rand (float[ ] weights)

  Returns a random float bewteen 0 and the sum of weights rounded to 2dp
- BeeLifeSpan CombineLifespan (BeeLifeSpan b1, BeeLifeSpan b2)

  Combines the BeeLifeSpan of the given BeeLifeSpan
- uint CombineFertility (uint b1, uint b2)

  Combines the fertility of the given fertility
- BeeEffect CombineEffect (BeeEffect b1, BeeEffect b2)

  Combines the BeeEffect of the given BeeEffect
- int ReturnChange (int b1, int b2, int maxChange, int minChange=0)

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

Private Attributes

- GameObject myGameobject

Additional Inherited Members

6.3.1 Detailed Description

Apiary Block

Definition at line 17 of file Apiary.cs.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 Apiary()

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

Constructor

Definition at line 30 of file Apiary.cs.

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

6.3.3 Member Function Documentation

6.3.3.1 BlockData()

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

The data that this block adds to the mesh

Parameters

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

Returns

Given *meshData* with this blocks data added to it

Only adds to the colision mesh as the model is handlled by the unity prefab system

Reimplemented from BeeGame.Blocks.Block.

Definition at line 72 of file Apiary.cs.

```
        {
            if (myGameobject == null)
            {
                myGameobject = UnityEngine.Object.Instantiate(
    PrefabDictionary.GetPrefab("Apiary"), new THVector3(x, y, z) + chunk.
    chunkWorldPos, Quaternion.identity, chunk.transform);
                myGameobject.GetComponent<ChestInventory>().inventoryPosition =
    new THVector3(x, y, z) + chunk.chunkWorldPos;
                myGameobject.GetComponent<ChestInventory>().SetChestInventory();
            }
            return base.BlockData(chunk, x, y, z, meshData, true);
        }
```

6.3.3.2 BreakBlock()

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

Breaks the block

Parameters

pos	Position of the block

Reimplemented from BeeGame.Blocks.Block.

Definition at line 87 of file Apiary.cs.

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

6.3.3.3 CombineEffect()

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

Combines the BeeEffect of the given BeeEffect

Parameters

b1	Fist BeeEffect
b2	Second BeeEffect

Returns

A new BeeEffect

Definition at line 308 of file Apiary.cs.

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

6.3.3.4 CombineFertility()

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

Combines the fertility of the given fertility

Parameters

b1	Fist Bees fertility
b2	Second Bees fertility

Returns

A new fertility, uint

Definition at line 297 of file Apiary.cs.

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

6.3.3.5 CombineLifespan()

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

Combines the BeeLifeSpan of the given BeeLifeSpan

Parameters

b1	Fist BeeLifeSpan
b2	Second BeeLifeSpan

Returns

A new BeeLifeSpan

Definition at line 286 of file Apiary.cs.

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

6.3.3.6 CombineProductionSpeed()

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

Combines the BeeProductionSpeed of the given BeeProductionSpeed

Parameters

b1	Fist BeeProductionSpeed
b2	Second BeeProductionSpeed

Returns

A new BeeProductionSpeed

Definition at line 319 of file Apiary.cs.

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

6.3.3.7 CombineSpecies()

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

Returns a BeeSpecies depending on the given BeeSpecies

Parameters

s1	First BeeSpecies
s2	Second BeeSpecies

Returns

A new BeeSpecies

Definition at line 239 of file Apiary.cs.

```
        {
            BeeSpecies[] possibleSpecies = BeeDictionarys.
      GetCombinations(s1, s2);
            float[] weights = possibleSpecies.Length > 2 ? BeeDictionarys.
      GetWeights(possibleSpecies) : new float[] { 0.5f, 0.5f };

            var randomNum = Rand(weights);
            var weightsSum = 0f;

            //* when the rumber generated is less than the current sum of the weights return that bee
            for (int i = 0; i < weights.Length; i++)
            {
                if(randomNum <= weightsSum)
                {
                    return possibleSpecies[i];
                }

                weightsSum += weights[i];
            }

            //* if for some reason the weights cannot work return the first bee in the combination list
            return possibleSpecies[0];
        }
```

6.3.3.8 GetGameObject()

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

Gets the game object for this apiary

Returns

THe chest game object

Reimplemented from BeeGame.Items.Item.

Definition at line 41 of file Apiary.cs.

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

6.3.3.9 GetHashCode()

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

ID of the item

Returns

3

Implements BeeGame.Items.AbstractItem.

Definition at line 102 of file Apiary.cs.

```
        {
            return ID;
        }
```

6.3.3.10 InteractWithBlock()

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

Toggles the ApiaryInventory for the block

Parameters

inv	

Returns

Definition at line 122 of file Apiary.cs.

```
        {
            myGameobject.GetComponent<ApiaryInventory>().myblock = this;
            myGameobject.GetComponent<ApiaryInventory>().ToggleInventory(inv);
            return true;
        }
```

6.3.3.11 MakeBee()

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

Nakes a new Bee

Parameters

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

Returns

A new Bee

Definition at line 208 of file Apiary.cs.

```
        {
            //* gives all of the primary and secondary stats to the bee
            NormalBee nb = new NormalBee()
            {
                pSpecies = CombineSpecies(queen.queen.sSpecies, queen.drone.sSpecies),
                sSpecies = CombineSpecies(queen.queen.sSpecies, queen.drone.sSpecies),

                pEffect = CombineEffect(queen.queen.sEffect, queen.drone.sEffect),
                sEffect = CombineEffect(queen.queen.sEffect, queen.drone.sEffect),

                pFertility = CombineFertility(queen.queen.sFertility, queen.drone.sFertility),
                sFertility = CombineFertility(queen.queen.sFertility, queen.drone.sFertility),

                pLifespan = CombineLifespan(queen.queen.sLifespan, queen.drone.sLifespan),
                sLifespan = CombineLifespan(queen.queen.sLifespan, queen.drone.sLifespan),

                pProdSpeed = CombineProductionSpeed(queen.queen.sProdSpeed, queen.drone.sProdSpeed),
                sProdSpeed = CombineProductionSpeed(queen.queen.sProdSpeed, queen.drone.sProdSpeed)
            };

            //* returns the new bee
            return new Bee(beeType, nb);
        }
```

6.3.3.12 MakeBees()

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

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

Parameters

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

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

Definition at line 138 of file Apiary.cs.

```
        {
            Item[] producedItems = new Item[9];

            //* will always return a new princess and drone
            producedItems[0] = MakeBee(BeeType.PRINCESS, queen.queenBee);
            producedItems[1] = MakeBee(BeeType.DRONE, queen.queenBee);

            var repeats = UnityEngine.Random.Range(0, queen.queenBee.queen.pFertility);

            //* produces as many other children as the bee staats will allow
            for (int i = 0; i < repeats; i++)
            {
                producedItems[i + 2] = MakeBee(queen.queenBee.queen.pFertility > 6 ? (BeeType)UnityEngine.Random.Range(1, 3) : BeeType.DRONE, queen.queenBee);

                if (producedItems[i + 2] is Bee b && b.beeType != BeeType.PRINCESS)
                    producedItems[i + 2].itemStackCount = UnityEngine.Random.Range(1, (int)queen.queenBee.queen.pFertility + 1);
            }

            //* gets the produced items
            var beeProduce = BeeDictionarys.GetBeeProduce(queen.queenBee.queen.pSpecies);

            //* chnages the stack count of the produced items to the correct number
            for (int i = 0; i < beeProduce.Length; i++)
            {
                beeProduce[i].itemStackCount += UnityEngine.Random.Range(1, (int)queen.queenBee.queen.sProdSpeed + 1);
            }

            //* adds the itmes that the bee species produces into the procued item array
            for (int i = (int)queen.queenBee.queen.pFertility + 2, prod = 0; prod < beeProduce.Length; i++, prod++)
            {
                producedItems[i] = beeProduce[prod];
            }

            //* puts the items into the inventory
            for (int i = 0; i < 9; i++)
            {
                if (inventory[i + 2] != null)
                {
                    //* if the slot has the same item in it and it wont be more than the max stack ount but the new item into it
                    if (producedItems[i] == inventory[i + 2] && inventory[i + 2].itemStackCount + 1 <= inventory[i + 2].maxStackCount)
                        inventory[i + 2].itemStackCount++;
                    else
                        //* otherwise find a new slot to put the item into
                        for (int j = i; j < (9 - i); j++)
                        {
                            if (inventory[j + 2] == null)
                            {
                                inventory[j + 2] = producedItems[i];
                                break;
                            }
                            else if (producedItems[i] == inventory[j + 2] && inventory[j + 2].itemStackCount + 1 <= inventory[j + 2].maxStackCount)
                            {
                                inventory[j + 2].itemStackCount++;
                                break;
                            }
                        }
                }
                //* if the slot is empty put the item into it
                else
                    inventory[i + 2] = producedItems[i];
            }
        }
```

6.3.3.13 Rand()

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

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

Parameters

weights	The weights

Returns

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

Definition at line 267 of file Apiary.cs.

```
        {
            var totalWeights = 0f;

            //* sums the weights
            for (int i = 0; i < weights.Length; i++)
            {
                totalWeights += weights[i];
            }

            return (float)Math.Round(UnityEngine.Random.Range(0, totalWeights), 2);
        }
```

6.3.3.14 ReturnChange()

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

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

Parameters

b1	First number
b2	Second number
maxChange	Max return value
minChange	Min return value

Returns

A number between *maxChange* and *minChange*

If *b1* and *b2* are the same their is still a chance of change due to this function also takeing mutationMultiplyer, the value of wich is dictated by the apairy

Definition at line 335 of file Apiary.cs.

```
        {
            //* b1 and b2 are checked for which one is bigger than the other here as the
            //* queen my have a lower stat the an the drone and the drone is always passed in second
            var change = UnityEngine.Random.Range(b1 < b2 ? b1 : b2, (b2 > b1 ? b2 : b1) + 2);

            //* this will make it possible for the bees to mutate during combination of the stats are the
     same
            //* it will also cause more random mutation more mimicing nature
            change += UnityEngine.Random.Range(-mutationMultiplyer,
    mutationMultiplyer);

            //* as all but on ef the stats are enums they have a min/max value so need to check that this
     is not exceded
            if (change > maxChange)
                change = maxChange;
            else if (minChange > change)
                change = minChange;

            return change;

        }
```

6.3.3.15 TexturePosition()

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

Returns the texture for the apiary Block

Parameters

direction	Direction of thhe desired face

Returns

Tile with the textture coordinates of the Block texture

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

Reimplemented from BeeGame.Items.Item.

Definition at line 54 of file Apiary.cs.

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

6.3.3.16 ToString()

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

The item name and ID as a string

Returns

A nicely formatted string

Definition at line 111 of file Apiary.cs.

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

6.3.4 Member Data Documentation

6.3.4.1 ID

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

Definition at line 24 of file Apiary.cs.

6.3.4.2 mutationMultiplyer

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

Definition at line 22 of file Apiary.cs.

6.3.4.3 myGameobject

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

Definition at line 20 of file Apiary.cs.

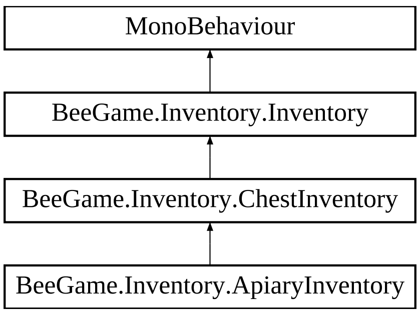
The documentation for this class was generated from the following file:

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

6.4 BeeGame.Inventory.ApiaryInventory Class Reference

Inventory for Apiarys Apiary

Inheritance diagram for BeeGame.Inventory.ApiaryInventory:


MonoBehaviour
BeeGame.Inventory.Inventory
BeeGame.Inventory.ChestInventory
BeeGame.Inventory.ApiaryInventory

Public Member Functions

- override void SetChestInventory (string invName="Apiary")

  Sets the size and name of this Inventory

Public Attributes

- float combinationTime = 0

  How long does the current combineing bee have left
- Slider timerSlideer

  Sider to give a visual indication of combinationTime

Private Member Functions

- void Update ()

  Updates the block every frame
- void FixedUpdate ()

  Updates the combination time because of this was frame rate dependand weird things would happen
- void CheckforBees ()

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

Private Attributes

- bool beesCombineing

  Are bees currently combineing

Additional Inherited Members

6.4.1 Detailed Description

Inventory for Apiarys Apiary

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

Definition at line 13 of file ApiaryInventory.cs.

6.4.2 Member Function Documentation

6.4.2.1 CheckforBees()

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

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

Definition at line 73 of file ApiaryInventory.cs.

```
        {
            Items.Item posOneItem = items.itemsInInventory[0];
            Items.Item posTwoItem = items.itemsInInventory[1];

            //* the item is checkd if it is a bee and if it is then a new variable is made for convenience
            //* if it is a queen then just set the combination time and go
            if (posOneItem is Items.Bee b && b.beeType == Core.Enums.BeeType.QUEEN)
            {
                combinationTime = ((float)b.queenBee.queen.pLifespan + 1) * 2;
                beesCombineing = true;
                SaveInv();

                timerSlideer.maxValue = combinationTime;

                return;
            }

            //* of one bee is a princess and another is a drone in the correct slots combine them
            if(posOneItem is Items.Bee b1 && posTwoItem is Items.Bee b2 && b1.beeType == Core.Enums.BeeType
      .PRINCESS && b2.beeType == Core.Enums.BeeType.DRONE)
            {
                //* comvert the princess to a queen with the paired drone
                Items.Bee.ConvertToQueen(ref b1, b2.normalBee);

                //* reduce number of drones in slot by 1 and check it is a valid stack number
                items.itemsInInventory[1].itemStackCount -= 1;
                slots[0].item = b1;

                if (items.itemsInInventory[1].itemStackCount <= 0)
                    items.itemsInInventory[1] = null;

                //* set the combination time
                combinationTime = ((float)b1.queenBee.queen.pLifespan + 1) * 2;
                beesCombineing = true;

                SaveInv();

                //* set the slider max to the combination time
                timerSlideer.maxValue = combinationTime;
            }
        }
```

6.4.2.2 FixedUpdate()

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

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

Definition at line 61 of file ApiaryInventory.cs.

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

6.4.2.3 SetChestInventory()

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

Sets the size and name of this Inventory

Parameters

invName	

Reimplemented from BeeGame.Inventory.ChestInventory.

Definition at line 121 of file ApiaryInventory.cs.

```
        {
            base.SetChestInventory("Apiary" );
        }
```

6.4.2.4 Update()

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

Updates the block every frame

Definition at line 36 of file ApiaryInventory.cs.

```
        {
            //* Updates the base class as unity Update function does not run on parent classes
            UpdateChestInventory();

            //* if the apiary is not an item on the ground and bees are not currently combineing check is
             bees should be combineing
            if(items.itemsInInventory.Length > 0 && !
      beesCombineing)
                CheckforBees();

            //* if the currently combineing bees has finished combineing
            if (combinationTime < 0 && beesCombineing)
            {
                //* make the items that the bees should make and destroy the spent queen
                ((Apiary)myblock).MakeBees(items.
      itemsInInventory[0] as Items.Bee, ref items.
      itemsInInventory);
                beesCombineing = false;
                items.itemsInInventory[0] = null;

                //* save the channges to the inventory
                SaveInv();
            }
        }
```

6.4.3 Member Data Documentation

6.4.3.1 beesCombineing

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

Are bees currently combineing

Definition at line 19 of file ApiaryInventory.cs.

6.4.3.2 combinationTime

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

How long does the current combineing bee have left

Definition at line 24 of file ApiaryInventory.cs.

6.4.3.3 timerSlideer

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

Sider to give a visual indication of combinationTime

Definition at line 29 of file ApiaryInventory.cs.

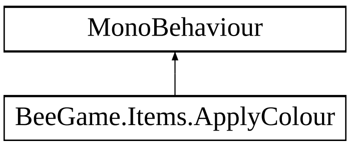
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/BlockInventory/Apiary↩ Inventory.cs

6.5 BeeGame.Items.ApplyColour Class Reference

Applies a given colour to a gameobject

Inheritance diagram for BeeGame.Items.ApplyColour:



Public Attributes

- Color colour

    Colour to apply
- GameObject [ ] objects

    Objects to apply the colour to

Private Member Functions

- void Start ()

    Applies the colour to the GameObjects in the objects array

6.5.1 Detailed Description

Applies a given colour to a gameobject

Definition at line 12 of file ApplyColour.cs.

6.5.2 Member Function Documentation

6.5.2.1 Start()

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

Applies the colour to the GameObjects in the objects array

Definition at line 32 of file ApplyColour.cs.

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

6.5.3 Member Data Documentation

6.5.3.1 colour

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

Colour to apply

Definition at line 18 of file ApplyColour.cs.

6.5.3.2 objects

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

Objects to apply the colour to

Array set in the editor

Definition at line 25 of file ApplyColour.cs.

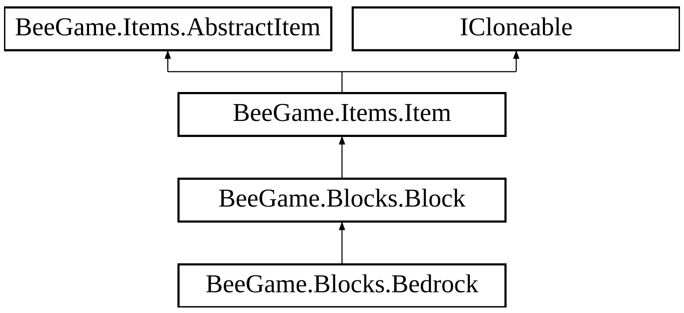
The documentation for this class was generated from the following file:

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

6.6 BeeGame.Blocks.Bedrock Class Reference

Bedrock Block

Inheritance diagram for BeeGame.Blocks.Bedrock:



Public Member Functions

- Bedrock ()

    Constructor
- override void BreakBlock (THVector3 pos)

    The block cannot be broken so nothing is done
- override Tile TexturePosition (Direction direction)

    Position if te bedrock texture in the atlas
- override int GetHashCode ()

    Returns the ID of the item
- override string ToString ()

    The item name and ID as a string

Static Public Attributes

- static new int ID => -1

Additional Inherited Members

6.6.1 Detailed Description

Bedrock Block

Definition at line 12 of file Bedrock.cs.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 Bedrock()

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

Constructor

Definition at line 22 of file Bedrock.cs.

```
                            : base("Bedrock")
            {
                breakable = false;
            }
```

6.6.3 Member Function Documentation

6.6.3.1 BreakBlock()

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

The block cannot be broken so nothing is done

Parameters

pos	positon of the block

Reimplemented from BeeGame.Blocks.Block.

Definition at line 33 of file Bedrock.cs.

```
            {
                return;
            }
```

6.6.3.2 GetHashCode()

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

Returns the ID of the item

Returns

-1

Implements BeeGame.Items.AbstractItem.

Definition at line 56 of file Bedrock.cs.

```
            {
                return ID;
            }
```

6.6.3.3 TexturePosition()

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

Position if te bedrock texture in the atlas

Parameters

direction	Direction

Returns

Position in the texture atlas

Reimplemented from BeeGame.Items.Item.

Definition at line 45 of file Bedrock.cs.

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

6.6.3.4 ToString()

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

The item name and ID as a string

Returns

A nicely formatted string

Definition at line 65 of file Bedrock.cs.

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

6.6.4 Member Data Documentation

6.6.4.1 ID

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

Definition at line 15 of file Bedrock.cs.

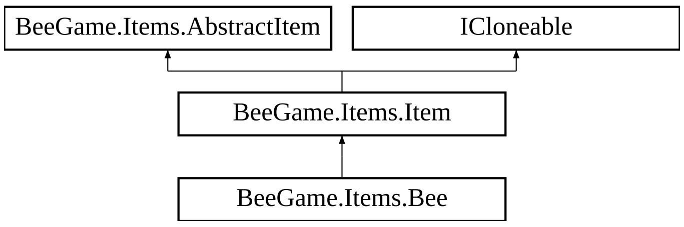
The documentation for this class was generated from the following file:

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

6.7 BeeGame.Items.Bee Class Reference

The bee item

Inheritance diagram for BeeGame.Items.Bee:



Public Member Functions

- Bee ()
- Bee (BeeType beeType, NormalBee normalBee)

  Create a bee from NormalBee
- Bee (BeeType beeType, QueenBee queenBee)

  Create a bee from QueenBee
- override Sprite GetItemSprite ()

  Returns the sprite for this, of the correct colour
- override string GetItemID ()

  Makes the item ID. For this it is the Normal ID \ the int value of the queenBee.GetHashCode() or normalBee.Get↩ HashCode() as a string
- Bee MakeBeeWithStats (BeeType beeType=BeeType.DRONE, BeeSpecies species=BeeSpecies.FORE↩ ST, BeeLifeSpan lifespan=BeeLifeSpan.NORMAL, uint fertility=2, BeeEffect effect=BeeEffect.NONE, Bee↩ ProductionSpeed prodSpeed=BeeProductionSpeed.NORMAL)

  Make a bee with given stats
- override int GetHashCode ()

  Retuens the hashcode for this Item

Static Public Member Functions

- static void ConvertToQueen (Bee princess, NormalBee drone)

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

  Will Convert this bee into a BeeType.QUEEN Bee

Public Attributes

- bool canSeeBeeData = false

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

Static Public Attributes

- static new int ID => 11

Properties

- BeeType beeType `[get, set]`

  This bees BeeType
- BeeType previousBeeType `[get, set]`

  What was this bees BeeType?
- override int maxStackCount `[get]`

  Overrided so can be set
- QueenBee queenBee `[get, set]`

  If this bee is a BeeType.QUEEN this will be not null
- NormalBee normalBee `[get, set]`

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

Private Attributes

- int maxStack = 64
- Sprite itemSprite

  This bees Sprite

Additional Inherited Members

6.7.1 Detailed Description

The bee item

Definition at line 14 of file Bee.cs.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 Bee() `[1/3]`

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

Definition at line 58 of file Bee.cs.

```
        {
            normalBee = new NormalBee();
        }
```

6.7.2.2 Bee() `[2/3]`

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

Create a bee from NormalBee

Parameters

beeType	BeeType of the bee
normalBee	NormalBee data

Definition at line 69 of file Bee.cs.

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

6.7.2.3 Bee() [3/3]

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

Create a bee from QueenBee

Parameters

beeType	BeeType of the bee
normalBee	QueenBee data

Definition at line 82 of file Bee.cs.

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

6.7.3 Member Function Documentation

6.7.3.1 ConvertToQueen() [1/2]

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

Will convery this bee to a BeeType.QUEEN useing this bees stats as the BeeType.PRINCESS stats

Parameters

drone	

Definition at line 142 of file Bee.cs.

```
        {
            ConvertToQueen(ref princess, drone);
        }
```

6.7.3.2 ConvertToQueen() [2/2]

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

Will Convert this bee into a BeeType.QUEEN Bee

Parameters

princess	The BeeType.PRINCESS Stats
drone	The BeeType.DRONE

Definition at line 152 of file Bee.cs.

```
        {
            princess.beeType = BeeType.QUEEN;
            princess.queenBee = new QueenBee(princess.normalBee, drone);
            princess.normalBee = null;

            princess.itemName = new CultureInfo("en-US", false).TextInfo.ToTitleCase($"
     {princess.queenBee.queen.pSpecies} {princess.beeType}".ToLower());
        }
```

6.7.3.3 GetHashCode()

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

Retuens the hashcode for this Item

Returns

9

Implements BeeGame.Items.AbstractItem.

Definition at line 202 of file Bee.cs.

```
        {
            return ID;
        }
```

6.7.3.4 GetItemID()

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

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

Returns

Item ID as a string

Implements BeeGame.Items.AbstractItem.

Definition at line 131 of file Bee.cs.

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

6.7.3.5 GetItemSprite()

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

Returns the sprite for this, of the correct colour

Returns

Sprite

Reimplemented from BeeGame.Items.Item.

Definition at line 96 of file Bee.cs.

```
        {
            //* if the bee has not change in any way dont rebuild the sprite as that takes time
            if(previousBeeType == beeType && itemSprite != null)
            {
                return itemSprite;
            }

            previousBeeType = beeType;

            //* set the correct sprite and colour
            if (beeType == BeeType.QUEEN)
            {
                //* avoids the crown, black body, yellow body, and both colours of the wings
                Color[] colorsToAvoid = { new Color(0, 0, 0), new Color(232f, 200f, 42f, 255f) / 255f, new Color(232f, 213f, 106f, 255f) / 255f, new Color(156f, 146f, 130f, 255f) / 255f, new Color(225f, 223f, 219f, 255f) / 255f };
                return itemSprite = SpriteDictionary.GetSprite("Queen").ColourSprite(BeeDictionarys.GetBeeColour((BeeSpecies)(queenBee?.queen.pSpecies)), coloursToAvoid: colorsToAvoid);
            }
            else if (beeType == BeeType.PRINCESS)
            {
                //* avoids the tiara, black body, yellow body, and both colours of the wings
                Color[] colorsToAvoid = { new Color(0, 0, 0), new Color(191f, 195f, 45f, 255f) / 255f, new Color(191f, 195f, 44f, 255f) / 255f, new Color(156f, 146f, 130f, 255f) / 255f, new Color(225f, 223f, 219f, 255f) / 255f, new Color(232f, 200, 42, 255f) / 255f };
                return itemSprite = SpriteDictionary.GetSprite("Princess").ColourSprite(BeeDictionarys.GetBeeColour((BeeSpecies)(normalBee?.pSpecies)), coloursToAvoid: colorsToAvoid);
            }
            else
            {
                //* avoids the block body, yellow body, and both wing colours
                Color[] colorsToAvoid = { new Color(0, 0, 0), new Color(156f, 146f, 130f, 255f) / 255f, new Color(225f, 223f, 219f, 255f) / 255f, new Color(232f, 200, 42, 255f) / 255f };
                return itemSprite = SpriteDictionary.GetSprite("Drone").ColourSprite(BeeDictionarys.GetBeeColour((BeeSpecies)normalBee?.pSpecies), coloursToAvoid: colorsToAvoid);
            }
        }
```

6.7.3.6 MakeBeeWithStats()

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

Make a bee with given stats

Parameters

beeType	BeeType
species	BeeSpecies
lifespan	BeeLifeSpan
fertility	1 or greater
effect	BeeEffect
prodSpeed	BeeProductionSpeed

Returns

A Bee with the given stats

Definition at line 171 of file Bee.cs.

```
        {
            NormalBee normBee = new NormalBee()
            {
                pSpecies = species,
                pLifespan = lifespan,
                pFertility = fertility,
                pProdSpeed = prodSpeed,
                pEffect = effect,
                sEffect = effect,
                sFertility = fertility,
                sLifespan = lifespan,
                sProdSpeed = prodSpeed,
                sSpecies = species
            };

            switch (beeType)
            {
                case BeeType.QUEEN:
                    return new Bee(beeType, new QueenBee(normBee, normBee));
                default:
                    return new Bee(beeType, normBee);
            }
        }
```

6.7.4 Member Data Documentation

6.7.4.1 canSeeBeeData

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

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

Definition at line 20 of file Bee.cs.

6.7.4.2 ID

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

Definition at line 54 of file Bee.cs.

6.7.4.3 itemSprite

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

This bees Sprite

Definition at line 40 of file Bee.cs.

6.7.4.4 maxStack

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

Definition at line 34 of file Bee.cs.

6.7.5 Property Documentation

6.7.5.1 beeType

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

This bees BeeType

Definition at line 25 of file Bee.cs.

6.7.5.2 maxStackCount

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

Overrided so can be set

Definition at line 33 of file Bee.cs.

6.7.5.3 normalBee

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

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

Definition at line 52 of file Bee.cs.

6.7.5.4 previousBeeType

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

What was this bees BeeType?

Definition at line 29 of file Bee.cs.

6.7.5.5 queenBee

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

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

Possibly change this to an array to 2 NormalBees

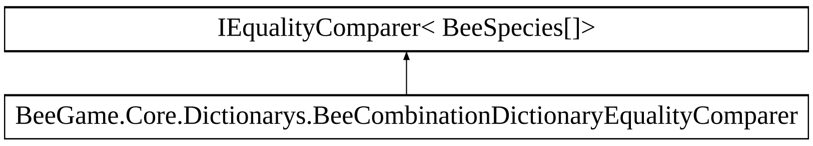
Definition at line 48 of file Bee.cs.

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

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

6.8 BeeGame.Core.Dictionarys.BeeCombinationDictionaryEqualityComparer Class Reference

Inheritance diagram for BeeGame.Core.Dictionarys.BeeCombinationDictionaryEqualityComparer:


IEqualityComparer< BeeSpecies[]>
BeeGame.Core.Dictionarys.BeeCombinationDictionaryEqualityComparer

Public Member Functions

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

6.8.1 Detailed Description

Definition at line 9 of file EqualityComperors.cs.

6.8.2 Member Function Documentation

6.8.2.1 Equals()

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

Definition at line 11 of file EqualityComperors.cs.

```
        {
            if (x.Contains(y[0]) && x.Contains(y[1]))
            {
                //* if the x length is greater than 2 this means that the combination can have duplicate bees for a product
                if (x.Length > 2)
                    return true;

                //* if 1 means both y elements are the same so no combination has been found
                if(y.Intersect(x).Count() <= 1)
                    return false;

                return true;
            }

            return false;
        }
```

6.8.2.2 GetHashCode()

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

Definition at line 29 of file EqualityComperors.cs.

```
        {
            unchecked
            {
                int hashcode = 13;

                for (int i = 0; i < obj.Length; i++)
                {
                    hashcode += (int)obj[i];
                }

                return hashcode;
            }
        }
```

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

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/EqualityComperors.cs

6.9 BeeGame.Core.Dictionarys.BeeDictionarys Class Reference

Static Public Member Functions

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

  Returns colour if the given honey coumb

Static Public Attributes

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

Static Private Member Functions

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

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

Static Private Attributes

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

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

6.9.1 Detailed Description

Definition at line 9 of file BeeDictionarys.cs.

6.9.2 Member Function Documentation

6.9.2.1 CombColour()

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

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

Parameters

r	Red
g	Green
b	Blue
a	Alpha, Default no alpha

Returns

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

Definition at line 118 of file BeeDictionarys.cs.

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

6.9.2.2 GetBeeColour()

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

Definition at line 91 of file BeeDictionarys.cs.

```
        {
            beeColour.TryGetValue(species, out Color colour);

            return colour != null ? colour : new Color();
        }
```

6.9.2.3 GetBeeProduce()

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

Definition at line 75 of file BeeDictionarys.cs.

```
        {
            beeProduce.TryGetValue(species, out Items.Item[] produce);

            //* of the produce cant be found then return a honey comb as it is probly a bug
            return produce ?? new Items.Item[1] { new Items.HoneyComb(
      HoneyCombType.HONEY) };
        }
```

6.9.2.4 GetCombColour()

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

Returns colour if the given honey coumb

Parameters

type	Type of the comb

Returns

The Color of the comb and a new Color.red if the given HoneyCombType does not exists as a key in the honeyCoumbColour dictionary

Definition at line 128 of file BeeDictionarys.cs.

```
        {
            honeyCoumbColour.TryGetValue(type, out var temp);

            if (temp == null)
                return new Color(1, 0, 0);

            return temp;
        }
```

6.9.2.5 GetCombinations()

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

Definition at line 40 of file BeeDictionarys.cs.

```
        {
            var beeSpecies = new BeeSpecies[2] { s1, s2 };
            var returnBeeList = new List<BeeSpecies>();

            var keys = beeCombinations.Keys.ToArray();
            var comparor = new BeeCombinationDictionaryEqualityComparer
    ();

            for (int i = 0; i < keys.Length; i++)
            {
                if(comparor.Equals(keys[i], beeSpecies))
                {
                    var temp = beeCombinations[keys[i]];

                    for (int j = 0; j < temp.Length; j++)
                    {
                        returnBeeList.Add(temp[i]);
                    }
                }
            }

            returnBeeList.Add(s1);
            returnBeeList.Add(s2);

            return returnBeeList.ToArray();
        }
```

6.9.2.6 GetWeights()

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

Definition at line 18 of file BeeDictionarys.cs.

```
        {
            var returnArray = new float[species.Length];

            for (int i = 0; i < species.Length; i++)
            {
                if(beeCombinationWeights.ContainsKey(species[i]))
                    returnArray[i] = beeCombinationWeights[species[i]];
                else
                    returnArray[i] = 0.5f;
            }

            return returnArray;
        }
```

6.9.3 Member Data Documentation

6.9.3.1 beeColour

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

Initial value:

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

Definition at line 85 of file BeeDictionarys.cs.

6.9.3.2 beeCombinations

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

Initial value:

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

Definition at line 35 of file BeeDictionarys.cs.

6.9.3.3 beeCombinationWeights

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

Initial value:

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

Definition at line 12 of file BeeDictionarys.cs.

6.9.3.4 beeProduce

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

Initial value:

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

Definition at line 69 of file BeeDictionarys.cs.

6.9.3.5 honeyCoumbColour

`Dictionary<HoneyCombType, Color> BeeGame.Core.Dictionarys.BeeDictionarys.honeyCoumbColour [static], [private]`

Initial value:

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

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

Definition at line 103 of file BeeDictionarys.cs.

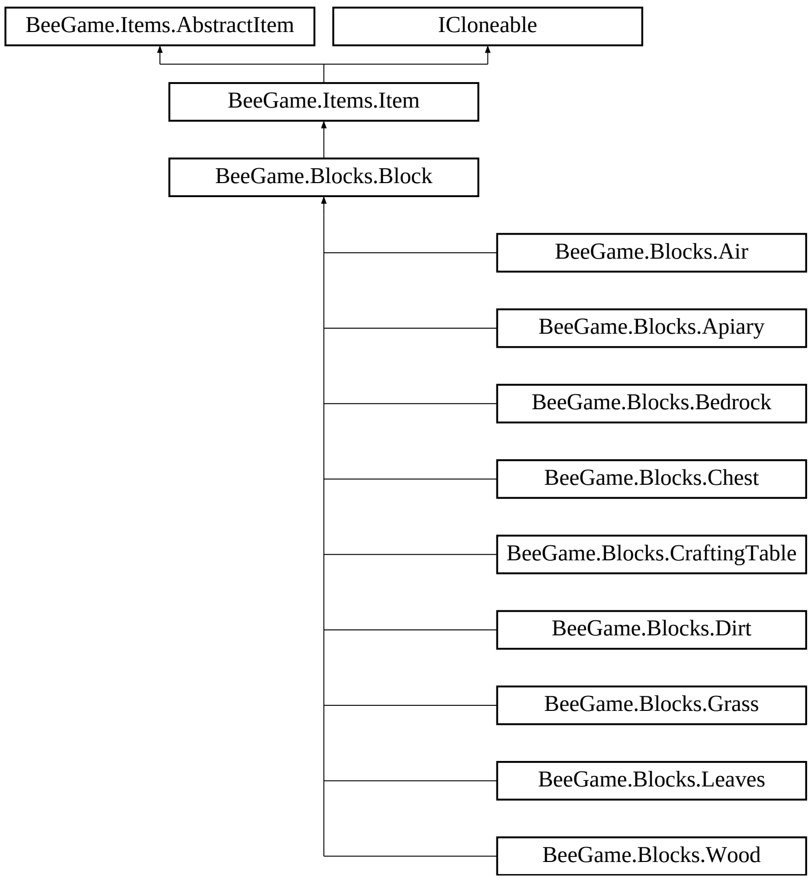
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/Bee↵ Dictionarys.cs

6.10 BeeGame.Blocks.Block Class Reference

Base class for blocks

Inheritance diagram for BeeGame.Blocks.Block:



Public Member Functions

- Block ()

  Constructor sets the Item.placeable to true
- Block (string name)

  Sets placeabel to true and sets name of the block/item
- override Sprite GetItemSprite ()

  Returns the sprite for the item
- virtual void BreakBlock (THVector3 pos)

  Spawns an item with the same texture as the broken block
- virtual void UpdateBlock (int x, int y, int z, Chunk chunk)

  Should this Block be updated when the mesh is made
- virtual bool InteractWithBlock (BeeGame.Inventory.Inventory inv)

  Can this block be interacted with?
- virtual MeshData BlockData (Chunk chunk, int x, int y, int z, MeshData meshData, bool addToRender↩ Mesh=true)

  The data that this block adds to the mesh
- virtual bool IsSolid (Direction direction)

  What Directions is this Block solid in

- override int GetHashCode ()

    Hascode for the Block
- override string ToString ()

    Returns the Block name and Id formatted nicely

Public Attributes

- bool breakable = true

    Can this Block be broken
- bool changed = true

    Has this block been placed by the player
- override bool placeable => true

    Sets so that blocks can be placed

Static Public Attributes

- static new int ID = 1

Additional Inherited Members

6.10.1 Detailed Description

Base class for blocks

Definition at line 14 of file Block.cs.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 Block() [1/2]

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

Constructor sets the Item.placeable to true

Definition at line 36 of file Block.cs.

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

6.10.2.2 Block() [2/2]

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

Sets placeabel to true and sets name of the block/item

Parameters

name	Name of the block/item

Definition at line 45 of file Block.cs.

```
                                   : base(name)
        {
        }
```

6.10.3 Member Function Documentation

6.10.3.1 BlockData()

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

The data that this block adds to the mesh

Parameters

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

Returns

Given *meshData* with this blocks data added to it

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

Reimplemented in BeeGame.Blocks.CraftingTable, BeeGame.Blocks.Chest, BeeGame.Blocks.Apiary, and BeeGame.Blocks.Air.

Definition at line 102 of file Block.cs.

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

            //* Adds the Bottom face of the block
            if (!chunk.GetBlock(x, y - 1, z, false).IsSolid(Direction.UP))
            {
                meshData = FaceDataDown(x, y, z, meshData, addToRenderMesh);
            }

            //* Adds the North face of the block
            if (!chunk.GetBlock(x, y, z + 1, false).IsSolid(Direction.SOUTH))
            {
                meshData = FaceDataNorth(x, y, z, meshData, addToRenderMesh);
            }

            //* Adds the South face of the block
            if (!chunk.GetBlock(x, y, z - 1, false).IsSolid(Direction.NORTH))
            {
                meshData = FaceDataSouth(x, y, z, meshData, addToRenderMesh);
            }

            //* Adds the East face of the block
            if (!chunk.GetBlock(x + 1, y, z, false).IsSolid(Direction.WEST))
            {
                meshData = FaceDataEast(x, y, z, meshData, addToRenderMesh);
            }

            //* Adds the West face of the block
            if (!chunk.GetBlock(x - 1, y, z, false).IsSolid(Direction.EAST))
            {
                meshData = FaceDataWest(x, y, z, meshData, addToRenderMesh);
            }

            return meshData;
        }
```

6.10.3.2 BreakBlock()

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

Spawns an item with the same texture as the broken block

Parameters

pos	position to spawn the Item

Reimplemented in BeeGame.Blocks.CraftingTable, BeeGame.Blocks.Chest, BeeGame.Blocks.Apiary, Bee↩ Game.Blocks.Bedrock, and BeeGame.Blocks.Air.

Definition at line 62 of file Block.cs.

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

6.10.3.3 GetHashCode()

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

Hascode for the Block

Returns

1

Implements BeeGame.Items.AbstractItem.

Reimplemented in BeeGame.Blocks.CraftingTable, BeeGame.Blocks.Chest, BeeGame.Blocks.Grass, BeeGame.↩
Blocks.Dirt, BeeGame.Blocks.Leaves, and BeeGame.Blocks.Wood.

Definition at line 159 of file Block.cs.

```
        {
            return ID;
        }
```

6.10.3.4 GetItemSprite()

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

Returns the sprite for the item

Returns

Sprite for this item

Reimplemented from BeeGame.Items.Item.

Reimplemented in BeeGame.Blocks.CraftingTable, BeeGame.Blocks.Grass, BeeGame.Blocks.Dirt, BeeGame.↩
Blocks.Wood, and BeeGame.Blocks.Leaves.

Definition at line 51 of file Block.cs.

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

6.10.3.5 InteractWithBlock()

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

Can this block be interacted with?

Returns

False by default

Definition at line 81 of file Block.cs.

```
        {
            return false;
        }
```

6.10.3.6 IsSolid()

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

What Directions is this Block solid in

Parameters

direction	Direction to check

Returns

Default returns true for all sides

Reimplemented in BeeGame.Blocks.Air, and BeeGame.Blocks.Leaves.

Definition at line 148 of file Block.cs.

```
        {
            return true;
        }
```

6.10.3.7 ToString()

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

Returns the Block name and Id formatted nicely

Returns

Definition at line 168 of file Block.cs.

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

6.10.3.8 UpdateBlock()

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

Should this Block be updated when the mesh is made

Parameters

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

Reimplemented in BeeGame.Blocks.Grass.

Definition at line 75 of file Block.cs.

```
00075 { }
```

6.10.4 Member Data Documentation

6.10.4.1 breakable

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

Can this Block be broken

Definition at line 21 of file Block.cs.

6.10.4.2 changed

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

Has this block been placed by the player

Definition at line 25 of file Block.cs.

6.10.4.3 ID

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

Definition at line 17 of file Block.cs.

6.10.4.4 placeable

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

Sets so that blocks can be placed

Definition at line 29 of file Block.cs.

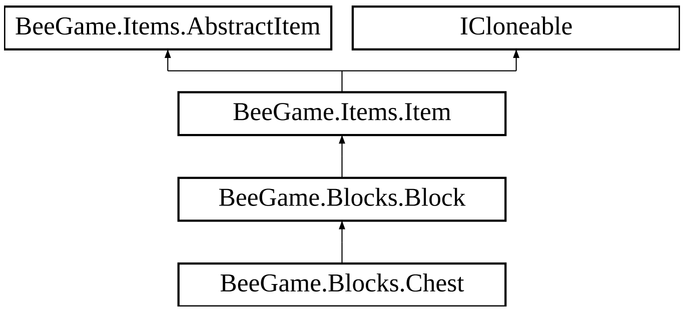
The documentation for this class was generated from the following file:

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

6.11 BeeGame.Blocks.Chest Class Reference

Chest Block

Inheritance diagram for BeeGame.Blocks.Chest:



Public Member Functions

- Chest ()

    Makes a new chest from a parmaterless constructor
- override GameObject GetGameObject ()

    Gets the gme object for this chest
- override Tile TexturePosition (Direction direction)

    Returns the texture for the chest Block
- override MeshData BlockData (Chunk chunk, int x, int y, int z, MeshData meshData, bool addToRender↩ Mesh=true)

    The data that this block adds to the mesh
- override void BreakBlock (THVector3 pos)

    Breaks the block
- override bool InteractWithBlock (BeeGame.Inventory.Inventory inv)

    Opens the ChestInventory when clicked on
- override int GetHashCode ()

    Gets the ID of the Block
- override string ToString ()

    Returns the Block name and Id formatted nicely

Static Public Attributes

- static new int ID => 8

Private Attributes

- GameObject myGameobject

    Chest model for when it is placed

Additional Inherited Members

6.11.1 Detailed Description

Chest Block

Definition at line 16 of file Chest.cs.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 Chest()

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

Makes a new chest from a parmaterless constructor

Definition at line 32 of file Chest.cs.

```
                        : base("Chest")
          {
              usesGameObject = true;
          }
```

6.11.3 Member Function Documentation

6.11.3.1 BlockData()

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

The data that this block adds to the mesh

Parameters

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

Returns

Given *meshData* with this blocks data added to it

Only adds to the colision mesh as the model is handlled by the unity prefab system

Reimplemented from BeeGame.Blocks.Block.

Definition at line 74 of file Chest.cs.

```
        {
            if (myGameobject == null)
            {
                myGameobject = UnityEngine.Object.Instantiate(
    PrefabDictionary.GetPrefab("Chest"), new THVector3(x, y, z) + chunk.
    chunkWorldPos, Quaternion.identity, chunk.transform);
                myGameobject.GetComponent<ChestInventory>().inventoryPosition =
    new THVector3(x, y, z) + chunk.chunkWorldPos;
                myGameobject.GetComponent<ChestInventory>().SetChestInventory();
            }
            return base.BlockData(chunk, x, y, z, meshData, true);
        }
```

6.11.3.2 BreakBlock()

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

Breaks the block

Parameters

pos	Position of the block

Reimplemented from BeeGame.Blocks.Block.

Definition at line 89 of file Chest.cs.

```
        {
            //* removes the blocks blocks inventory save file and destroys the game object
            Serialization.Serialization.DeleteFile(myGameobject.GetComponent<
    ChestInventory>().inventoryName);
            UnityEngine.Object.Destroy(myGameobject);
            //* removes the collision mesh from the chunk
            base.BreakBlock(pos);
        }
```

6.11.3.3 GetGameObject()

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

Gets the gme object for this chest

Returns

THe chest game object

Reimplemented from BeeGame.Items.Item.

Definition at line 43 of file Chest.cs.

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

6.11.3.4 GetHashCode()

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

Gets the ID of the Block

Returns

8

Reimplemented from BeeGame.Blocks.Block.

Definition at line 117 of file Chest.cs.

```
        {
            return ID;
        }
```

6.11.3.5 InteractWithBlock()

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

Opens the ChestInventory when clicked on

Parameters

inv	Inventory that the chest is interactiong with

Returns

true

Definition at line 105 of file Chest.cs.

```
        {
            myGameobject.GetComponent<ChestInventory>().ToggleInventory(inv);
            return true;
        }
```

6.11.3.6 TexturePosition()

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

Returns the texture for the chest Block

Parameters

direction	Direction of thhe desired face

Returns

Tile with the textture coordinates of the Block texture

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

Reimplemented from BeeGame.Items.Item.

Definition at line 56 of file Chest.cs.

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

6.11.3.7 ToString()

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

Returns the Block name and Id formatted nicely

Returns

Definition at line 126 of file Chest.cs.

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

6.11.4 Member Data Documentation

6.11.4.1 ID

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

Definition at line 25 of file Chest.cs.

6.11.4.2 myGameobject

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

Chest model for when it is placed

Definition at line 23 of file Chest.cs.

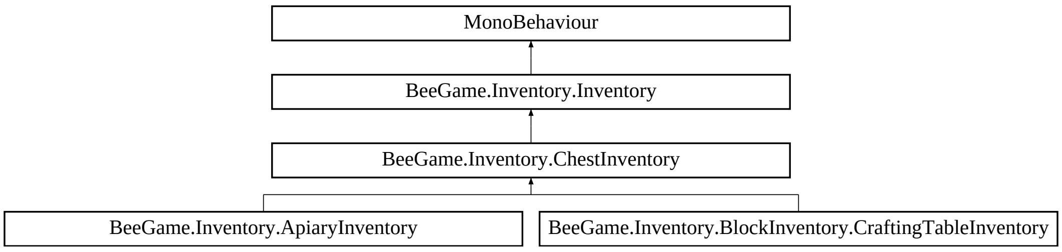
The documentation for this class was generated from the following file:

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

6.12 BeeGame.Inventory.ChestInventory Class Reference

Incentory for the chests

Inheritance diagram for BeeGame.Inventory.ChestInventory:


MonoBehaviour
BeeGame.Inventory.Inventory
BeeGame.Inventory.ChestInventory
BeeGame.Inventory.ApiaryInventory
BeeGame.Inventory.BlockInventory.CraftingTableInventory

Public Member Functions

- void UpdateChestInventory ()

  The unity Update method is not called if the class is is child...annoyingly
- virtual void SetChestInventory (string invName="Chest")

  Sets the Size and name of this Inventory
- override void ToggleInventory (Inventory inv)

  Opens and closes the inventory

Public Attributes

- THVector3 inventoryPosition

  Position in worldspace of the chest
- Inventory playerinventory

  Refernce to the players Inventory so that it can be updated when chest is closed
- GameObject inventory

  The inventory gameobject that will be displayed
- int inventorySize

  How many slots are in this Inventory

Private Member Functions

- void Update ()

  Updates the slots and checks if the inventory should be closed
- void SetPlayerItems ()

  Puts the player items into the chest
- void ApplyPlayerItems ()

  Applies the changes made to the playerinventory in this

Additional Inherited Members

6.12.1 Detailed Description

Incentory for the chests

Definition at line 11 of file ChestInventory.cs.

6.12.2 Member Function Documentation

6.12.2.1 ApplyPlayerItems()

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

Applies the changes made to the playerinventory in this

Definition at line 88 of file ChestInventory.cs.

```
        {
            for (int i = 0; i < playerinventory.items.
    itemsInInventory.Length; i++)
            {
                playerinventory.items.itemsInInventory[i] =
    items.itemsInInventory[i + (inventorySize - 36)];
            }

            playerinventory.SaveInv();
        }
```

6.12.2.2 SetChestInventory()

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

Sets the Size and name of this Inventory

Reimplemented in BeeGame.Inventory.BlockInventory.CraftingTableInventory, and BeeGame.Inventory.Apiary↩
Inventory.

Definition at line 60 of file ChestInventory.cs.

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

            //* sets the name and postion if this inventory used during serialization and deserialization
            inventoryName = $"{invName} @ {(ChunkWorldPos)inventoryPosition}";

            //* loads the inventory if it had had items put in it last time it existed
            Serialization.Serialization.DeSerializeInventory(this, inventoryName);
        }
```

6.12.2.3 SetPlayerItems()

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

Puts the player items into the chest

Definition at line 77 of file ChestInventory.cs.

```
        {
            for (int i = 0; i < playerinventory.items.itemsInInventory.Length; i++)
            {
                items.itemsInInventory[i + (inventorySize - 36)] = playerinventory.items.itemsInInventory[i];
            }
        }
```

6.12.2.4 ToggleInventory()

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

Opens and closes the inventory

Parameters

inv	

Reimplemented from BeeGame.Inventory.Inventory.

Reimplemented in BeeGame.Inventory.BlockInventory.CraftingTableInventory.

Definition at line 103 of file ChestInventory.cs.

```
        {
            //* sets the player inventory
            playerinventory = inv;

            thisInventoryOpen = !thisInventoryOpen;

            isAnotherInventoryOpen = thisInventoryOpen;

            inventory.SetActive(!inventory.activeInHierarchy);

            if (inventory.activeInHierarchy)
            {
                chestOpen = true;

                //* stops the player invnetory from being opened immidiatly after this is closed
                blockInventoryJustClosed = true;
                SetPlayerItems();
                //* hides and locks the cursor
                Cursor.lockState = CursorLockMode.None;
                Cursor.visible = true;
            }
            else
            {
                chestOpen = false;

                //* puts the items into the chest
                //* shows and unlocks the cursor
                ApplyPlayerItems();
                Cursor.lockState = CursorLockMode.Locked;
                Cursor.visible = false;
            }
        }
```

6.12.2.5 Update()

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

Updates the slots and checks if the inventory should be closed

Definition at line 37 of file ChestInventory.cs.

```
        {
            UpdateChestInventory();
        }
```

6.12.2.6 UpdateChestInventory()

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

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

Definition at line 45 of file ChestInventory.cs.

```
        {
            //* the chest should always have a player inventory when it does this but checks just in case
            if (playerinventory != null)
                UpdateBase();

            //* checks if the inventory should be closed
            if (GetButtonDown("Player Inventory") && thisInventoryOpen &&
     floatingItem == null)
                ToggleInventory(playerinventory);
        }
```

6.12.3 Member Data Documentation

6.12.3.1 inventory

`GameObject BeeGame.Inventory.ChestInventory.inventory`

The inventory gameobject that will be displayed

Definition at line 25 of file ChestInventory.cs.

6.12.3.2 inventoryPosition

`THVector3 BeeGame.Inventory.ChestInventory.inventoryPosition`

Position in worldspace of the chest

Definition at line 17 of file ChestInventory.cs.

6.12.3.3 inventorySize

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

How many slots are in this Inventory

Definition at line 30 of file ChestInventory.cs.

6.12.3.4 playerinventory

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

Refernce to the players Inventory so that it can be updated when chest is closed

Definition at line 21 of file ChestInventory.cs.

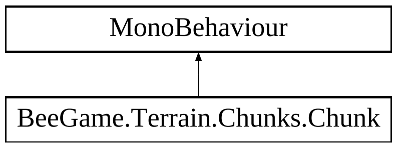
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/BlockInventory/Chest↩
  Inventory.cs

6.13 BeeGame.Terrain.Chunks.Chunk Class Reference

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

Inheritance diagram for BeeGame.Terrain.Chunks.Chunk:



Public Member Functions

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

Static Public Member Functions

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

Public Attributes

- Block [,,] blocks = new Block[chunkSize, chunkSize, chunkSize]

    All of the Blocks in the Chunk
- bool update = true

    Should the Chunk be updated?
- bool rendered

    Is the Chunk rendered?
- bool updateCollsionMesh = false

    Should the chunks collision mesh be updated?
- bool applyCollisionMesh = false

    Should the collision mesh be applied
- World world

    World that this chunk is in as MonoBehaviours cannot be static this is for convenicence
- ChunkWorldPos chunkWorldPos

    Chunks position in the world as a ChunkWorldPos (int verson of Core.THVector3)

Static Public Attributes

- static int chunkSize = 16

    Size of the Chunk

Private Member Functions

- void Start ()

    Sets the meshCollider and filter variables
- void Update ()

    Checks if the Chunk should be updated
- void UpdateChunk ()

    Updates the mesh for the Chunk
- void RenderMesh (MeshData meshData)

    Renders the given MeshData into a unity Mesh
- void ColliderMesh ()

    Makes a collision mesh from the mesh

Private Attributes

- MeshData mesh = new MeshData()

    MeshData of this chunk
- MeshFilter filter

    This Chunks mesh filter
- MeshCollider meshCollider

    This Chunks mesh colldier

6.13.1 Detailed Description

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

Definition at line 14 of file Chunk.cs.

6.13.2 Member Function Documentation

6.13.2.1 ColliderMesh()

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

Makes a collision mesh from the mesh

Definition at line 237 of file Chunk.cs.

```
        {
            //* if the chunk has been told to update the collsions but the chunk has ne verts dont do it as
    their is no point
            if (this.mesh.verts.Count == 0)
                return;

            //* if the render and collision meshes should be shared set the render mesh to the collision
    mesh otherwise make a collision mesh
            if (this.mesh.shareMeshes)
            {
                world.chunkHasMadeCollisionMesh = true;
                applyCollisionMesh = false;
                meshCollider.sharedMesh = filter.mesh;
                return;
            }

            world.chunkHasMadeCollisionMesh = true;
            //* Applying the mesh takes the longest but nothing can be done with the mesh class in a
    secondary thread...thanks Unity

            //* makes a new mesh setting the name for convenience
            Mesh mesh = new Mesh()
            {
                name = "Collider Mesh",
                vertices = this.mesh.colVerts.ToArray(),
                triangles = this.mesh.colTris.ToArray()
            };

            //* recalcs the normals and applies the mesh
            mesh.RecalculateNormals();

            meshCollider.sharedMesh = mesh;

            applyCollisionMesh = false;
        }
```

6.13.2.2 GetBlock()

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

Returns the Block in the given x, y, z

Parameters

x	X pos if the Block
y	Z pos if the Block
z	Y pos if the Block
checkNebouringChunks	Shoud this check nebouring 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.

```
        {
            //* checks that block is in the chunk
            if (InRange(x) && InRange(y) && InRange(z))
                return blocks[x, y, z];

            //* if the block is not in the chunk and we should check other chunks do that, otherwise return
 an air block (empty block)
            //if(checkNebouringChunks)
                return world.GetBlock(chunkWorldPos.x + x,
chunkWorldPos.y + y, chunkWorldPos.z + z);

            //return new Air();
        }
```

6.13.2.3 InRange()

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

Checks that a given value is within the Chunk

Parameters

i	Value to check

Returns

true if the value is in the Chunk

Definition at line 162 of file Chunk.cs.

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

6.13.2.4 RenderMesh()

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

Renders the given MeshData into a unity Mesh

Parameters

meshData	Mesh data to render

Definition at line 213 of file Chunk.cs.

```
        {
            //* Applying the mesh takes the longest but nothing can be dont with the mesh class in a
    secondary thread...thanks unity

            mesh.done = false;
            //* clears the current chunk mesh
            filter.mesh.Clear();
            //* name for convenience
            filter.mesh.name = "Render Mesh";
            //* puts the tris and verts from the meshdata into the chunk mesh
            filter.mesh.vertices = meshData.verts.ToArray();
            filter.mesh.triangles = meshData.tris.ToArray();

            //* sets the uvs
            filter.mesh.uv = meshData.uv.ToArray();

            //* redoes the normals incase they got messed up
            filter.mesh.RecalculateNormals();
            //* is this necissary as it causes alsot of lag?
        }
```

6.13.2.5 SetBlock()

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

Sets a Block in the given position

Parameters

x	X pos of the Block
y	Y pos of the Block
z	Z pos of the Block
block	Block to set

Definition at line 143 of file Chunk.cs.

```
        {
            //* sets the block in the position if it is in the chunk, then return early
            if (InRange(x) && InRange(y) && InRange(z))
            {
                blocks[x, y, z] = block;
                return;
            }

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

6.13.2.6 SetBlocksUnmodified()

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

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

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

Definition at line 178 of file Chunk.cs.

```
        {
            foreach (var block in blocks)
            {
                block.changed = false;
            }
        }
```

6.13.2.7 Start()

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

Sets the meshCollider and filter variables

Definition at line 77 of file Chunk.cs.

```
        {
            filter = GetComponent<MeshFilter>();
            meshCollider = GetComponent<MeshCollider>();
        }
```

6.13.2.8 Update()

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

Checks if the Chunk should be updated

Definition at line 86 of file Chunk.cs.

```
        {
            lock(mesh)
            {
                if (update)
                {
                    update = false;
                    updateCollsionMesh = true;
                    mesh = new MeshData();
                    //* Enabling threading here works in editor but not in build?
                    //* ok whatever...
                    //* Thread thread = new Thread(UpdateChunk);

                    //* thread.Start();
                    UpdateChunk();
                }

                if (mesh.done && mesh != new MeshData())
                {
                    RenderMesh(mesh);
                }

                if (applyCollisionMesh)
                    ColliderMesh();
            }
        }
```

6.13.2.9 UpdateChunk()

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

Updates the mesh for the Chunk

Definition at line 189 of file Chunk.cs.

```
        {
            //* says that this chunk is rendered and initialtes the mesh
            rendered = true;

            //* goes through every block in the blocks array getting their mesh data
            for (int x = 0; x < chunkSize; x ++)
            {
                for (int z = 0; z < chunkSize; z ++)
                {
                    for (int y = 0; y < chunkSize; y ++)
                    {
                        blocks[x, y, z]?.UpdateBlock(x, y, z, this);
                        mesh = blocks[x, y, z]?.BlockData(this, x, y, z,
      mesh) ?? mesh;
                    }
                }
            }
            mesh.done = true;
        }
```

6.13.3 Member Data Documentation

6.13.3.1 applyCollisionMesh

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

Should the collision mesh be applied

Definition at line 47 of file Chunk.cs.

6.13.3.2 blocks

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

All of the Blocks in the Chunk

Definition at line 29 of file Chunk.cs.

6.13.3.3 chunkSize

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

Size of the Chunk

Same size for x, y, z
Posibly some place has 16 hard coded as reduceing the number breaks things TODO: find

Definition at line 24 of file Chunk.cs.

6.13.3.4 chunkWorldPos

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

Chunks position in the world as a ChunkWorldPos (int verson of Core.THVector3)

Definition at line 56 of file Chunk.cs.

6.13.3.5 filter

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

This Chunks mesh filter

Definition at line 66 of file Chunk.cs.

6.13.3.6 mesh

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

MeshData of this chunk

Definition at line 61 of file Chunk.cs.

6.13.3.7 meshCollider

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

This Chunks mesh colldier

Definition at line 70 of file Chunk.cs.

6.13.3.8 rendered

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

Is the Chunk rendered?

Definition at line 38 of file Chunk.cs.

6.13.3.9 update

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

Should the Chunk be updated?

Definition at line 34 of file Chunk.cs.

6.13.3.10 updateCollsionMesh

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

Should the chunks collision mesh be updated?

Definition at line 43 of file Chunk.cs.

6.13.3.11 world

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

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

Definition at line 52 of file Chunk.cs.

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

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/Chunk.cs

6.14 BeeGame.Terrain.ChunkWorldPos Struct Reference

Serializable int version of THVector3

Public Member Functions

- ChunkWorldPos (int x, int y, int z)

  Constructor so that values can be input on creation of the vector
- override string ToString ()

  Formats the values nicely incase it is needed
- override bool Equals (object obj)
- override int GetHashCode ()

  Makes a unique hascode for the vector

Static Public Member Functions

- static implicit operator THVector3 (ChunkWorldPos pos)

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

  Converts a ChunkWorldPos to a THVector3

Public Attributes

- int x

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

6.14.1 Detailed Description

Serializable int version of THVector3

Definition at line 10 of file ChunkWorldPos.cs.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 ChunkWorldPos()

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

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

Parameters

x	X Value
y	Y Value
z	Z Value

Definition at line 23 of file ChunkWorldPos.cs.

```
        {
            this.x = x;
            this.y = y;
            this.z = z;
        }
```

6.14.3 Member Function Documentation

6.14.3.1 Equals()

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

Definition at line 41 of file ChunkWorldPos.cs.

```
        {
            //* possibly remove and just check if obj is null
            if (!(obj is ChunkWorldPos))
                return false;

            ChunkWorldPos temp = (ChunkWorldPos)obj;

            //* possibly change to hashcode checking
            if (temp.x == x && temp.y == y && temp.z == z)
                return true;

            return false;
        }
```

6.14.3.2 GetHashCode()

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

Makes a unique hascode for the vector

Returns

unique int value for the vector

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

Definition at line 63 of file ChunkWorldPos.cs.

```
        {
            unchecked
            {
                int hashcode = 47;

                hashcode *= 227 + x.GetHashCode();
                hashcode *= 227 + y.GetHashCode();
                hashcode *= 227 + z.GetHashCode();

                return hashcode;
            }
        }
```

6.14.3.3 operator ChunkWorldPos()

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

Converts a ChunkWorldPos to a THVector3

Parameters

pos	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.

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

6.14.3.4 operator THVector3()

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

Converts a ChunkWorldPos to a THVector3 without the need for an explicit cast as no data will be lost

Parameters

pos	this ChunkWorldPos

Definition at line 81 of file ChunkWorldPos.cs.

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

6.14.3.5 ToString()

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

Formats the values nicely incase it is needed

Returns

Definition at line 34 of file ChunkWorldPos.cs.

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

6.14.4 Member Data Documentation

6.14.4.1 x

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

x, y, z values for the vector

Definition at line 15 of file ChunkWorldPos.cs.

6.14.4.2 y

`int BeeGame.Terrain.ChunkWorldPos.y`

Definition at line 15 of file ChunkWorldPos.cs.

6.14.4.3 z

`int BeeGame.Terrain.ChunkWorldPos.z`

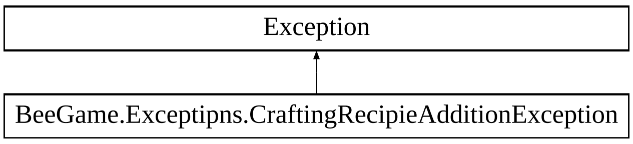
Definition at line 15 of file ChunkWorldPos.cs.

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

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/ChunkWorldPos.cs

6.15 BeeGame.Exceptipns.CraftingRecipieAdditionException Class Reference

Inheritance diagram for BeeGame.Exceptipns.CraftingRecipieAdditionException:



Public Member Functions

- CraftingRecipieAdditionException ()
- CraftingRecipieAdditionException (string message)
- CraftingRecipieAdditionException (string message, Exception innerException)

6.15.1 Detailed Description

Definition at line 8 of file CraftingRecipieAdditionException.cs.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 CraftingRecipieAdditionException() [1/3]

`BeeGame.Exceptipns.CraftingRecipieAdditionException.CraftingRecipieAdditionException ( )`

Definition at line 10 of file CraftingRecipieAdditionException.cs.

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

6.15.2.2 CraftingRecipieAdditionException() [2/3]

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

Definition at line 15 of file CraftingRecipieAdditionException.cs.

```
                                                                       : base(message)
        {

        }
```

6.15.2.3 CraftingRecipieAdditionException() [3/3]

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

Definition at line 20 of file CraftingRecipieAdditionException.cs.

```
                                                                                            : base(message,
     innerException)
        {

        }
```

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

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Exceptipns/CraftingRecipie←
  AdditionException.cs

6.16 BeeGame.Core.Dictionarys.CraftingRecipies Class Reference

Static Public Member Functions

- static void AddShapedRecipie (object[ ] recipie, Item result)

  Will add a shaped crafting recipie to the game
- static Item GetShapedRecipeItem (string recipie)

  Returns an Item from the shapedCraftingRecipies dictionary
- static void AddShaplessRecipie (object[ ] recipie, Item result)

  Adds a Shapless recipie to the dictionary
- static string GetShaplessRecipieString (Item[ ] recipie)

  Gets a shapless recipie string from a given recipie
- static Item GetShaplessRecipieResult (int[ ] recipie)

  Trys to get a shapless recipe
- static Item GetShaplessRecipieResult (string recipie)

  Trys to get a shapless recipie
- static Item GetShaplessRecipieResult (Item[ ] recipie)

  Trys to get a shapless recipie

Static Private Attributes

- static Dictionary< string, Item > shapedCraftingRecipies = new Dictionary<string, Item>()

  Contains all crafting recipies that require a certian layout in the crafting grid (Blocks.CraftingTable
- static Dictionary< string, Item > shaplessRecipies

  All shapless recipies

6.16.1 Detailed Description

Definition at line 10 of file CraftingRecipies.cs.

6.16.2 Member Function Documentation

6.16.2.1 AddShapedRecipie()

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

Will add a shaped crafting recipie to the game

Parameters

recipie	The desired recipie. Layout is {"XXX", "XXX", "XXX", "X", ItemID} where each X is a slot in the crafting grid, Each group of 3 is a row, and a "X", ItemID is the Item ID X represents (for each new item a new symbol is required), a Sapce is no item required in that slot
result	The Item that the recipie will produce

This example shows how to call AddShapedRecipie(object[ ], Item)

```
void Main()
{
    CraftingRecipies.AddShapedRecipie(new object[] { " X ", "X@X", " X ", "X", Wood.GetItemID(), "@", Stone
      .GetItemID() }, new Chest());
}
```

Definition at line 32 of file CraftingRecipies.cs.

```
        {
            //* converts the given blocks of 3 haracters to a 9 character string
            var stringRecipie = "";

            for (int i = 0; i < 3; i++)
            {
                stringRecipie += recipie[i] as string;
            }

            //* gets what character represents which item
            for (int i = 3; i < recipie.Length; i += 2)
            {
                var character = (string)recipie[i];
                var itemID = (int)recipie[i + 1];

                //* replaces the character with the items id
                stringRecipie = stringRecipie.Replace(character, $"{itemID.ToString()}:");
            }

            //* converts empty sots " " into "0:"
            stringRecipie = stringRecipie.Replace(" ", "0:");

            //* if the recipe exists an exception is thrown as two recipies cannot be the same
            if (shapedCraftingRecipies.ContainsKey(stringRecipie))
                throw new CraftingRecipieAdditionException($"Shaped Recipie already exists: {stringRecipie}");

            //* adds the recipie to the dictionary
            shapedCraftingRecipies.Add(stringRecipie, result);
        }
```

6.16.2.2 AddShaplessRecipie()

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

Adds a Shapless recipie to the dictionary

Parameters

recipie	Recipie to add. Format as { Item, Number of items }
result	Result of the crafting recipie

2 Examples of adding a shapless recipie

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

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

Definition at line 106 of file CraftingRecipies.cs.

```
        {
            var itemList = new List<int>();
            var stringRecpie = "";

            for (int i = 0; i < recipie.Length; i+=2)
            {
                for (int j = 0; j < (int)recipie[i+1]; j++)
                {
                    itemList.Add(int.Parse(((Item)recipie[i]).GetItemID()));
                }
            }

            itemList.Sort();

            for (int i = 0; i < itemList.Count; i++)
            {
                stringRecpie += $"{itemList[i]}:";
            }

            if (shaplessRecipies.ContainsKey(stringRecpie))
                throw new CraftingRecipieAdditionException($"Shaped Recipie already exists: {stringRecpie}");

            shaplessRecipies.Add(stringRecpie, result);
        }
```

6.16.2.3 GetShapedRecipeItem()

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

Returns an Item from the shapedCraftingRecipies dictionary

Parameters

recipie	Recipie for Item

Returns

An Item or null is recipie was not found

Definition at line 68 of file CraftingRecipies.cs.

```
        {
            shapedCraftingRecipies.TryGetValue(recipie, out var item);

            return item;
        }
```

6.16.2.4 GetShaplessRecipieResult() [1/3]

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

Trys to get a shapless recipe

Parameters

recipie	Recipie to get

Returns

Item for the recipie, null if recipie does not exist

Definition at line 166 of file CraftingRecipies.cs.

```
        {
            var list = recipie.ToList();
            list.Sort();

            var stringRecipe = "";

            for (int i = 0; i < list.Count; i++)
            {
                stringRecipe += $"{list[i]}:";
            }

            return GetShaplessRecipieResult(stringRecipe);
        }
```

6.16.2.5 GetShaplessRecipieResult() [2/3]

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

Trys to get a shapless recipie

Parameters

recipie	Recipie to get

Returns

Item for the recipie, null if recipie does not exist

Definition at line 186 of file CraftingRecipies.cs.

```
        {
            shaplessRecipies.TryGetValue(recipie, out var item);

            return item;
        }
```

6.16.2.6 GetShaplessRecipieResult() [3/3]

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

Trys to get a shapless recipie

Parameters

recipie	Recipie to get

Returns

Item for the recipie, null if recipie does not exist

Definition at line 198 of file CraftingRecipies.cs.

```
        {
            shaplessRecipies.TryGetValue(
     GetShaplessRecipieString(recipie), out var item);

            return item;
        }
```

6.16.2.7 GetShaplessRecipieString()

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

Gets a shapless recipie string from a given recipie

Parameters

recipie	Recipie for string

Returns

A string of the given shapless recipie

Definition at line 137 of file CraftingRecipies.cs.

```
        {
            var IDList = new List<int>();
            var stringRecipe = "";

            //* converts tthe given item list to an ID list so it can be sorted
            for (int i = 0; i < recipie.Length; i++)
            {
                if(recipie[i] != null)
                    IDList.Add(recipie[i].GetHashCode());
            }

            IDList.Sort();

            //* converts the sorted ID list to a string so can be used as a dictionary key
            for (int i = 0; i < IDList.Count; i++)
            {
                //* : after each ID as it is possible for ID clashes without eg ID: 11 can be seen as 2 *
    ID: 1
                stringRecipe += $"{IDList[i]}:";
            }

            return stringRecipe;
        }
```

6.16.3 Member Data Documentation

6.16.3.1 shapedCraftingRecipies

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

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

Definition at line 16 of file CraftingRecipies.cs.

6.16.3.2 shaplessRecipies

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

Initial value:

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

        }
```

All shapless recipies

Definition at line 80 of file CraftingRecipies.cs.

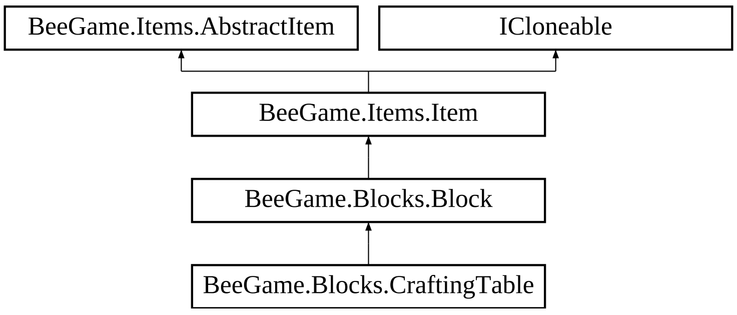
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/Crafting↩
Recipies.cs

6.17 BeeGame.Blocks.CraftingTable Class Reference

The Workbanch Block class

Inheritance diagram for BeeGame.Blocks.CraftingTable:



Public Member Functions

- CraftingTable ()

  Constructor
- Item ReturnShapedRecipieItem (Item[ ] items)

  Makes a shaped crafting recipie from the given items and return if it is a recipie
- virtual Item ReturnShapelessRecipieItem (Item[ ] items)
- virtual Item ReturnShapedRecipieItem (string recipie)

  Returns a crafting recipie from a given recipie
- override bool InteractWithBlock (Inventory.Inventory inv)

  Toggles the CraftingTableInventory for the block
- override GameObject GetGameObject ()

  Returns this Blocks game object
- override MeshData BlockData (Chunk chunk, int x, int y, int z, MeshData meshData, bool addToRender←Mesh=true)

  The data that this block adds to the mesh
- override void BreakBlock (THVector3 pos)

  Breaks the Block
- override Sprite GetItemSprite ()

  Returns the sprite for the Item
- override Tile TexturePosition (Direction direction)

  Returns the texture for the apiary Block
- override int GetHashCode ()

  Returns the ID of the Item

Static Public Attributes

- static new int ID => 9

  This blocks ID

Private Attributes

- GameObject myGameobject

  The GameObject for this block

Additional Inherited Members

6.17.1 Detailed Description

The Workbanch Block class

Definition at line 15 of file CraftingTable.cs.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 CraftingTable()

```
BeeGame.Blocks.CraftingTable.CraftingTable ( )
```

Constructor

Definition at line 34 of file CraftingTable.cs.

```
                                : base("Workbench")
        {
            usesGameObject = true;
        }
```

6.17.3 Member Function Documentation

6.17.3.1 BlockData()

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

The data that this block adds to the mesh

Parameters

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

Returns

Given *meshData* with this blocks data added to it

Only adds to the colision mesh as the model is handlled by the unity prefab system

Reimplemented from BeeGame.Blocks.Block.

Definition at line 118 of file CraftingTable.cs.

```
        {
            if (myGameobject == null)
            {
                myGameobject = UnityEngine.Object.Instantiate(
    PrefabDictionary.GetPrefab("CraftingTable"), new
    THVector3(x, y, z) + chunk.chunkWorldPos, Quaternion.identity, chunk.transform);
            }
            return base.BlockData(chunk, x, y, z, meshData, true);
        }
```

6.17.3.2 BreakBlock()

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

Breaks the Block

Parameters

pos	Positon of the Block

Reimplemented from BeeGame.Blocks.Block.

Definition at line 131 of file CraftingTable.cs.

```
        {
            //* removes the game object
            UnityEngine.Object.Destroy(myGameobject);
            //* removes the collision mesh from the chunk
            base.BreakBlock(pos);
        }
```

6.17.3.3 GetGameObject()

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

Returns this Blocks game object

Returns

Reimplemented from BeeGame.Items.Item.

Definition at line 100 of file CraftingTable.cs.

```
        {
            return PrefabDictionary.GetPrefab("CraftingTable");
        }
```

6.17.3.4 GetHashCode()

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

Returns the ID of the Item

Returns

ID

Reimplemented from BeeGame.Blocks.Block.

Definition at line 167 of file CraftingTable.cs.

```
        {
            return ID;
        }
```

6.17.3.5 GetItemSprite()

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

Returns the sprite for the Item

Returns

Sprite for this Item

Reimplemented from BeeGame.Blocks.Block.

Definition at line 143 of file CraftingTable.cs.

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

6.17.3.6 InteractWithBlock()

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

Toggles the CraftingTableInventory for the block

Parameters

inv	

Returns

Definition at line 89 of file CraftingTable.cs.

```
        {
            myGameobject.GetComponent<Inventory.BlockInventory.CraftingTableInventory>().
     myblock = this;
            myGameobject.GetComponent<Inventory.BlockInventory.CraftingTableInventory>().
     ToggleInventory(inv);
            return true;
        }
```

6.17.3.7 ReturnShapedRecipieItem() [1/2]

```
Item BeeGame.Blocks.CraftingTable.ReturnShapedRecipieItem (
             Item [] items )
```

Makes a shaped crafting recipie from the given items and return if it is a recipie

Parameters

items	Items to make the recipie from

Returns

A Item if the recipe exists

Definition at line 46 of file CraftingTable.cs.

```
        {
            var recipie = "";

            for (int i = 0; i < items.Length; i++)
            {
                if (items[i] == null)
                {
                    recipie += "0:";
                    continue;
                }

                recipie += $"{items[i].GetItemID()}:";
            }

            return ReturnShapedRecipieItem(recipie);
        }
```

6.17.3.8 ReturnShapedRecipieItem() [2/2]

```
virtual Item BeeGame.Blocks.CraftingTable.ReturnShapedRecipieItem (
             string recipie )  [virtual]
```

Returns a crafting recipie from a given recipie

Parameters

recipie	

Returns

A Item if the recipie exists

Virtual incase needs to be overriden by a different crafting system

Definition at line 77 of file CraftingTable.cs.

```
{
    return BeeGame.Core.Dictionarys.
CraftingRecipies.GetShapedRecipeItem(recipie);
}
```

6.17.3.9 ReturnShapelessRecipieItem()

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

Definition at line 64 of file CraftingTable.cs.

```
{
    return CraftingRecipies.GetShaplessRecipieResult(items)
;
}
```

6.17.3.10 TexturePosition()

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

Returns the texture for the apiary Block

Parameters

direction	Direction of thhe desired face

Returns

Tile with the textture coordinates of the Block texture

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

Reimplemented from BeeGame.Items.Item.

Definition at line 156 of file CraftingTable.cs.

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

6.17.4 Member Data Documentation

6.17.4.1 ID

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

This blocks ID

Definition at line 27 of file CraftingTable.cs.

6.17.4.2 myGameobject

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

The GameObject for this block

Definition at line 22 of file CraftingTable.cs.

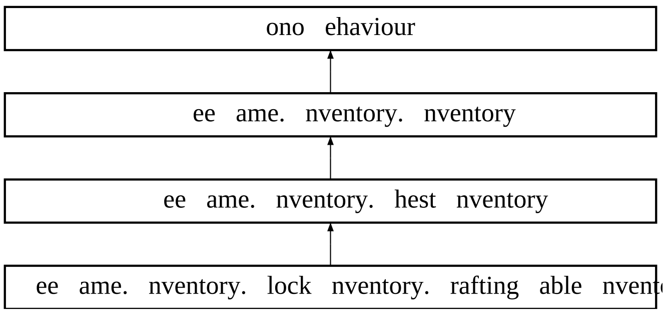
The documentation for this class was generated from the following file:

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

6.18 BeeGame.Inventory.BlockInventory.CraftingTableInventory Class Reference

Invnetory for the CraftingTable Block

Inheritance diagram for BeeGame.Inventory.BlockInventory.CraftingTableInventory:


ono ehaviour
ee ame. nventory. nventory
ee ame. nventory. hest nventory
ee ame. nventory. lock nventory. rafting able nvent

Public Member Functions

- delegate void ItemRemovedFromResult ()

  Makes the delegate
- virtual void CheckShapedRecipie ()

  Check in the recpie in the grid for a shaped crafting recipie
- virtual void CheckShapelessRecipie ()

  Check in the recipie grid for a shapless crafting recipie
- void CraftedItemRemoved ()

  Removes the items form the crafting grid one an item has been removed from the crafting result slot, Called via the result delegate from InventorySlot.OnPointerClick(UnityEngine.EventSystems.PointerEventData)
- virtual void DropItemsFromInventory ()

  Removes all Items from the inventory when it is closed
- override void ToggleInventory (Inventory inv)

  Opens/Closes the inventory
- override void SetChestInventory (string invName="Workbench")

  Set the size of the Inventory
- override void AddItemToSlots (int slotIndex, Item item)

  Adds an item to a InventorySlot
- override void SaveInv ()

  Oerriden so the inventory is not saved in any way

Public Attributes

- ItemRemovedFromResult result

  Holds the method for the delegate to call

Protected Member Functions

- void Start ()

  Sets the size of the inventory
- void Update ()

  Updates the base and checks crafting recipies
- void OnDestroy ()

  Ensureing no memory leaks occur due to the delegate

Additional Inherited Members

6.18.1 Detailed Description

Invnetory for the CraftingTable Block

Definition at line 14 of file CraftingTableInventory.cs.

6.18.2 Member Function Documentation

6.18.2.1 AddItemToSlots()

```
override void BeeGame.Inventory.BlockInventory.CraftingTableInventory.AddItemToSlots (
             int slotIndex,
             Item item )  [virtual]
```

Adds an item to a InventorySlot

Parameters

slotIndex	InventorySlot.slotIndex to add the items to
item	Item to add

Overriden so serialization does not occur

Reimplemented from BeeGame.Inventory.Inventory.

Definition at line 179 of file CraftingTableInventory.cs.

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

6.18.2.2 CheckShapedRecipie()

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

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

Definition at line 69 of file CraftingTableInventory.cs.

```
        {
            var items = new Item[9];

            for (int i = 0; i < items.Length; i++)
            {
                items[i] = base.items.itemsInInventory[i];
            }

            //* if it is a recipie put the result into the crafting result slot
            Item item = ((CraftingTable)myblock).ReturnShapedRecipieItem(items);
            if (item != base.items.itemsInInventory[9])
                base.items.itemsInInventory[9] = item;
        }
```

6.18.2.3 CheckShapelessRecipie()

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

Check in the recipie grid for a shapless crafting recipie

Definition at line 87 of file CraftingTableInventory.cs.

```
        {
            var items = new Item[9];

            for (int i = 0; i < items.Length; i++)
            {
                items[i] = base.items.itemsInInventory[i];
            }

            Item item = ((CraftingTable)myblock).ReturnShapelessRecipieItem(items);
            if (item != base.items.itemsInInventory[9])
                base.items.itemsInInventory[9] = item;
        }
```

6.18.2.4 CraftedItemRemoved()

```
void BeeGame.Inventory.BlockInventory.CraftingTableInventory.CraftedItemRemoved ( )
```

Removes the items form the crafting grid one an item has been removed from the crafting result slot, Called via the result delegate from InventorySlot.OnPointerClick(UnityEngine.EventSystems.PointerEventData)

Definition at line 104 of file CraftingTableInventory.cs.

```
        {
            if (items.itemsInInventory[9] != null)
            {
                Events.CallShapedRecipieCraftedEvent(items.itemsInInventory[9]);
                for (int i = 0; i < 9; i++)
                {
                    if (items.itemsInInventory[i] != null)
                        items.itemsInInventory[i].itemStackCount -= 1;
                }
            }
        }
```

6.18.2.5 DropItemsFromInventory()

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

Removes all Items from the inventory when it is closed

Called by the output invenotry slot as it is a button

Definition at line 125 of file CraftingTableInventory.cs.

```
        {
            //* looks at every item in the crafting grid
            for (int i = 0; i < 9; i++)
            {
                if (items.itemsInInventory[i] != null)
                {
                    //* spwns it and removes it from the inventory if an items exists within
                    for (int j = 0; j < items.itemsInInventory[i].itemStackCount; j++)
                    {
                        items.itemsInInventory[i].SpawnItem((THVector3)this.transform.position + new THVector3(0, 1, 0));
                    }
                    items.itemsInInventory[i] = null;
                }
            }
        }
```

6.18.2.6 ItemRemovedFromResult()

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

Makes the delegate

6.18.2.7 OnDestroy()

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

Ensureing no memory leaks occur due to the delegate

Definition at line 58 of file CraftingTableInventory.cs.

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

6.18.2.8 SaveInv()

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

Oerriden so the inventory is not saved in any way

Reimplemented from BeeGame.Inventory.Inventory.

Definition at line 187 of file CraftingTableInventory.cs.

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

6.18.2.9 SetChestInventory()

```
override void BeeGame.Inventory.BlockInventory.CraftingTableInventory.SetChestInventory (
             string invName = "Workbench" )  [virtual]
```

Set the size of the Inventory

Parameters

invName	Workbench

overridden here so that no attemp is made to deserialize the inventory helping with performance

Reimplemented from BeeGame.Inventory.ChestInventory.

Definition at line 164 of file CraftingTableInventory.cs.

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

6.18.2.10 Start()

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

Sets the size of the inventory

Definition at line 31 of file CraftingTableInventory.cs.

```
        {
            SetChestInventory();
            result = CraftedItemRemoved;
        }
```

6.18.2.11 ToggleInventory()

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

Opens/Closes the inventory

Parameters

inv	The inventory to toggle

Reimplemented from BeeGame.Inventory.ChestInventory.

Definition at line 148 of file CraftingTableInventory.cs.

```
        {
            base.ToggleInventory(inv);

            //* if the inventory was closed drop the items within
            if (!inventory.activeInHierarchy)
               DropItemsFromInventory();
        }
```

6.18.2.12 Update()

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

Updates the base and checks crafting recipies

Definition at line 41 of file CraftingTableInventory.cs.

```
        {
            UpdateChestInventory();

            if (inventory.activeInHierarchy)
            {
                CheckShapedRecipie();

                //* checks for shapless recipies second
                if(items.itemsInInventory[9] == null)
                    CheckShapelessRecipie();
            }
        }
```

6.18.3 Member Data Documentation

6.18.3.1 result

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

Holds the method for the delegate to call

Definition at line 24 of file CraftingTableInventory.cs.

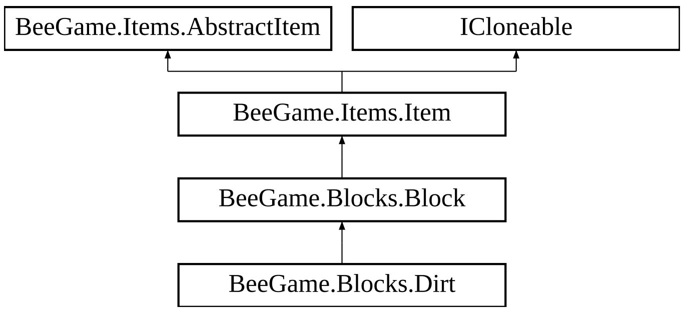
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/BlockInventory/Crafting↩ TableInventory.cs

6.19 BeeGame.Blocks.Dirt Class Reference

Dirt Block

Inheritance diagram for BeeGame.Blocks.Dirt:



Public Member Functions

- Dirt ()

  Constructor
- override Sprite GetItemSprite ()

  Returns the sprite for the item
- override Tile TexturePosition (Direction direction)

  Position of the dirt texture in the atlas
- override int GetHashCode ()

  Base ID of the block
- override string ToString ()

  Returns the name and ID of the block as a string

Static Public Attributes

- static new int ID => 3

Additional Inherited Members

6.19.1 Detailed Description

Dirt Block

Definition at line 13 of file Dirt.cs.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 Dirt()

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

Constructor

Definition at line 21 of file Dirt.cs.

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

6.19.3 Member Function Documentation

6.19.3.1 GetHashCode()

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

Base ID of the block

Returns

5

Reimplemented from BeeGame.Blocks.Block.

Definition at line 48 of file Dirt.cs.

```
00049         {
00050             return ID;
00051         }
```

6.19.3.2 GetItemSprite()

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

Returns the sprite for the item

Returns

Sprite for this item

Reimplemented from BeeGame.Blocks.Block.

Definition at line 25 of file Dirt.cs.

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

6.19.3.3 TexturePosition()

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

Position of the dirt texture in the atlas

Parameters

direction	

Returns

Reimplemented from BeeGame.Items.Item.

Definition at line 37 of file Dirt.cs.

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

6.19.3.4 ToString()

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

Returns the name and ID of the block as a string

Returns

A nicely formatted string

Definition at line 57 of file Dirt.cs.

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

6.19.4 Member Data Documentation

6.19.4.1 ID

```
new int BeeGame.Blocks.Dirt.ID => 3  [static]
```

Definition at line 15 of file Dirt.cs.

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

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

6.20 BeeGame.Core.Events Class Reference

Public Member Functions

- delegate void ItemCraftedEvent (Item item)

Static Public Member Functions

- static void CallShapedRecipieCraftedEvent (Item item)
- static void CallShaplessRecipirCraftedEvent (Item item)
- static void CallBeeCraftedEvent (Item item)

Static Public Attributes

- static ItemCraftedEvent shapedRecipieCrafted
- static ItemCraftedEvent shaplessRecipieCrafted
- static ItemCraftedEvent beeCraftedEvent

6.20.1 Detailed Description

Definition at line 10 of file Events.cs.

6.20.2 Member Function Documentation

6.20.2.1 CallBeeCraftedEvent()

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

6.20.2.2 CallShapedRecipieCraftedEvent()

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

6.20.2.3 CallShaplessRecipirCraftedEvent()

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

6.20.2.4 ItemCraftedEvent()

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

6.20.3 Member Data Documentation

6.20.3.1 beeCraftedEvent

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

Definition at line 15 of file Events.cs.

6.20.3.2 shapedRecipieCrafted

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

Definition at line 13 of file Events.cs.

6.20.3.3 shaplessRecipieCrafted

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

Definition at line 14 of file Events.cs.

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

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Events.cs

6.21 BeeGame.Core.Extensions Class Reference

Static Public Member Functions

- static T CloneObject< T > (this T obj)

  Allows the copying of a class by value useing reflection
- static Sprite ColourSprite (this Sprite sprite, Color colour, Color[ ] coloursToAvoid=null, bool setTransparent←
ToWhite=false)

  Will colour the sprite given a colour and optionaly colours to avoid
- static void SpawnItem (this Item item, THVector3 position, Quaternion rotation=new Quaternion())

6.21.1 Detailed Description

Definition at line 12 of file Extensions.cs.

6.21.2 Member Function Documentation

6.21.2.1 CloneObject< T >()

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

Allows the copying of a class by value useing reflection

Parameters

obj	Object to copy

Returns

a new object with all values copyed

Mush faster than the serialize method however alot more complicated

Definition at line 22 of file Extensions.cs.

```
        {
            //* gets the tyoe of the given object
            Type typeSource = obj.GetType();

            //* makes a new object of type T
            T objTarget = (T)Activator.CreateInstance(typeSource);

            //* gets the properties in T
            PropertyInfo[] propertyInfo = typeSource.GetProperties(BindingFlags.Public | BindingFlags.
      NonPublic | BindingFlags.Instance);

            //* applies the properties in T to the new type T object
            foreach (var property in propertyInfo)
            {
                if (property.CanWrite)
                {
                    //* if the propertly is a value just set it
                    if (property.PropertyType.IsValueType || property.PropertyType.IsEnum || property.
PropertyType.Equals(typeof(string)))
                    {
                        property.SetValue(objTarget, property.GetValue(obj, null), null);
                    }
                    else
                    {
                        //* if the propertly is not a value type this function will need to be called
 recursivly as it could also have non value type veriables
                        object propertyValue = property.GetValue(obj, null);

                        if (propertyValue == null)
                        {
                            property.SetValue(objTarget, null, null);
                        }
                        else
                        {
                            property.SetValue(objTarget, propertyValue.CloneObject(), null);
                        }
                    }
                }
            }

            //* gets all of the field in T
            FieldInfo[] fieldInfo = typeSource.GetFields();

            //* applies all of the fiels of T to the new object if type T in the same manor that the
 properites are applied
            foreach (var field in fieldInfo)
            {
                if(field.FieldType.IsValueType || field.FieldType.IsEnum || field.FieldType.Equals(typeof(
string)))
                {
                    field.SetValue(objTarget, field.GetValue(obj));
                }
                else
                {
                    object fieldValue = field.GetValue(obj);

                    if(fieldValue == null)
                    {
                        field.SetValue(objTarget, null);
                    }
                    else
                    {
                        field.SetValue(objTarget, field.CloneObject());
                    }
                }
            }

            return objTarget;
        }
```

6.21.2.2 ColourSprite()

```
static Sprite BeeGame.Core.Extensions.ColourSprite (
            this Sprite sprite,
            Color colour,
            Color [] coloursToAvoid = null,
            bool setTransparentToWhite = false )  [static]
```

Will colour the sprite given a colour and optionaly colours to avoid

Parameters

sprite	Sprite to colour
colour	Colour to set the sprite to
coloursToAvoid	Colours to avoid, Optional
setTransparentToWhite	Should transparent value to set wo white, Default true

Returns

Definition at line 96 of file Extensions.cs.

```
        {
            Texture2D tex = new Texture2D((int)sprite.rect.width, (int)sprite.rect.height)
            {
                filterMode = FilterMode.Point,
                wrapMode = TextureWrapMode.Clamp
            };

            //* sets the teture pixels to the pixels of teh sprite so the original sprite is not modified
            tex.SetPixels(sprite.texture.GetPixels());

            for (int x = 0; x < tex.width; x++)
            {
                for (int y = 0; y < tex.height; y++)
                {
                    //* if we dont have to avoid any colours set the pixel
                    if (coloursToAvoid == null)
                    {
                        tex.SetPixel(x, y, tex.GetPixel(x, y) * colour);
                    }
                    else
                    {
                        for (int i = 0; i < coloursToAvoid.Length; i++)
                        {
                            //* if this colour should be avoided skip this iteration of the loop and move on
                            if (tex.GetPixel(x, y) == coloursToAvoid[i])
                                goto Skip;
                        }

                        tex.SetPixel(x, y, tex.GetPixel(x, y) * colour);
                    }

                    //* if transparent pixels should be set to white do that
                    if (setTransparentToWhite && tex.GetPixel(x, y).a == 0)
                        tex.SetPixel(x, y, Color.white);

                    Skip:
                        continue;
                }
            }

            //* apply the new texture with its colours
            tex.Apply();

            //* return the Texture2D as a sprite
            return Sprite.Create(tex, new Rect(0, 0, tex.width, tex.height), new THVector2(0.5f, 0.5f));
        }
```

6.21.2.3 SpawnItem()

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

Definition at line 144 of file Extensions.cs.

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

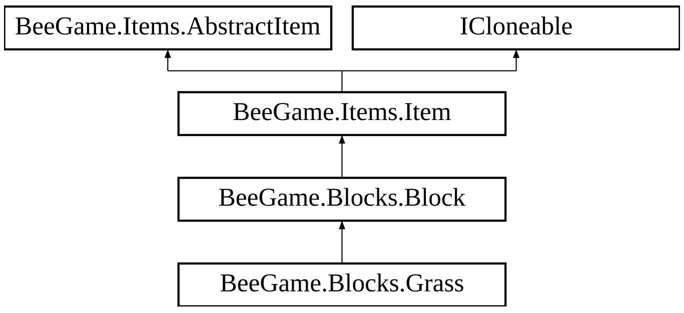
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Extensions.cs

6.22 BeeGame.Blocks.Grass Class Reference

Grass Block

Inheritance diagram for BeeGame.Blocks.Grass:



Public Member Functions

- Grass ()

    Constructor also sets teh items name
- override Sprite GetItemSprite ()

    Returns the sprite for the item
- override void UpdateBlock (int x, int y, int z, Chunk chunk)

    Will turn this Block into a Dirt block if another block is above it
- override Tile TexturePosition (Direction direction)

    Texture position of the Block face
- override int GetHashCode ()

    The Base id for the block
- override string ToString ()

    REturns the name and value for the block as a string

Static Public Attributes

- static new int ID => 4

Additional Inherited Members

6.22.1 Detailed Description

Grass Block

Definition at line 14 of file Grass.cs.

6.22.2 Constructor & Destructor Documentation

6.22.2.1 Grass()

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

Constructor also sets teh items name

Definition at line 22 of file Grass.cs.

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

6.22.3 Member Function Documentation

6.22.3.1 GetHashCode()

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

The Base id for the block

Returns

4

Reimplemented from BeeGame.Blocks.Block.

Definition at line 82 of file Grass.cs.

```
        {
            return ID;
        }
```

6.22.3.2 GetItemSprite()

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

Returns the sprite for the item

Returns

Sprite for this item

Reimplemented from BeeGame.Blocks.Block.

Definition at line 26 of file Grass.cs.

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

6.22.3.3 TexturePosition()

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

Texture position of the Block face

Parameters

direction	Direction of the block face

Returns

Texture positon as a Tile

Reimplemented from BeeGame.Items.Item.

Definition at line 51 of file Grass.cs.

```
        {
            //All textures are on the dame Y value for the texture atlas so Y can be set
            Tile tile = new Tile()
            {
                y = 9
            };

            switch (direction)
            {
                //if we want the top face return the full grass texture
                case Direction.UP:
                    tile.x = 3;
                    return tile;
                //if we want the bottom face return the dirt texture
                case Direction.DOWN:
                    tile.x = 2;
                    return tile;
                //return the 1/2 grass testure if a side face is wanted
                default:
                    tile.x = 4;
                    return tile;
            }
        }
```

6.22.3.4 ToString()

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

REturns the name and value for the block as a string

Returns

A nicely formatted string

Definition at line 91 of file Grass.cs.

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

6.22.3.5 UpdateBlock()

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

Will turn this Block into a Dirt block if another block is above it

Parameters

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

Reimplemented from BeeGame.Blocks.Block.

Definition at line 40 of file Grass.cs.

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

6.22.4 Member Data Documentation

6.22.4.1 ID

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

Definition at line 16 of file Grass.cs.

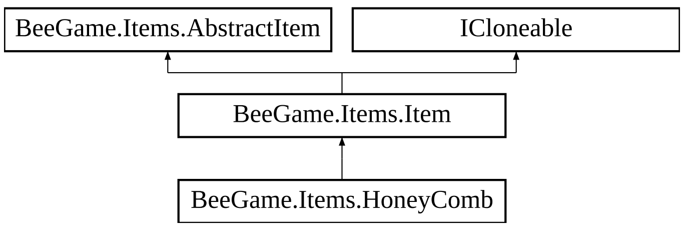
The documentation for this class was generated from the following file:

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

6.23 BeeGame.Items.HoneyComb Class Reference

Honey comb item produced by bees

Inheritance diagram for BeeGame.Items.HoneyComb:



Public Member Functions

- HoneyComb ()

    Make the Item from no arguments giveing it the default honey comb value HoneyCombType.HONEY
- HoneyComb (HoneyCombType type)

    Makes a HoneyComb for the given HoneyCombType
- override Sprite GetItemSprite ()

    Retuens the sprite for the this of the correct colour
- override GameObject GetGameObject ()

    Returns the game object for this and gives the object the correct colouring
- override string GetItemID ()

    Makes the item ID. For this it is the Normal ID \ the int value of the type this comb is
- override int GetHashCode ()

    Returns the hashcode for this Item

Static Public Attributes

- static new int ID => 12

Properties

- HoneyCombType type `[get, set]`

    The type of comb this is, HoneyCombType
- Color CombColour `[get]`

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

Private Attributes

- Sprite itemSprite

    The Sprite for this honey comb

Additional Inherited Members

6.23.1 Detailed Description

Honey comb item produced by bees

Definition at line 14 of file HoneyComb.cs.

6.23.2 Constructor & Destructor Documentation

6.23.2.1 HoneyComb() [1/2]

`BeeGame.Items.HoneyComb.HoneyComb ( )`

Make the Item from no arguments giveing it the default honey comb value HoneyCombType.HONEY

Definition at line 46 of file HoneyComb.cs.

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

6.23.2.2 HoneyComb() [2/2]

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

Makes a HoneyComb for the given HoneyCombType

Parameters

type	that this comb is

Definition at line 56 of file HoneyComb.cs.

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

6.23.3 Member Function Documentation

6.23.3.1 GetGameObject()

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

Returns the game object for this and gives the object the correct colouring

Returns

GameObject for this

Reimplemented from BeeGame.Items.Item.

Definition at line 77 of file HoneyComb.cs.

```
        {
            GameObject obj = PrefabDictionary.GetPrefab("HoneyComb");
            //* cannot acess the instance material from here have to do it on the obejct
            obj.GetComponent<ApplyColour>().colour = CombColour;
            return obj;
        }
```

6.23.3.2 GetHashCode()

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

Returns the hashcode for this Item

Returns

8

Implements BeeGame.Items.AbstractItem.

Definition at line 100 of file HoneyComb.cs.

```
        {
            return ID;
        }
```

6.23.3.3 GetItemID()

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

Makes the item ID. For this it is the Normal ID \ the int value of the type this comb is

Returns

Item ID as a string

Implements BeeGame.Items.AbstractItem.

Definition at line 89 of file HoneyComb.cs.

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

6.23.3.4 GetItemSprite()

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

Retuens the sprite for the this of the correct colour

Returns

Sprite

Reimplemented from BeeGame.Items.Item.

Definition at line 68 of file HoneyComb.cs.

```
        {
            return itemSprite ?? (itemSprite =
      SpriteDictionary.GetSprite("HoneyComb").ColourSprite(
      CombColour));
        }
```

6.23.4 Member Data Documentation

6.23.4.1 ID

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

Definition at line 39 of file HoneyComb.cs.

6.23.4.2 itemSprite

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

The Sprite for this honey comb

Definition at line 37 of file HoneyComb.cs.

6.23.5 Property Documentation

6.23.5.1 CombColour

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

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

Definition at line 26 of file HoneyComb.cs.

6.23.5.2 type

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

The type of comb this is, HoneyCombType

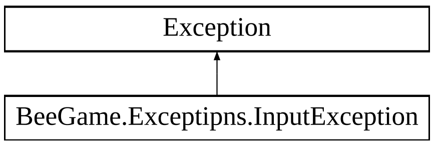
Definition at line 20 of file HoneyComb.cs.

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

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

6.24 BeeGame.Exceptipns.InputException Class Reference

Inheritance diagram for BeeGame.Exceptipns.InputException:



Public Member Functions

- InputException ()
- InputException (string message)
- InputException (string message, Exception innerException)

6.24.1 Detailed Description

Definition at line 8 of file InputException.cs.

6.24.2 Constructor & Destructor Documentation

6.24.2.1 InputException() [1/3]

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

Definition at line 10 of file InputException.cs.

```
                                 : base()
         {

         }
```

6.24.2.2 InputException() [2/3]

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

Definition at line 15 of file InputException.cs.

```
                                               : base(message)
         {

         }
```

6.24.2.3 InputException() [3/3]

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

Definition at line 20 of file InputException.cs.

```
                                                                   : base(message, innerException)
        {

        }
```

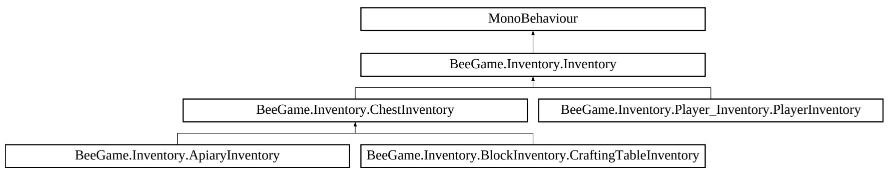
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Exceptipns/InputException.cs

6.25 BeeGame.Inventory.Inventory Class Reference

Base class for all inventorys in the game

Inheritance diagram for BeeGame.Inventory.Inventory:


MonoBehaviour
BeeGame.Inventory.Inventory
BeeGame.Inventory.ChestInventory
BeeGame.Inventory.Player_Inventory.PlayerInventory
BeeGame.Inventory.ApiaryInventory
BeeGame.Inventory.BlockInventory.CraftingTableInventory

Public Member Functions

- bool InventorySet ()
  Is the inventory set?
- void SetInventorySize (int inventorySize)
  Sets the inventory soze to the number of slots in the invnetory
- void SetAllItems (ItemsInInventory items)
  Sets the items to the given ItemsInInventory
- void UpdateBase ()
  Things in the inventory that should be updated
- virtual void ToggleInventory (Inventory inv)
- virtual void SaveInv ()
  Saves the inventory
- ItemsInInventory GetAllItems ()
  Gets all of the items in the invntory
- virtual void AddItemToSlots (int slotIndex, Item item)
  Adds the given item to the inventory in the given slotIndex
- bool AddItemToInventory (Item item)
  Add an item to the inventory

Public Attributes

- ItemsInInventory items

  Items in the invemtory
- InventorySlot [ ] slots

  Slots in the inventory
- string inventoryName = ""

  Name of this inventory
- Blocks.Block myblock

  The block class that this inventory is part of

Protected Attributes

- bool thisInventoryOpen = false

  is this inventory open?

Package Attributes

- Item floatingItem

  Item that is currenty being moved

Private Member Functions

- void DrawItemAtCursor ()

  Draws the floatingItems Item.GetItemSprite() at the mouse position
- void PutItemsInSlots ()

  Sets an Item in the ItemsInInventory.itemsInInventory array to a InventorySlot.item

Private Attributes

- GameObject spriteAtCursor

  The sprite at the cursor

6.25.1 Detailed Description

Base class for all inventorys in the game

Definition at line 11 of file Inventory.cs.

6.25.2 Member Function Documentation

6.25.2.1 AddItemToInventory()

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

Add an item to the inventory

Parameters

item	Item to add

Returns

true if item wasa added

Definition at line 176 of file Inventory.cs.

```
        {
            return items.AddItem(item);
        }
```

6.25.2.2 AddItemToSlots()

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

Adds the given *item* to the inventory in the given *slotIndex*

Parameters

slotIndex	Slot to add item to
item	Item to add

Reimplemented in BeeGame.Inventory.BlockInventory.CraftingTableInventory.

Definition at line 164 of file Inventory.cs.

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

6.25.2.3 DrawItemAtCursor()

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

Draws the floatingItems Item.GetItemSprite() at the mouse position

Definition at line 95 of file Inventory.cs.

```
        {
            if(floatingItem != null)
            {
                if (spriteAtCursor == null)
                {
                    spriteAtCursor = Instantiate(PrefabDictionary.GetPrefab("ItemIcon"));
                    spriteAtCursor.GetComponentInChildren<UnityEngine.UI.Image>().sprite = floatingItem.GetItemSprite();
                }
                //* will update a the sprite of in item is swapped between a slot and teh floating item if the previous item wasnt put into a slot first
                else if(spriteAtCursor != null)
                {
                    spriteAtCursor.GetComponentInChildren<UnityEngine.UI.Image>().sprite = floatingItem.GetItemSprite();
                }

                spriteAtCursor.transform.GetChild(0).position = Input.mousePosition;
            }
            else
            {
                Destroy(spriteAtCursor);
            }
        }
```

6.25.2.4 GetAllItems()

`ItemsInInventory BeeGame.Inventory.Inventory.GetAllItems ( )`

Gets all of the items in the invntory

Returns

All of the items in the inventory as ItemsInInventory

Definition at line 154 of file Inventory.cs.

```
        {
            return items;
        }
```

6.25.2.5 InventorySet()

`bool BeeGame.Inventory.Inventory.InventorySet ( )`

Is the inventory set?

Returns

true if items == null

Definition at line 52 of file Inventory.cs.

```
        {
            if (items == null)
                return true;

            return false;
        }
```

6.25.2.6 PutItemsInSlots()

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

Sets an Item in the ItemsInInventory.itemsInInventory array to a InventorySlot.item

Definition at line 139 of file Inventory.cs.

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

6.25.2.7 SaveInv()

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

Saves the inventory

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

Reimplemented in BeeGame.Inventory.BlockInventory.CraftingTableInventory.

Definition at line 131 of file Inventory.cs.

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

6.25.2.8 SetAllItems()

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

Sets the items to the given ItemsInInventory

Parameters

items	Items to set this inventory to

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

Definition at line 76 of file Inventory.cs.

```
        {
            this.items = items;
        }
```

6.25.2.9 SetInventorySize()

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

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

Parameters

inventorySize	

Definition at line 64 of file Inventory.cs.

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

6.25.2.10 ToggleInventory()

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

Reimplemented in BeeGame.Inventory.BlockInventory.CraftingTableInventory, and BeeGame.Inventory.Chest↩ Inventory.

Definition at line 120 of file Inventory.cs.

```
        {
            throw new NotImplementedException();
        }
```

6.25.2.11 UpdateBase()

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

Things in the inventory that should be updated

Definition at line 86 of file Inventory.cs.

```
        {
            PutItemsInSlots();
            DrawItemAtCursor();
        }
```

6.25.3 Member Data Documentation

6.25.3.1 floatingItem

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

Item that is currenty being moved

Definition at line 25 of file Inventory.cs.

6.25.3.2 inventoryName

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

Name of this inventory

Definition at line 29 of file Inventory.cs.

6.25.3.3 items

`ItemsInInventory BeeGame.Inventory.Inventory.items`

Items in the invemtory

Definition at line 17 of file Inventory.cs.

6.25.3.4 myblock

`Blocks.Block BeeGame.Inventory.Inventory.myblock`

The block class that this inventory is part of

currently only used for the Blocks.Apiary but could be used so that block inventorys are stord in the chunk and not in a seperate file

Definition at line 44 of file Inventory.cs.

6.25.3.5 slots

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

Slots in the inventory

Definition at line 21 of file Inventory.cs.

6.25.3.6 spriteAtCursor

```
GameObject BeeGame.Inventory.Inventory.spriteAtCursor  [private]
```

The sprite at the cursor

Definition at line 37 of file Inventory.cs.

6.25.3.7 thisInventoryOpen

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

is this inventory open?

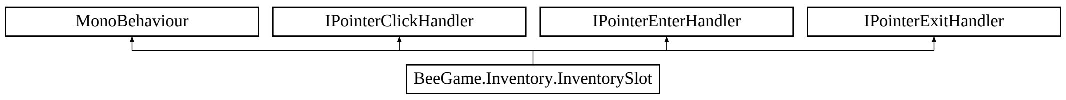
Definition at line 33 of file Inventory.cs.

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

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

6.26 BeeGame.Inventory.InventorySlot Class Reference

Inheritance diagram for BeeGame.Inventory.InventorySlot:



Public Member Functions

- void OnPointerClick (PointerEventData eventData)

  Allows the player to interact with the item slot
- void OnPointerEnter (PointerEventData eventData)

  Makes the text object when the cursor is over the slot
- void OnPointerExit (PointerEventData eventData)

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

Public Attributes

- Item item

  The item this slot has in it
- Inventory myInventory

  The Inventory this slot is in
- GameObject itemText

  If the slot currently has the item text object made this will be not null otherwise it is null
- bool selectedSlot = false

  Is this slot currently the selected slot in the hotbar?
- bool itemsCanBeInserted = true

  Can items be inserted into this slot by the player

Package Attributes

- int slotIndex

  The slot in the inventory this is

Private Member Functions

- void Update ()

  Updates the slot
- void UpdateIcon ()

  Applies the correct icon to the slot depending on what is in the slot
- void AddToFloatingItem ()

  Add items from the slot to the Inventory.floatingItem
- void AddToSlot (int numerToAdd)

  Adds a number to items into the slot
- void SplitStack ()

  Halfs a Item.itemStackCount between the slot and the Inventory.floatingItem
- void SwapItems ()

  Swaps the Item in the Inventory.floatingItem with the slots item
- void CheckFloatingItem ()

  Checks if the Inventory.floatingItem should be null
- void CheckItem ()

  checks that the item is valid
- void OnDisable ()

  Destroys the item text when the inventory is closed

6.26.1 Detailed Description

Definition at line 10 of file InventorySlot.cs.

6.26.2 Member Function Documentation

6.26.2.1 AddToFloatingItem()

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

Add items from the slot to the Inventory.floatingItem

Definition at line 184 of file InventorySlot.cs.

```
        {
            //* if the whole stack can be added do it and move on
            if(myInventory.floatingItem.itemStackCount +
    item.itemStackCount <= item.maxStackCount)
            {
                myInventory.floatingItem.itemStackCount +=
    item.itemStackCount;

                item = null;

                myInventory.AddItemToSlots(slotIndex,
    item);

                return;
            }

            //* if the whole stack cannot be added calculate how many need to be removed from the slots
     item stack
            item.itemStackCount -= (item.maxStackCount -
    myInventory.floatingItem.itemStackCount);
            //* set the floating item to the max stack count
            myInventory.floatingItem.itemStackCount =
    item.maxStackCount;

            myInventory.AddItemToSlots(slotIndex,
    item);
        }
```

6.26.2.2 AddToSlot()

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

Adds a number to items into the slot

Parameters

numerToAdd	Numebr or items to add to the slot

Definition at line 210 of file InventorySlot.cs.

```
        {
            //* if the item in the slot is null create it
            if (item == null)
            {
                item = myInventory.floatingItem.CloneObject();
                item.itemStackCount = 0;
            }

            //* add to number to add to the stack count
            item.itemStackCount += numerToAdd;

            //* if the stack count is now larger than it should be dont let it be
            if (item.itemStackCount > item.maxStackCount)
            {
                item.itemStackCount = item.maxStackCount;
            }

            //* remove the numebr if items form the floating item then check the floating item is not null
            myInventory.floatingItem.itemStackCount -= numerToAdd;
            CheckFloatingItem();
            //* save the inventory changes
            myInventory.AddItemToSlots(slotIndex,
    item);
        }
```

6.26.2.3 CheckFloatingItem()

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

Checks if the Inventory.floatingItem should be null

Definition at line 275 of file InventorySlot.cs.

```
        {
            if(myInventory.floatingItem.itemStackCount <= 0)
            {
                myInventory.floatingItem = null;
            }
        }
```

6.26.2.4 CheckItem()

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

checks that the item is valid

Definition at line 287 of file InventorySlot.cs.

```
        {
            if (item != null && myInventory != null)
            {
                if (item.itemStackCount == 0 || item.
     itemName == "TestItem")
                {
                    myInventory.items.itemsInInventory[
     slotIndex] = null;
                    Destroy(itemText);
                }
            }
        }
```

6.26.2.5 OnDisable()

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

Destroys the item text when the inventory is closed

Definition at line 329 of file InventorySlot.cs.

```
        {
            Destroy(itemText);
        }
```

6.26.2.6 OnPointerClick()

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

Allows the player to interact with the item slot

Parameters

eventData	Right or Left click

Called by the unity event handler when the slot is clicked on

Definition at line 83 of file InventorySlot.cs.

```
        {
            if (myInventory.floatingItem != null)
            {
                //* Left click moves whole stacks of items
                if (eventData.button == PointerEventData.InputButton.Left)
                {
                    //* If the item in the slot is empty put the floating item into it then clear it and the slot can have items inserted
                    if (item == null && itemsCanBeInserted)
                    {
                        item = myInventory.floatingItem;
                        myInventory.floatingItem = null;
                        myInventory.AddItemToSlots(slotIndex, item);
                        return;
                    }
                    //* if the items are the same
                    if(myInventory.floatingItem == item && itemsCanBeInserted)
                    {
                        //* if the item in the inventoys stack count + the floating items stack count is less than the max stack count
                        if (myInventory.floatingItem.itemStackCount + item.itemStackCount <= item.maxStackCount)
                        {
                            AddToSlot(myInventory.floatingItem.itemStackCount);
                            return;
                        }
                        //* if the item stack added is larger than the max count add as many as you can and move on
                        else
                        {
                            AddToSlot(item.maxStackCount - item.itemStackCount);
                            return;
                        }
                    }
                    //* if the tiems are the same but items cannot be inserted into the slot add as many items as you
                    //* can from the slot to the floating item
                    else if(myInventory.floatingItem == item && !itemsCanBeInserted)
                    {
                        AddToFloatingItem();
                        {
                            if (myInventory is BlockInventory.CraftingTableInventory c)
                                c.result.Invoke();
                        }
                        return;
                    }
                    //* If the items were not == swap them
                    else
                    {
                        //* only if items can be inserted into the slot
                        if(itemsCanBeInserted)
                            SwapItems();
                        return;
                    }
                }
                else if(eventData.button == PointerEventData.InputButton.Right)
                {
                    //* if the item in slot is null add 1 from the floating item to it
                    if(item == null && itemsCanBeInserted)
                    {
                        AddToSlot(1);
                        return;
                    }
                    //* if the items are the same add 1 from the floating item to this item
                    else if(item == myInventory.floatingItem && itemsCanBeInserted)
                    {
                        AddToSlot(1);
                        return;
                    }
                }
            }
            //* if the floating item is null
            else
            {
                //* add 1/2 of the stack into the floating item if right click was pressed
                if(eventData.button == PointerEventData.InputButton.Right)
                {
                    SplitStack();

                    //* blocks removed some weird name confliction
                    {
                        if (myInventory is BlockInventory.CraftingTableInventory c)
                            c.result.Invoke();
                    }

                    return;
                }

                //* otherwie add the items into the floating item slot
                SwapItems();
                //* ^ does not need to check that the slot cannot be inserted into as null be being inserted because the floating item is null

                {
                    if (myInventory is BlockInventory.CraftingTableInventory c)
                        c.result.Invoke();
                }

                return;
            }

        }
```

6.26.2.7 OnPointerEnter()

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

Makes the text object when the cursor is over the slot

Parameters

eventData	Not used but required for the interface

Definition at line 304 of file InventorySlot.cs.

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

6.26.2.8 OnPointerExit()

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

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

Parameters

eventData	Not used but required for the interface

Definition at line 321 of file InventorySlot.cs.

```
        {
            Destroy(itemText);
        }
```

6.26.2.9 SplitStack()

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

Halfs a Item.itemStackCount between the slot and the Inventory.floatingItem

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

Definition at line 241 of file InventorySlot.cs.

```
        {
            myInventory.floatingItem = item.CloneObject();
            int give = (item.itemStackCount + 1) / 2;
            myInventory.floatingItem.itemStackCount = give;
            item.itemStackCount -= give;

            if (item.itemStackCount <= 0)
                item = null;

            myInventory.AddItemToSlots(slotIndex,
      item);
            Destroy(itemText);
        }
```

6.26.2.10 SwapItems()

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

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

Definition at line 258 of file InventorySlot.cs.

```
        {
            //* temp copy of the item
            Item temp = myInventory.floatingItem;
            //* sets the floating item
            myInventory.floatingItem = item;
            //* sets the item that was in the floating item to the item in the the slot
            item = temp;
            //* Saves the changes to the inventory
            myInventory.AddItemToSlots(slotIndex,
      item);
            //* destroys the text as it is not needed anymore
            Destroy(itemText);
        }
```

6.26.2.11 Update()

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

Updates the slot

Definition at line 42 of file InventorySlot.cs.

```
        {
            CheckItem();
            UpdateIcon();
        }
```

6.26.2.12 UpdateIcon()

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

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

Definition at line 52 of file InventorySlot.cs.

```
        {
            if(item == null)
            {
                GetComponent<Image>().sprite = null;
            }
            else
            {
                if(!item.Equals(new Item()))
                    GetComponent<Image>().sprite = item.GetItemSprite();
            }

            //* if the slot is selected in the hotbar give the player some indication by colouring it grey
            if (selectedSlot)
            {
                GetComponent<Image>().color = Color.gray;
            }
            else
            {
                GetComponent<Image>().color = Color.white;
            }
        }
```

6.26.3 Member Data Documentation

6.26.3.1 item

```
Item BeeGame.Inventory.InventorySlot.item
```

The item this slot has in it

Definition at line 20 of file InventorySlot.cs.

6.26.3.2 itemsCanBeInserted

```
bool BeeGame.Inventory.InventorySlot.itemsCanBeInserted = true
```

Can items be inserted into this slot by the player

Definition at line 36 of file InventorySlot.cs.

6.26.3.3 itemText

```
GameObject BeeGame.Inventory.InventorySlot.itemText
```

If the slot currently has the item text object made this will be not null otherwise it is null

Definition at line 28 of file InventorySlot.cs.

6.26.3.4 myInventory

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

The Inventory this slot is in

Definition at line 24 of file InventorySlot.cs.

6.26.3.5 selectedSlot

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

Is this slot currently the selected slot in the hotbar?

Definition at line 32 of file InventorySlot.cs.

6.26.3.6 slotIndex

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

The slot in the inventory this is

Definition at line 16 of file InventorySlot.cs.

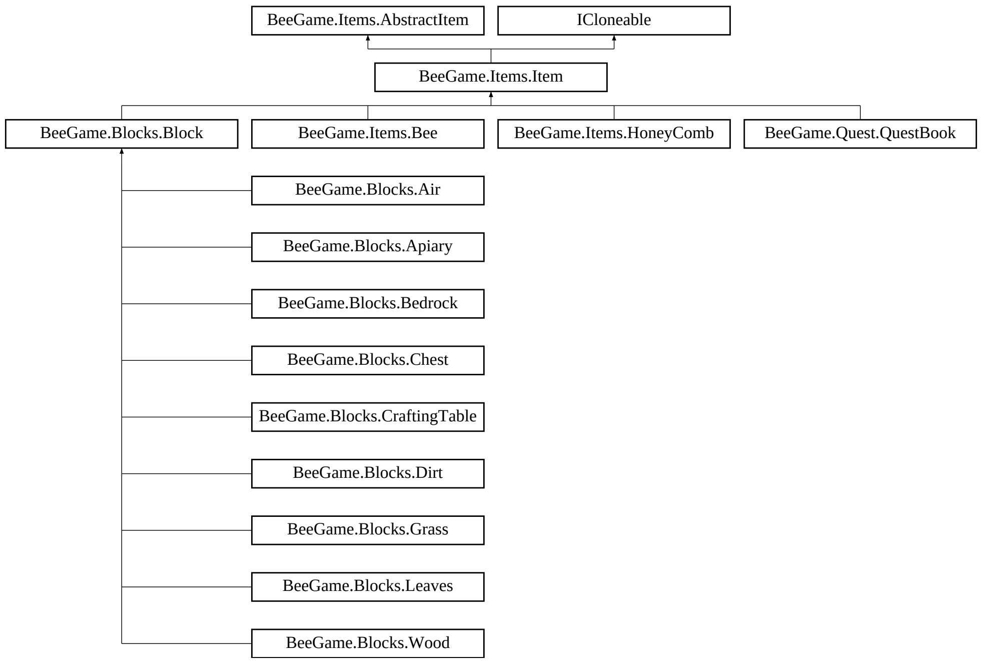
The documentation for this class was generated from the following file:

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

6.27 BeeGame.Items.Item Class Reference

Base class for all Items and Blocks in the game

Inheritance diagram for BeeGame.Items.Item:



Public Member Functions

- Item ()
- Item (string name)
- virtual bool InteractWithObject ()
- virtual GameObject GetGameObject ()

    Returns the GameObject for the item of it has one
- override string GetItemID ()

    Returns the id for the item as a string
- virtual Sprite GetItemSprite ()

    Returns the sprite for the item
- override string GetItemName ()

    Returns the items name
- virtual Tile TexturePosition (Direction direction)

    Texture postion of the items texture
- virtual MeshData ItemMesh (int x, int y, int z, MeshData meshData)

    Returns the mesh for the item
- virtual Vector2 [ ] FaceUVs (Direction direction)

    Sets the UVs for the given Direction
- object Clone ()

    Slow try no to use. Instead use Extensions.CloneObject< T >(T)
- override string ToString ()

    Returns the item name an id formatted nicely
- override int GetHashCode ()

    Returns the hashcode for the item
- override bool Equals (object obj)

    Checks if the item is equal to another

Static Public Member Functions

- static bool operator== (Item a, Item b)

    Overides the default == operator as different things need to be checked
- static bool operator!= (Item a, Item b)

    Inverse of ==

Public Attributes

- virtual bool placeable => false

    Is this item placeable. Saves checking if the item is a block type
- virtual int maxStackCount => 64

    Max number of items in a stack

Static Public Attributes

- static int ID => 0

Protected Member Functions

- virtual MeshData FaceDataUp (int x, int y, int z, MeshData meshData, bool addToRenderMesh=true, float blockSize=0.5f)

    Adds the Upwards face to the given MeshData
- virtual MeshData FaceDataDown (int x, int y, int z, MeshData meshData, bool addToRenderMesh=true, float blockSize=0.5f)

    Adds the Bottom face to the given MeshData
- virtual MeshData FaceDataNorth (int x, int y, int z, MeshData meshData, bool addToRenderMesh=true, float blockSize=0.5f)

    Adds the North face to the given MeshData
- virtual MeshData FaceDataEast (int x, int y, int z, MeshData meshData, bool addToRenderMesh=true, float blockSize=0.5f)

    Adds the East face to the given MeshData
- virtual MeshData FaceDataSouth (int x, int y, int z, MeshData meshData, bool addToRenderMesh=true, float blockSize=0.5f)

    Adds the South face to the given MeshData
- virtual MeshData FaceDataWest (int x, int y, int z, MeshData meshData, bool addToRenderMesh=true, float blockSize=0.5f)

    Adds the West face to the given MeshData

Properties

- string itemName `[get, set]`

    Name of the item
- bool usesGameObject `[get, set]`

    Does the item use a gameobject
- int itemStackCount `[get, set]`

    Number of items in the stack

Private Attributes

- const float tileSize = 0.1f

    How big are the texture tiles in the texture map (1/tile number x)
- int count = 1

6.27.1 Detailed Description

Base class for all Items and Blocks in the game

Definition at line 16 of file Item.cs.

6.27.2 Constructor & Destructor Documentation

6.27.2.1 Item() [1/2]

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

Definition at line 51 of file Item.cs.

```
        {
            itemName = "TestItem";
        }
```

6.27.2.2 Item() [2/2]

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

Definition at line 56 of file Item.cs.

```
        {
            itemName = name;
        }
```

6.27.3 Member Function Documentation

6.27.3.1 Clone()

`object BeeGame.Items.Item.Clone ( )`

Slow try no to use. Instead use Extensions.CloneObject<T>(T)

Returns

A deep copy of this

Definition at line 330 of file Item.cs.

```
        {
            //* Saves this to a file then reads it back so that a copy and not a reference is passed
            BinaryFormatter bf = new BinaryFormatter();
            MemoryStream ms = new MemoryStream();

            bf.Serialize(ms, this);
            ms.Seek(0, SeekOrigin.Begin);

            return bf.Deserialize(ms);
        }
```

6.27.3.2 Equals()

```
override bool BeeGame.Items.Item.Equals (
            object obj )
```

Checks if the item is equal to another

Parameters

obj	object to check against

Returns

true if items are the same

Definition at line 367 of file Item.cs.

```
        {
            if (!(obj is Item))
                return false;

            return this == (obj as Item);
        }
```

6.27.3.3 FaceDataDown()

```
virtual MeshData BeeGame.Items.Item.FaceDataDown (
            int x,
            int y,
            int z,
            MeshData meshData,
            bool addToRenderMesh = true,
            float blockSize = 0.5f ) [protected], [virtual]
```

Adds the Bottom face to the given MeshData

Parameters

x	X pos of the item
y	Y pos of the item
z	Z pos of the item
meshData	MeshData to add the face to
addToRenderMesh	Should the mesh be added to the render mesh (default true)
blockSize	how big is the item

Returns

Given MeshData with the face data added

Definition at line 194 of file Item.cs.

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

            //* adds teh tirs for the quad
            meshData.AddQuadTriangles(addToRenderMesh);

            //* if the data should be added to the render mesh also add the uvs to the mesh
            if (addToRenderMesh)
                meshData.uv.AddRange(FaceUVs(Direction.DOWN));

            return meshData;
        }
```

6.27.3.4 FaceDataEast()

```
virtual MeshData BeeGame.Items.Item.FaceDataEast (
            int x,
            int y,
            int z,
            MeshData meshData,
            bool addToRenderMesh = true,
            float blockSize = 0.5f ) [protected], [virtual]
```

Adds the East face to the given MeshData

Parameters

x	X pos of the item
y	Y pos of the item
z	Z pos of the item
meshData	MeshData to add the face to
addToRenderMesh	Should the mesh be added to the render mesh (default true)
blockSize	how big is the item

Returns

Given MeshData with the face data added

Definition at line 250 of file Item.cs.

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

            //* adds teh tirs for the quad
            meshData.AddQuadTriangles(addToRenderMesh);

            //* if the data should be added to the render mesh also add the uvs to the mesh
            if (addToRenderMesh)
                meshData.uv.AddRange(FaceUVs(Direction.EAST));

            return meshData;
        }
```

6.27.3.5 FaceDataNorth()

```
virtual MeshData BeeGame.Items.Item.FaceDataNorth (
                int x,
                int y,
                int z,
                MeshData meshData,
                bool addToRenderMesh = true,
                float blockSize = 0.5f )  [protected], [virtual]
```

Adds the North face to the given MeshData

Parameters

x	X pos of the item
y	Y pos of the item
z	Z pos of the item
meshData	MeshData to add the face to
addToRenderMesh	Should the mesh be added to the render mesh (default true)
blockSize	how big is the item

Returns

Given MeshData with the face data added

Definition at line 222 of file Item.cs.

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

            //* adds teh tirs for the quad
            meshData.AddQuadTriangles(addToRenderMesh);

            //* if the data should be added to the render mesh also add the uvs to the mesh
            if (addToRenderMesh)
                meshData.uv.AddRange(FaceUVs(Direction.NORTH));

            return meshData;
        }
```

6.27.3.6 FaceDataSouth()

```
virtual MeshData BeeGame.Items.Item.FaceDataSouth (
                int x,
                int y,
                int z,
                MeshData meshData,
                bool addToRenderMesh = true,
                float blockSize = 0.5f )  [protected], [virtual]
```

Adds the South face to the given MeshData

Parameters

x	X pos of the item
y	Y pos of the item
z	Z pos of the item
meshData	MeshData to add the face to
addToRenderMesh	Should the mesh be added to the render mesh (default true)
blockSize	how big is the item

Returns

Given MeshData with the face data added

Definition at line 278 of file Item.cs.

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

```
            meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z -
blockSize), addToRenderMesh);
            meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z -
blockSize), addToRenderMesh);
            meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z -
blockSize), addToRenderMesh);

            //* adds teh tirs for the quad
            meshData.AddQuadTriangles(addToRenderMesh);

            //* if the data should be added to the render mesh also add the uvs to the mesh
            if (addToRenderMesh)
                meshData.uv.AddRange(FaceUVs(Direction.SOUTH));

            return meshData;
        }
```

6.27.3.7 FaceDataUp()

```
virtual MeshData BeeGame.Items.Item.FaceDataUp (
            int x,
            int y,
            int z,
            MeshData meshData,
            bool addToRenderMesh = true,
            float blockSize = 0.5f ) [protected], [virtual]
```

Adds the Upwards face to the given MeshData

Parameters

x	X pos of the item
y	Y pos of the item
z	Z pos of the item
meshData	MeshData to add the face to
addToRenderMesh	Should the mesh be added to the render mesh (default true)
blockSize	how big is the item

Returns

Given MeshData with the face data added

Definition at line 166 of file Item.cs.

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

            //* adds teh tirs for the quad
            meshData.AddQuadTriangles(addToRenderMesh);

            //* if the data should be added to the render mesh also add the uvs to the mesh
            if (addToRenderMesh)
                meshData.uv.AddRange(FaceUVs(Direction.UP));

            return meshData;
        }
```

6.27.3.8 FaceDataWest()

```
virtual MeshData BeeGame.Items.Item.FaceDataWest (
              int x,
              int y,
              int z,
              MeshData meshData,
              bool addToRenderMesh = true,
              float blockSize = 0.5f )  [protected], [virtual]
```

Adds the West face to the given MeshData

Parameters

x	X pos of the item
y	Y pos of the item
z	Z pos of the item
meshData	MeshData to add the face to
addToRenderMesh	Should the mesh be added to the render mesh (default true)
blockSize	how big is the item

Returns

Given MeshData with the face data added

Definition at line 306 of file Item.cs.

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

            //* adds teh tirs for the quad
            meshData.AddQuadTriangles(addToRenderMesh);

            //* if the data should be added to the render mesh also add the uvs to the mesh
            if (addToRenderMesh)
                meshData.uv.AddRange(FaceUVs(Direction.WEST));

            return meshData;
        }
```

6.27.3.9 FaceUVs()

```
virtual Vector2 [] BeeGame.Items.Item.FaceUVs (
              Direction direction )  [virtual]
```

Sets the UVs for the given Direction

Parameters

direction	Direction to add the texture

Returns

Array of Vector2 to add to the UVsreturns>

Definition at line 141 of file Item.cs.

```
        {
            //* only 4 uvs per face
            Vector2[] UVs = new Vector2[4];
            Tile tilePos = TexturePosition(direction);

            //* sets the UVs for each vertex
            UVs[0] = new THVector2(tileSize * tilePos.x +
   tileSize - 0.01f, tileSize * tilePos.y + 0.01f);
            UVs[1] = new THVector2(tileSize * tilePos.x +
   tileSize - 0.01f, tileSize * tilePos.y + tileSize - 0.01f);
            UVs[2] = new THVector2(tileSize * tilePos.x + 0.01f,
   tileSize * tilePos.y + tileSize - 0.01f);
            UVs[3] = new THVector2(tileSize * tilePos.x + 0.01f,
   tileSize * tilePos.y + 0.01f);

            return UVs;
        }
```

6.27.3.10 GetGameObject()

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

Returns the GameObject for the item of it has one

Returns

GameObject for the item

Reimplemented in BeeGame.Blocks.CraftingTable, BeeGame.Items.HoneyComb, BeeGame.Blocks.Chest, and BeeGame.Blocks.Apiary.

Definition at line 74 of file Item.cs.

```
{ return null; }
```

6.27.3.11 GetHashCode()

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

Returns the hashcode for the item

Returns

1

Implements BeeGame.Items.AbstractItem.

Reimplemented in BeeGame.Quest.QuestBook.

Definition at line 357 of file Item.cs.

```
        {
            return ID;
        }
```

6.27.3.12 GetItemID()

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

Returns the id for the item as a string

Returns

Implements BeeGame.Items.AbstractItem.

Definition at line 80 of file Item.cs.

```
        {
            return $"{GetHashCode()}";
        }
```

6.27.3.13 GetItemName()

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

Returns the items name

Returns

Implements BeeGame.Items.AbstractItem.

Definition at line 98 of file Item.cs.

```
        {
            return $"{itemName}";
        }
```

6.27.3.14 GetItemSprite()

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

Returns the sprite for the item

Returns

Sprite for this item

Reimplemented in BeeGame.Blocks.CraftingTable, BeeGame.Items.Bee, BeeGame.Items.HoneyComb, Bee↩
Game.Blocks.Block, BeeGame.Blocks.Grass, BeeGame.Blocks.Dirt, BeeGame.Quest.QuestBook, BeeGame.↩
Blocks.Wood, and BeeGame.Blocks.Leaves.

Definition at line 89 of file Item.cs.

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

6.27.3.15 InteractWithObject()

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

Reimplemented in BeeGame.Quest.QuestBook.

Definition at line 63 of file Item.cs.

```
        {
            return false;
        }
```

6.27.3.16 ItemMesh()

```
virtual MeshData BeeGame.Items.Item.ItemMesh (
            int x,
            int y,
            int z,
            MeshData meshData )  [virtual]
```

Returns the mesh for the item

Parameters

x	X pos if the item
y	Y pos if the item
z	Z pos if the item
meshData	data to add the mesh to

Returns

given MeshData with the items mesh added

Definition at line 123 of file Item.cs.

```
        {
            //* adds all faces of the item to the mesh as all faces could be seen at any time
            meshData = FaceDataUp(x, y, z, meshData, true, 0.25f);
            meshData = FaceDataDown(x, y, z, meshData, true, 0.25f);
            meshData = FaceDataNorth(x, y, z, meshData, true, 0.25f);
            meshData = FaceDataEast(x, y, z, meshData, true, 0.25f);
            meshData = FaceDataSouth(x, y, z, meshData, true, 0.25f);
            meshData = FaceDataWest(x, y, z, meshData, true, 0.25f);

            return meshData;
        }
```

6.27.3.17 operator"!=()

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

Inverse of ==

Parameters

a	Item
b	Item

Returns

True if *a* != *b*

Definition at line 400 of file Item.cs.

```
        {
            return !(a == b);
        }
```

6.27.3.18 operator==()

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

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

Parameters

a	Item
b	Item

Returns

true if *a* == *b*

Definition at line 381 of file Item.cs.

```
        {
            if (ReferenceEquals(a, null) && ReferenceEquals(b, null))
                return true;
            if (ReferenceEquals(a, null) || ReferenceEquals(b, null))
                return false;

            if(a.GetItemID() == b.GetItemID())
                return true;

            return false;
        }
```

6.27.3.19 TexturePosition()

```
virtual Tile BeeGame.Items.Item.TexturePosition (
            Direction direction )  [virtual]
```

Texture postion of the items texture

Parameters

direction	Direction for the texture

Returns

Position of the texture

Reimplemented in BeeGame.Blocks.CraftingTable, BeeGame.Blocks.Chest, BeeGame.Blocks.Apiary, Bee↩ Game.Blocks.Grass, BeeGame.Blocks.Bedrock, BeeGame.Blocks.Dirt, BeeGame.Blocks.Wood, and BeeGame.↩ Blocks.Leaves.

Definition at line 110 of file Item.cs.

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

6.27.3.20 ToString()

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

Returns the item name an id formatted nicely

Returns

Definition at line 348 of file Item.cs.

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

6.27.4 Member Data Documentation

6.27.4.1 count

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

Definition at line 40 of file Item.cs.

6.27.4.2 ID

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

Definition at line 47 of file Item.cs.

6.27.4.3 maxStackCount

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

Max number of items in a stack

Definition at line 45 of file Item.cs.

6.27.4.4 placeable

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

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

Definition at line 26 of file Item.cs.

6.27.4.5 tileSize

```
const float BeeGame.Items.Item.tileSize = 0.1f  [private]
```

How big are the texture tiles in the texture map (1/tile number x)

Definition at line 34 of file Item.cs.

6.27.5 Property Documentation

6.27.5.1 itemName

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

Name of the item

Definition at line 22 of file Item.cs.

6.27.5.2 itemStackCount

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

Number of items in the stack

Definition at line 39 of file Item.cs.

6.27.5.3 usesGameObject

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

Does the item use a gameobject

Definition at line 30 of file Item.cs.

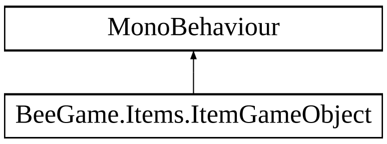
The documentation for this class was generated from the following file:

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

6.28 BeeGame.Items.ItemGameObject Class Reference

Interface between item and inity gameobjects

Inheritance diagram for BeeGame.Items.ItemGameObject:



Public Attributes

- Item item

    Item that this gameobject repersents
- GameObject go

    GameObject to make

Private Member Functions

- void Start ()

    Makes the mesh or instantiates the items gameobject
- void Update ()

    Destroys the game object if it falls to low
- void MakeMesh ()

    Makes the items mesh

6.28.1 Detailed Description

Interface between item and inity gameobjects

Definition at line 18 of file ItemGameObject.cs.

6.28.2 Member Function Documentation

6.28.2.1 MakeMesh()

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

Makes the items mesh

Definition at line 58 of file ItemGameObject.cs.

```
        {
            MeshData meshData = new MeshData();
            if(item != null)
                meshData = item.ItemMesh(0, 0, 0, meshData);

            Mesh mesh = new Mesh()
            {
                vertices = meshData.verts.ToArray(),
                triangles = meshData.tris.ToArray(),
                uv = meshData.uv.ToArray()
            };

            mesh.RecalculateNormals();

            GetComponent<MeshFilter>().mesh = mesh;
        }
```

6.28.2.2 Start()

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

Makes the mesh or instantiates the items gameobject

Definition at line 32 of file ItemGameObject.cs.

```
        {
            if (!item.usesGameObject)
                MakeMesh();

            if (item.usesGameObject)
            {
                Instantiate(item.GetGameObject(), transform, false);
                transform.localScale = new Vector3(0.5f, 0.5f, 0.5f);
            }
        }
```

6.28.2.3 Update()

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

Destroys the game object if it falls to low

Definition at line 47 of file ItemGameObject.cs.

```
        {
            if(transform.position.y < -100)
             {
                 Destroy(gameObject);
             }
        }
```

6.28.3 Member Data Documentation

6.28.3.1 go

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

GameObject to make

Definition at line 27 of file ItemGameObject.cs.

6.28.3.2 item

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

Item that this gameobject repersents

Definition at line 23 of file ItemGameObject.cs.

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

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

6.29 BeeGame.Inventory.ItemsInInventory Class Reference

Class that holds all of the items in the inventory. Can be serialized so inventory may be saved

Public Member Functions

- ItemsInInventory (int numberOfInventorySlots)

  Sets the size of the inventory
- void AddItem (int index, Item item)

  Add an Item to a specific index in the inventory
- bool AddItem (Item item)

  Adds a Item to the inventory

Public Attributes

- Item [ ] itemsInInventory
  All of the items in the inventory

6.29.1 Detailed Description

Class that holds all of the items in the inventory. Can be serialized so inventory may be saved

Definition at line 10 of file ItemsInInventory.cs.

6.29.2 Constructor & Destructor Documentation

6.29.2.1 ItemsInInventory()

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

Sets the size of the inventory

Parameters

numberOfInventorySlots	

Definition at line 21 of file ItemsInInventory.cs.

```
        {
            itemsInInventory = new Item[numberOfInventorySlots];
        }
```

6.29.3 Member Function Documentation

6.29.3.1 AddItem() [1/2]

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

Add an Item to a specific index in the inventory

Parameters

index	Were to add the item
item	What Item to put in the inventory

Definition at line 31 of file ItemsInInventory.cs.

```
        {
            itemsInInventory[index] = item;
        }
```

6.29.3.2 AddItem() [2/2]

```
bool BeeGame.Inventory.ItemsInInventory.AddItem (
            Item item )
```

Adds a Item to the inventory

Parameters

item	Item to add

Returns

true if *item* was added to the inventory

Definition at line 41 of file ItemsInInventory.cs.

```
        {
            for (int i = 0; i < itemsInInventory.Length; i++)
            {
                if (itemsInInventory[i] == null)
                {
                    itemsInInventory[i] = item;
                    return true;
                }
                if (itemsInInventory[i] == item &&
      itemsInInventory[i].itemStackCount + 1 <= itemsInInventory[i].maxStackCount
      )
                {
                    itemsInInventory[i].itemStackCount++;
                    return true;
                }
            }

            return false;
        }
```

6.29.4 Member Data Documentation

6.29.4.1 itemsInInventory

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

All of the items in the inventory

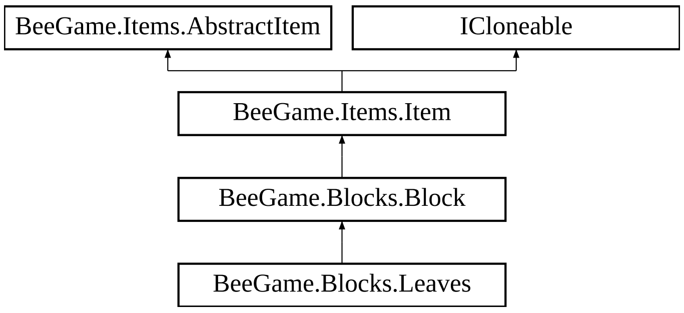
Definition at line 15 of file ItemsInInventory.cs.

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

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

6.30 BeeGame.Blocks.Leaves Class Reference

Inheritance diagram for BeeGame.Blocks.Leaves:



Public Member Functions

- Leaves ()
- override Sprite GetItemSprite ()

  Returns the sprite for the item
- override Tile TexturePosition (Direction direction)

  Texture postion of the items texture
- override bool IsSolid (Direction direction)

  What Directions is this Block solid in
- override int GetHashCode ()

  Base ID of the block
- override string ToString ()

  Returns the name and ID of the block as a string

Static Public Attributes

- static new int ID => 6

Additional Inherited Members

6.30.1 Detailed Description

Definition at line 10 of file Leaves.cs.

6.30.2 Constructor & Destructor Documentation

6.30.2.1 Leaves()

```
BeeGame.Blocks.Leaves.Leaves ( )
```

Definition at line 14 of file Leaves.cs.

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

6.30.3 Member Function Documentation

6.30.3.1 GetHashCode()

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

Base ID of the block

Returns

5

Reimplemented from BeeGame.Blocks.Block.

Definition at line 41 of file Leaves.cs.

```
        {
            return ID;
        }
```

6.30.3.2 GetItemSprite()

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

Returns the sprite for the item

Returns

Sprite for this item

Reimplemented from BeeGame.Blocks.Block.

Definition at line 20 of file Leaves.cs.

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

6.30.3.3 IsSolid()

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

What Directions is this Block solid in

Parameters

direction	Direction to check

Returns

Default returns true for all sides

Reimplemented from BeeGame.Blocks.Block.

Definition at line 31 of file Leaves.cs.

```
        {
            return false;
        }
```

6.30.3.4 TexturePosition()

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

Texture postion of the items texture

Parameters

direction	Direction for the texture

Returns

Position of the texture

Reimplemented from BeeGame.Items.Item.

Definition at line 26 of file Leaves.cs.

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

6.30.3.5 ToString()

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

Returns the name and ID of the block as a string

Returns

A nicely formatted string

Definition at line 50 of file Leaves.cs.

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

6.30.4 Member Data Documentation

6.30.4.1 ID

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

Definition at line 12 of file Leaves.cs.

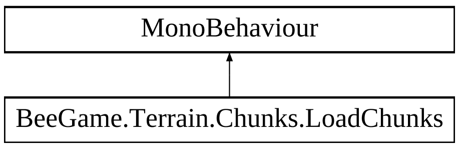
The documentation for this class was generated from the following file:

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

6.31 BeeGame.Terrain.Chunks.LoadChunks Class Reference

Loads the Chunks around the player

Inheritance diagram for BeeGame.Terrain.Chunks.LoadChunks:



Public Attributes

- World world

  The world the player is in

Private Member Functions

- void Start ()

  Sets the world
- void Update ()

  Builds, Renders, and Remmoves Chunks
- void ApplyCollsionMeshToNearbyChunks ()

  Makes a collsion mesh for the Chunks nearest to the player to reduce lag created by PhysX mesh bakeing
- void LoadAndRenderChunks ()

  Gets the chunks that sould 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 aroud the player ///
- static ChunkWorldPos [ ] nearbyChunks

  Chunks in a 3x3 radius around the player that should have a collision mesh
- static int timer = 0

  Timer for chunk removal

6.31.1 Detailed Description

Loads the Chunks around the player

Definition at line 11 of file LoadChunks.cs.

6.31.2 Member Function Documentation

6.31.2.1 ApplyCollsionMeshToNearbyChunks()

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

Makes a collsion mesh for the Chunks nearest to the player to reduce lag created by PhysX mesh bakeing

We dont need to worry about removeing Chunk collision meshes as once PhysX has baked then they have minimal performance impact Doing things this wayt also spreads out the PhysX mesh bakeing

Definition at line 111 of file LoadChunks.cs.

```
        {
            //* gets the player position in chunk coordinates
            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);

            for (int i = 0; i < nearbyChunks.Length; i++)
            {
                ChunkWorldPos chunkPos = new ChunkWorldPos(nearbyChunks[i].x * Chunk.chunkSize
     + playerPos.x, 0, nearbyChunks[i].z * Chunk.chunkSize + playerPos.z);

                for (int j = -1; j < 2; j++)
                {
                    Chunk nearbyChunk = world.GetChunk(chunkPos.x, j * Chunk.chunkSize,
     chunkPos.z);

                    if (nearbyChunk != null)
                        nearbyChunk.applyCollisionMesh = true;
                }
            }
        }
```

6.31.2.2 BuildChunk()

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

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

Parameters

pos	hte positon of the new chunk

Definition at line 186 of file LoadChunks.cs.

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

6.31.2.3 DeleteChunks()

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

Destroys Chunks every 10 calls

Returns

true if Chunks were destroyed

Definition at line 196 of file LoadChunks.cs.

```
        {
            //* destroys every 10 call to reduce load on CPU so that chunks are not destroyed and created
      at the same time
            if(timer == 10)
            {
                timer = 0;
                var chunksToDelete = new List<ChunkWorldPos>();

                // *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
                foreach (var chunk in world.chunks)
                {
                    float distance = Vector3.Distance(chunk.Value.transform.position, transform.position);

                    if (distance > 256)
                        chunksToDelete.Add(chunk.Key);
                }

                foreach (var chunk in chunksToDelete)
                {
                    world.DestroyChunk(chunk.x, chunk.y, chunk.z);
                }

                return true;
            }

            timer++;

            return false;
        }
```

6.31.2.4 FindChunksToLoad()

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

Finds the Chunks that should be rendered

Definition at line 150 of file LoadChunks.cs.

```
        {
            if (buildList.Count == 0)
            {
                //* gets the player position in chunk coordinates
                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);

                //* check all of the chunk positions and if that position does not have a chunk in it make it
                for (int i = 0; i < chunkPositions.Length; i++)
                {
                    ChunkWorldPos newChunkPos = new ChunkWorldPos(chunkPositions[i].x * Chunk.chunkSize + playerPos.x, 0, chunkPositions[i].z * Chunk.chunkSize + playerPos.z);

                    Chunk newChunk = world.GetChunk(newChunkPos.x, newChunkPos.y, newChunkPos.z);

                    if (newChunk != null && (newChunk.rendered || buildList.Contains(newChunkPos)))
                        continue;

                    for (int y = -1; y < 2; y++)
                    {
                        for (int x = newChunkPos.x - Chunk.chunkSize; x < newChunkPos.x + Chunk.chunkSize; x += Chunk.chunkSize)
                        {
                            for (int z = newChunkPos.z - Chunk.chunkSize; z < newChunkPos.z + Chunk.chunkSize; z += Chunk.chunkSize)
                            {
                                buildList.Add(new ChunkWorldPos(x, y * Chunk.chunkSize, z));
                            }
                        }
                    }
                    return;
                }
            }
        }
```

6.31.2.5 LoadAndRenderChunks()

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

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

Definition at line 133 of file LoadChunks.cs.

```
        {
            //* if their is somethign in the build list new chunks can be made
            if (buildList.Count != 0)
            {
                //* 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
                for (int i = buildList.Count - 1, j = 0; i >= 0 && j < 8; i--, j++)
                {
                    BuildChunk(buildList[0]);
                    buildList.RemoveAt(0);
                }
            }
        }
```

6.31.2.6 Start()

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

Sets the world

Definition at line 82 of file LoadChunks.cs.

```
        {
            LandGeneration.Terrain.world = world;
        }
```

6.31.2.7 Update()

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

Builds, Renders, and Remmoves Chunks

Definition at line 90 of file LoadChunks.cs.

```
        {
            if (DeleteChunks())
                return;
            if (!world.chunkHasMadeCollisionMesh)
            {
                FindChunksToLoad();
                LoadAndRenderChunks();
                ApplyCollsionMeshToNearbyChunks();
            }
            //* stops chunks being made and collision meshes being made at the same time
            world.chunkHasMadeCollisionMesh = false;
        }
```

6.31.3 Member Data Documentation

6.31.3.1 buildList

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

List if chunks to build

Definition at line 22 of file LoadChunks.cs.

6.31.3.2 chunkPositions

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

Positions to make chunks aroud the player ///

Definition at line 27 of file LoadChunks.cs.

6.31.3.3 nearbyChunks

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

Initial value:

```
= new ChunkWorldPos[] { new ChunkWorldPos(0, 0, 0), new ChunkWorldPos(1, 0, 0), new ChunkWorldPos(-1, 0, 0)
      , new ChunkWorldPos(0, 0, 1), new ChunkWorldPos(0, 0, -1),
                                                                             new ChunkWorldPos(1, 0, 1), new
       ChunkWorldPos(1, 0, -1), new ChunkWorldPos(-1, 0, 1), new ChunkWorldPos(-1, 0, -1)}
```

Chunks in a 3x3 radius around the player that should have a collision mesh

Definition at line 70 of file LoadChunks.cs.

6.31.3.4 timer

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

Timer for chunk removal

Definition at line 76 of file LoadChunks.cs.

6.31.3.5 world

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

The world the player is in

Definition at line 17 of file LoadChunks.cs.

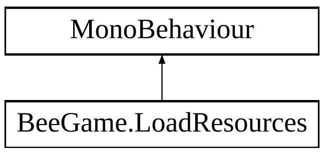
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/Load↩
  Chunks.cs

6.32 BeeGame.LoadResources Class Reference

Loads all of the resources in the game

Inheritance diagram for BeeGame.LoadResources:



Private Member Functions

- void Awake ()

  Loads the sprites and prefab dictionarys

6.32.1 Detailed Description

Loads all of the resources in the game

Definition at line 9 of file LoadResources.cs.

6.32.2 Member Function Documentation

6.32.2.1 Awake()

```
void BeeGame.LoadResources.Awake ( )  [private]
```

Loads the sprites and prefab dictionarys

Definition at line 14 of file LoadResources.cs.

```
        {
            Serialization.Serialization.MakeDirectorys();

            Serialization.Serialization.LoadPlayerPosition(GameObject.Find("Player").GetComponent<Transform
   >());

            SpriteDictionary.LoadSprites();
            PrefabDictionary.LoadPrefabs();
        }
```

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

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/LoadResources.cs

6.33 BeeGame.Terrain.Chunks.MeshData Class Reference

The data for a Chunks's Mesh

Public Member Functions

- void AddQuadTriangles (bool addToRenderMesh=true)

  Adds 2 triangles to the triangle list
- void AddVertices (THVector3 pos, bool addToRenderMesh=true, Direction direction=Direction.DOWN)

  Adds vertices to the render and collision Meshes
- void AddTriangle (int tri)

  Adds a triangle to both the render and collidson meshes

Public Attributes

- List< Vector3 > verts = new List<Vector3>()

  Verticies for the Chunk render Mesh
- List< int > tris = new List<int>()

  Triangles for the Chunk render Mesh
- List< Vector2 > uv = new List<Vector2>()

  UV mapping for the Chunk render Mesh
- List< Vector3 > colVerts = new List<Vector3>()

  Vertices for the Chunk collider Mesh
- List< int > colTris = new List<int>()

  Triangles for the Chunk collider Mesh
- bool shareMeshes = true

  Should this chunk share is collider and render Meshes
- bool done = false

6.33.1 Detailed Description

The data for a Chunks's Mesh

Definition at line 11 of file MeshData.cs.

6.33.2 Member Function Documentation

6.33.2.1 AddQuadTriangles()

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

Adds 2 triangles to the triangle list

Parameters

addToRenderMesh	Should the triangles be added to the render Mesh

Definition at line 46 of file MeshData.cs.

```
        {
            //*adds the triangles in an anticlockwise order

            if (addToRenderMesh)
            {
                tris.Add(verts.Count - 4);
                tris.Add(verts.Count - 3);
                tris.Add(verts.Count - 2);
                tris.Add(verts.Count - 4);
                tris.Add(verts.Count - 2);
                tris.Add(verts.Count - 1);
            }

            colTris.Add(colVerts.Count - 4);
            colTris.Add(colVerts.Count - 3);
            colTris.Add(colVerts.Count - 2);
            colTris.Add(colVerts.Count - 4);
            colTris.Add(colVerts.Count - 2);
            colTris.Add(colVerts.Count - 1);
        }
```

6.33.2.2 AddTriangle()

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

Adds a triangle to both the render and collidson meshes

Parameters

tri	triangle

not used anymore remove?

Definition at line 91 of file MeshData.cs.

```
        {
            tris.Add(tri);

            colTris.Add(tri - (verts.Count - colVerts.Count));
        }
```

6.33.2.3 AddVertices()

```
void BeeGame.Terrain.Chunks.MeshData.AddVertices (
            THVector3 pos,
            bool addToRenderMesh = true,
            Direction direction = Direction.DOWN )
```

Adds vertices to the render and collision Meshes

Parameters

pos	Position of the vertice
addToRenderMesh	Should the vertice be added to the render Mesh
direction	What face is this vertice on

Definition at line 74 of file MeshData.cs.

```
        {
            if (addToRenderMesh)
                verts.Add(pos);

            //* if the vertice is on the top face make its positon slightly smaller
            if(direction == Direction.UP)
                colVerts.Add(pos - new THVector3(0.01f, 0, 0.01f));
        }
```

6.33.3 Member Data Documentation

6.33.3.1 colTris

`List<int> BeeGame.Terrain.Chunks.MeshData.colTris = new List<int>()`

Triangles for the Chunk collider Mesh

Definition at line 33 of file MeshData.cs.

6.33.3.2 colVerts

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

Vertices for the Chunk collider Mesh

Definition at line 29 of file MeshData.cs.

6.33.3.3 done

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

Definition at line 40 of file MeshData.cs.

6.33.3.4 shareMeshes

`bool BeeGame.Terrain.Chunks.MeshData.shareMeshes = true`

Should this chunk share is collider and render Meshes

Definition at line 38 of file MeshData.cs.

6.33.3.5 tris

`List<int> BeeGame.Terrain.Chunks.MeshData.tris = new List<int>()`

Triangles for the Chunk render Mesh

Definition at line 20 of file MeshData.cs.

6.33.3.6 uv

```
List<Vector2> BeeGame.Terrain.Chunks.MeshData.uv = new List<Vector2>()
```

UV mapping for the Chunk render Mesh

Definition at line 24 of file MeshData.cs.

6.33.3.7 verts

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

Verticies for the Chunk render Mesh

Definition at line 16 of file MeshData.cs.

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

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/MeshData.cs

6.34 BeeGame.Items.NormalBee Class Reference

Public Member Functions

- override int GetHashCode ()

Public Attributes

- BeeSpecies pSpecies

  Primary BeeSpecies of the Bee
- BeeLifeSpan pLifespan

  Primary BeeLifeSpan of the Bee
- uint pFertility

  Primary Fertility of the Bee
- BeeEffect pEffect

  Primary BeeEffect of the Bee
- BeeProductionSpeed pProdSpeed

  Primary BeeProductionSpeed of the Bee
- BeeSpecies sSpecies

  Secondary BeeGame.Enums.BeeSpecies of the Bee
- BeeLifeSpan sLifespan

  Secondary BeeGame.Enums.BeeLifeSpan of the Bee
- uint sFertility

  Secondary Fertility of the Bee
- BeeEffect sEffect

  Secondary BeeGame.Enums.BeeEffect of the Bee
- BeeProductionSpeed sProdSpeed

  Secondary BeeGame.Enums.BeeProductionSpeed of the Bee

6.34.1 Detailed Description

Definition at line 240 of file Bee.cs.

6.34.2 Member Function Documentation

6.34.2.1 GetHashCode()

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

Definition at line 292 of file Bee.cs.

```
        {
            unchecked
            {
                //int hashcode = 13;

                var temp = $"
{(int)pSpecies}{(int)sSpecies}{(int)pLifespan}{(int)sLifespan}{(int)pFertility}{(int)sFertility}{(int)pEffect}{(int)sEf

                var hashcode = (int)(Int64.Parse(temp) ^ (127 * 13) / 159);

                //hashcode += ((int)pSpecies ^ (int)pLifespan ^ (int)pFertility ^ (int)pEffect ^
 (int)pProdSpeed) * 127;
                //hashcode += ((int)sSpecies ^ (int)sLifespan ^ (int)sFertility ^ (int)sEffect ^
 (int)sProdSpeed) * 307;

                return hashcode;
            }
        }
```

6.34.3 Member Data Documentation

6.34.3.1 pEffect

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

Primary BeeEffect of the Bee

Definition at line 260 of file Bee.cs.

6.34.3.2 pFertility

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

Primary Fertility of the Bee

Definition at line 256 of file Bee.cs.

6.34.3.3 pLifespan

`BeeLifeSpan BeeGame.Items.NormalBee.pLifespan`

Primary BeeLifeSpan of the Bee

Definition at line 252 of file Bee.cs.

6.34.3.4 pProdSpeed

`BeeProductionSpeed BeeGame.Items.NormalBee.pProdSpeed`

Primary BeeProductionSpeed of the Bee

Definition at line 264 of file Bee.cs.

6.34.3.5 pSpecies

`BeeSpecies BeeGame.Items.NormalBee.pSpecies`

Primary BeeSpecies of the Bee

Definition at line 248 of file Bee.cs.

6.34.3.6 sEffect

`BeeEffect BeeGame.Items.NormalBee.sEffect`

Secondary BeeGame.Enums.BeeEffect of the Bee

Definition at line 285 of file Bee.cs.

6.34.3.7 sFertility

`uint BeeGame.Items.NormalBee.sFertility`

Secondary Fertility of the Bee

Definition at line 281 of file Bee.cs.

6.34.3.8 sLifespan

`BeeLifeSpan BeeGame.Items.NormalBee.sLifespan`

Secondary BeeGame.Enums.BeeLifeSpan of the Bee

Definition at line 277 of file Bee.cs.

6.34.3.9 sProdSpeed

`BeeProductionSpeed BeeGame.Items.NormalBee.sProdSpeed`

Secondary BeeGame.Enums.BeeProductionSpeed of the Bee

Definition at line 289 of file Bee.cs.

6.34.3.10 sSpecies

`BeeSpecies BeeGame.Items.NormalBee.sSpecies`

Secondary BeeGame.Enums.BeeSpecies of the Bee

Definition at line 273 of file Bee.cs.

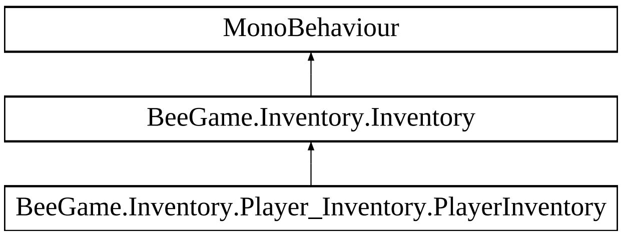
The documentation for this class was generated from the following file:

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

6.35 BeeGame.Inventory.Player_Inventory.PlayerInventory Class Reference

Controlls the player inventory

Inheritance diagram for BeeGame.Inventory.Player_Inventory.PlayerInventory:


MonoBehaviour
BeeGame.Inventory.Inventory
BeeGame.Inventory.Player_Inventory.PlayerInventory

Public Member Functions

- void SelectedSlot (int index)

  Updates the currrently selected hotbar slot
- bool GetItemFromHotBar (int slotIndex, out Item outItem)

  Gets an item from the hotbar (9 InventorySlots at the bottom of the screen)
- void RemoveItemFromInventory (int index)

  Removes 1 item from the given inventory index

Public Attributes

- GameObject playerInventory

  Object that the inventory is

Private Member Functions

- void Awake ()

  Sets all requred params for the inventory and loads ant saved versions of it
- void SetPlayerInventory ()

  Set the size of the player inventory
- void Update ()

  Goves the inventory update ticks
- void OpenPlayerInventory ()

  Show/Hide the player inventory
- void PickupItem (ItemGameObject item)

  Pickup an item and put it into the Inventory

Additional Inherited Members

6.35.1 Detailed Description

Controlls the player inventory

Definition at line 10 of file PlayerInventory.cs.

6.35.2 Member Function Documentation

6.35.2.1 Awake()

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

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

Definition at line 23 of file PlayerInventory.cs.

```
        {
            SetPlayerInventory();
            inventoryName = "PlayerInventory";
            Serialization.Serialization.DeSerializeInventory(this, inventoryName);
        }
```

6.35.2.2 GetItemFromHotBar()

```
bool BeeGame.Inventory.Player_Inventory.PlayerInventory.GetItemFromHotBar (
            int slotIndex,
            out Item outItem )
```

Gets an item from the hotbar (9 InventorySlots at the bottom of the screen)

Parameters

slotIndex	Index to get Item from
outItem	Item in the slot

Returns

true if *outItem* is placeable, false if *outItem* is null or not placeable

Definition at line 97 of file PlayerInventory.cs.

```
        {
            //* get the item
            outItem = GetAllItems().itemsInInventory[slotIndex];

            if (outItem == null)
                return false;

            //* if the item is placebale and is not null remove 1 from the inventory as it is assumed it is
about to be placed in the world
            if(outItem.placeable)
                RemoveItemFromInventory(slotIndex);

            return outItem.placeable;
        }
```

6.35.2.3 OpenPlayerInventory()

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

Show/Hide the player inventory

Definition at line 117 of file PlayerInventory.cs.

```
        {
            if (floatingItem != null)
                return;
            thisInventoryOpen = !thisInventoryOpen;
            playerInventory.SetActive(!playerInventory.activeInHierarchy);
            THInput.isAnotherInventoryOpen = !
THInput.isAnotherInventoryOpen;

            //* hides/shows the mouse depending on if te inventory is open or not
            if (playerInventory.activeInHierarchy)
            {
                Cursor.lockState = CursorLockMode.None;
                Cursor.visible = true;
            }
            else
            {
                Cursor.visible = false;
                Cursor.lockState = CursorLockMode.Locked;
            }
        }
```

6.35.2.4 PickupItem()

```
void BeeGame.Inventory.Player_Inventory.PlayerInventory.PickupItem (
            ItemGameObject item ) [private]
```

Pickup an item and put it into the Inventory

Parameters

item	Item to try to put into the inventory

Definition at line 161 of file PlayerInventory.cs.

```
        {
            item.item.itemStackCount = 1;

            //* if the item can be added to the inventory do that
            if (AddItemToInventory(item.item))
            {
                //* if the item was added destroyits gameobject and save the inventory
                Destroy(item.gameObject);
                Serialization.Serialization.SerializeInventory(this, inventoryName);
            }
        }
```

6.35.2.5 RemoveItemFromInventory()

```
void BeeGame.Inventory.Player_Inventory.PlayerInventory.RemoveItemFromInventory (
            int index )
```

Removes 1 item from the given inventory index

Parameters

index	

Definition at line 142 of file PlayerInventory.cs.

```
        {
            //* if the item is already null nothign needs to be removed
            if (GetAllItems().itemsInInventory[index] != null)
            {
                //* remove 1 item and if that was the last in the stack remove the item from the inventory
                GetAllItems().itemsInInventory[index].itemStackCount -= 1;

                if (GetAllItems().itemsInInventory[index].itemStackCount <= 0)
                    GetAllItems().itemsInInventory[index] = null;

                Serialization.Serialization.SerializeInventory(this, inventoryName);
            }
        }
```

6.35.2.6 SelectedSlot()

```
void BeeGame.Inventory.Player_Inventory.PlayerInventory.SelectedSlot (
            int index )
```

Updates the currrently selected hotbar slot

Parameters

index	Slot that is selected

Definition at line 81 of file PlayerInventory.cs.

```
        {
            for (int i = 0; i < slots.Length; i++)
            {
                slots[i].selectedSlot = false;
            }

            slots[index].selectedSlot = true;
        }
```

6.35.2.7 SetPlayerInventory()

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

Set the size of the player inventory

Definition at line 33 of file PlayerInventory.cs.

```
        {
            if (!InventorySet())
                SetInventorySize(36);
        }
```

6.35.2.8 Update()

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

Goves the inventory update ticks

Definition at line 43 of file PlayerInventory.cs.

```
        {
            UpdateBase();

            //* checks if the inventory should be opened/closed
            if ((thisInventoryOpen || !playerInventory.activeInHierarchy)
      && !THInput.chestOpen && THInput.GetButtonDown("Player Inventory"))
            {
                if (THInput.blockInventoryJustClosed)
                {
                    THInput.blockInventoryJustClosed = false;
                    return;
                }
                else
                {
                    OpenPlayerInventory();
                }
            }

            //* dont pickup items if the inventory is open
            if (THInput.isAnotherInventoryOpen)
                return;

            //* checks if somethig should be picked up and put into the inventory
            RaycastHit[] hit = Physics.SphereCastAll(transform.position, 1f, transform.forward);

            for (int i = hit.Length - 1; i >= 0; i--)
            {
                if (hit[i].collider.GetComponent<ItemGameObject>())
                    PickupItem(hit[i].collider.GetComponent<
      ItemGameObject>());
            }

        }
```

6.35.3 Member Data Documentation

6.35.3.1 playerInventory

```
GameObject BeeGame.Inventory.Player_Inventory.PlayerInventory.playerInventory
```

Object that the inventory is

Definition at line 16 of file PlayerInventory.cs.

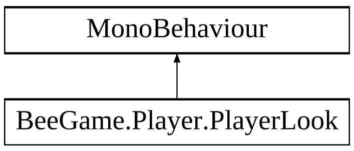
The documentation for this class was generated from the following file:

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

6.36 BeeGame.Player.PlayerLook Class Reference

The look for the player

Inheritance diagram for BeeGame.Player.PlayerLook:



Public Attributes

- Transform myTransform

    Player transfrom
- Transform cameraTransform

    Camera transfom
- float rotationLock

    Lock for camera X rotation
- float speed = 5

    Look move speed

Private Member Functions

- void Start ()

    Locks teh cursor and hides it
- void Update ()

    Every fixed update check if the look shoud be moved
- void Look ()

    Moves the look rotation

Private Attributes

- float yRot = 0

  Current Y rotation
- float xRot = 0

  Current X rotation

6.36.1 Detailed Description

The look for the player

Definition at line 9 of file PlayerLook.cs.

6.36.2 Member Function Documentation

6.36.2.1 Look()

```
void BeeGame.Player.PlayerLook.Look ( )  [private]
```

Moves the look rotation

Definition at line 66 of file PlayerLook.cs.

```
        {
            //Only X/Y rotation needed as Z rotation would be wierd
            yRot += Input.GetAxis("Mouse X") * speed * Time.timeScale;
            xRot -= Input.GetAxis("Mouse Y") * speed * Time.timeScale;

            //clamps the X rotation so the player camera cannot do flips
            xRot = Mathf.Clamp(xRot, -rotationLock,
      rotationLock);

            myTransform.rotation = Quaternion.Euler(0, yRot, 0);
            cameraTransform.localRotation = Quaternion.Euler(xRot, 0, 0);
        }
```

6.36.2.2 Start()

```
void BeeGame.Player.PlayerLook.Start ( )  [private]
```

Locks teh cursor and hides it

Definition at line 43 of file PlayerLook.cs.

```
        {
            Cursor.lockState = CursorLockMode.Locked;
            Cursor.visible = false;
        }
```

6.36.2.3 Update()

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

Every fixed update check if the look shoud be moved

Definition at line 52 of file PlayerLook.cs.

```
        {
            //*the look wil not update when a inventory GUI is open
            if (!THInput.isAnotherInventoryOpen)
            {
                Look();
            }
        }
```

6.36.3 Member Data Documentation

6.36.3.1 cameraTransform

```
Transform BeeGame.Player.PlayerLook.cameraTransform
```

Camera transfom

Definition at line 19 of file PlayerLook.cs.

6.36.3.2 myTransform

```
Transform BeeGame.Player.PlayerLook.myTransform
```

Player transfrom

Definition at line 15 of file PlayerLook.cs.

6.36.3.3 rotationLock

```
float BeeGame.Player.PlayerLook.rotationLock
```

Lock for camera X rotation

Definition at line 24 of file PlayerLook.cs.

6.36.3.4 speed

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

Look move speed

Definition at line 28 of file PlayerLook.cs.

6.36.3.5 xRot

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

Current X rotation

Definition at line 36 of file PlayerLook.cs.

6.36.3.6 yRot

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

Current Y rotation

Definition at line 32 of file PlayerLook.cs.

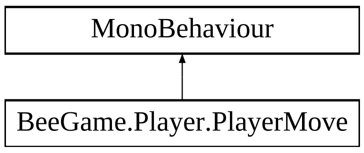
The documentation for this class was generated from the following file:

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

6.37 BeeGame.Player.PlayerMove Class Reference

Moves the player

Inheritance diagram for BeeGame.Player.PlayerMove:



Public Attributes

- float speed = 10f

  Speed of the player
- float gravity = 9.81f

  Gravity of the player
- float maxVelocity = 10f

  Max velocity of the player
- float jumpHeight = 2f

  How high can the player jump

Private Member Functions

- void Awake ()

  Gets the rigidbody and sets its variables
- void FixedUpdate ()

  Updates the player move
- void OnCollisionStay (Collision collision)

  Sets that the player can jump when it hits the ground
- void MovePlayer ()

  Moves the player
- float VerticalJumpSpeed ()

  Vertical Jump speed of the character

Private Attributes

- bool canJump = false

  Can the player jump?
- Rigidbody myRigidBody

  Rigidbody for the player

6.37.1 Detailed Description

Moves the player

Definition at line 14 of file PlayerMove.cs.

6.37.2 Member Function Documentation

6.37.2.1 Awake()

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

Gets the rigidbody and sets its variables

Definition at line 49 of file PlayerMove.cs.

```
        {
            myRigidBody = GetComponent<Rigidbody>();

            //i want to use myown gravity and rotation
            myRigidBody.useGravity = false;
            myRigidBody.freezeRotation = true;
        }
```

6.37.2.2 FixedUpdate()

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

Updates the player move

Definition at line 61 of file PlayerMove.cs.

```
        {
            //If the player is grounded it can move
            if (canJump)
            {
                MovePlayer();
            }

            //adds the downward force
            myRigidBody.AddForce(new Vector3(0, myRigidBody.mass * -gravity, 0));
        }
```

6.37.2.3 MovePlayer()

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

Moves the player

Definition at line 87 of file PlayerMove.cs.

```
        {
            //Calculate the speed we want to achive
            Vector3 targetVelocity = new Vector3(THInput.GetAxis("Horizontal"), 0, THInput.GetAxis("Vertical"));
            targetVelocity = transform.TransformDirection(targetVelocity);
            targetVelocity *= speed;

            //Apply a force to reach the target speed
            Vector3 velocity = myRigidBody.velocity;
            Vector3 velocityChange = (targetVelocity - velocity);

            //Clamping the velocity so that the player does not infinatly accelerate
            velocityChange.x = Mathf.Clamp(velocityChange.x, -maxVelocity, maxVelocity);
            velocityChange.z = Mathf.Clamp(velocityChange.z, -maxVelocity, maxVelocity);
            velocityChange.y = 0;

            //Adds the force to the player so they move in the correct direction
            myRigidBody.AddForce(velocityChange, ForceMode.Impulse);

            //Jumping
            if (canJump && THInput.GetButton("Jump"))
            {
                canJump = false;
                myRigidBody.velocity = new Vector3(velocity.x, VerticalJumpSpeed(), velocity.z);
            }
        }
```

6.37.2.4 OnCollisionStay()

```
void BeeGame.Player.PlayerMove.OnCollisionStay (
              Collision collision ) [private]
```

Sets that the player can jump when it hits the ground

Parameters

collision	What the player hit

Definition at line 77 of file PlayerMove.cs.

```
        {
            canJump = true;
        }
```

6.37.2.5 VerticalJumpSpeed()

`float BeeGame.Player.PlayerMove.VerticalJumpSpeed ( )  [private]`

Vertical Jump speed of the character

Returns

Speed of the jump

Definition at line 118 of file PlayerMove.cs.

```
        {
            //*Gets the correct of fore required for the player to reach the desired apex
            //*Can this be done without Square Root as that take alot of work?
            return Mathf.Sqrt(2 * jumpHeight * gravity);
        }
```

6.37.3 Member Data Documentation

6.37.3.1 canJump

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

Can the player jump?

Definition at line 33 of file PlayerMove.cs.

6.37.3.2 gravity

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

Gravity of the player

Definition at line 24 of file PlayerMove.cs.

6.37.3.3 jumpHeight

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

How high can the player jump

Definition at line 37 of file PlayerMove.cs.

6.37.3.4 maxVelocity

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

Max velocity of the player

Definition at line 28 of file PlayerMove.cs.

6.37.3.5 myRigidBody

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

Rigidbody for the player

Definition at line 42 of file PlayerMove.cs.

6.37.3.6 speed

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

Speed of the player

Definition at line 20 of file PlayerMove.cs.

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

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

6.38 BeeGame.Core.Dictionarys.PrefabDictionary Class Reference

The prefabs avaliable 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 avaliable to spawn in

6.38.1 Detailed Description

The prefabs avaliable to the game

Definition at line 9 of file PrefabDictionary.cs.

6.38.2 Member Function Documentation

6.38.2.1 GetPrefab()

```
static GameObject BeeGame.Core.Dictionarys.PrefabDictionary.GetPrefab (
            string prefab )  [static]
```

Returns a GameObject in the prefab dictionary

Parameters

prefab	Name of th prefab to get

Returns

Prefab of the given name

Definition at line 29 of file PrefabDictionary.cs.

```
        {
            return prefabDictionary[prefab];
        }
```

6.38.2.2 LoadPrefabs()

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

Loads the prefabs into the Dictionary

Definition at line 19 of file PrefabDictionary.cs.

```
        {
            prefabDictionary = Resources.Resources.GetPrefabs();
        }
```

6.38.3 Member Data Documentation

6.38.3.1 prefabDictionary

```
Dictionary<string, GameObject> BeeGame.Core.Dictionarys.PrefabDictionary.prefabDictionary =
new Dictionary<string, GameObject>()  [static], [private]
```

All of the prefabs avaliable to spawn in

Definition at line 14 of file PrefabDictionary.cs.

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

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/Prefab↩
  Dictionary.cs

6.39 BeeGame.Items.QueenBee Class Reference

Public Member Functions

- QueenBee ()
- QueenBee (NormalBee princess, NormalBee drone)
- override int GetHashCode ()

Properties

- NormalBee queen `[get, set]`
  Original princess traits
- NormalBee drone `[get, set]`
  Paired drone traits

6.39.1 Detailed Description

Definition at line 210 of file Bee.cs.

6.39.2 Constructor & Destructor Documentation

6.39.2.1 QueenBee() [1/2]

```
BeeGame.Items.QueenBee.QueenBee ( )
```

Definition at line 222 of file Bee.cs.

```
00222 { }
```

6.39.2.2 QueenBee() [2/2]

```
BeeGame.Items.QueenBee.QueenBee (
              NormalBee princess,
              NormalBee drone )
```

Definition at line 224 of file Bee.cs.

```
        {
            this.queen = princess;
            this.drone = drone;
        }
```

6.39.3 Member Function Documentation

6.39.3.1 GetHashCode()

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

Definition at line 230 of file Bee.cs.

```
        {
            unchecked
            {
                return (int)Int64.Parse($"{queen.GetHashCode()}{drone.GetHashCode()}");
            }
        }
```

6.39.4 Property Documentation

6.39.4.1 drone

```
NormalBee BeeGame.Items.QueenBee.drone  [get], [set]
```

Paired drone traits

Definition at line 220 of file Bee.cs.

6.39.4.2 queen

```
NormalBee BeeGame.Items.QueenBee.queen  [get], [set]
```

Original princess traits

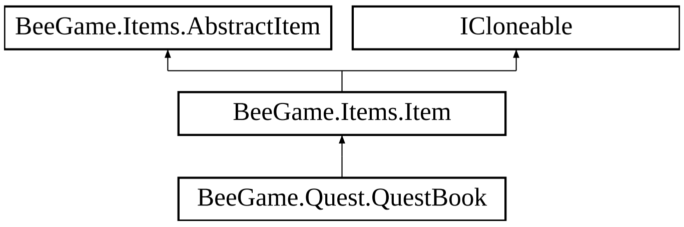
Definition at line 216 of file Bee.cs.

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

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

6.40 BeeGame.Quest.QuestBook Class Reference

Inheritance diagram for BeeGame.Quest.QuestBook:



Public Member Functions

- QuestBook ()
- override bool InteractWithObject ()
- override Sprite GetItemSprite ()

    Returns the sprite for the item
- override int GetHashCode ()

    Returns the hashcode for this Item

Public Attributes

- override int maxStackCount => 1

Additional Inherited Members

6.40.1 Detailed Description

Definition at line 11 of file QuestBook.cs.

6.40.2 Constructor & Destructor Documentation

6.40.2.1 QuestBook()

```
BeeGame.Quest.QuestBook.QuestBook ( )
```

Definition at line 15 of file QuestBook.cs.

```
                                 : base("Quest Book")
        {
        }
```

6.40.3 Member Function Documentation

6.40.3.1 GetHashCode()

`override int BeeGame.Quest.QuestBook.GetHashCode ( ) [virtual]`

Returns the hashcode for this Item

Returns

10

Reimplemented from BeeGame.Items.Item.

Definition at line 34 of file QuestBook.cs.

```
        {
            return 10;
        }
```

6.40.3.2 GetItemSprite()

`override Sprite BeeGame.Quest.QuestBook.GetItemSprite ( ) [virtual]`

Returns the sprite for the item

Returns

Sprite for this item

Reimplemented from BeeGame.Items.Item.

Definition at line 25 of file QuestBook.cs.

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

6.40.3.3 InteractWithObject()

`override bool BeeGame.Quest.QuestBook.InteractWithObject ( ) [virtual]`

Reimplemented from BeeGame.Items.Item.

Definition at line 20 of file QuestBook.cs.

```
        {
            return true;
        }
```

6.40.4 Member Data Documentation

6.40.4.1 maxStackCount

```
override int BeeGame.Quest.QuestBook.maxStackCount => 1
```

Definition at line 13 of file QuestBook.cs.

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

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Quest/QuestBook.cs

6.41 BeeGame.Resources.Resources Class Reference

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

Package Functions

- Resources ()

Static Package Functions

- static Dictionary< string, Sprite > GetSprites ()
- static Dictionary< string, GameObject > GetPrefabs ()

Properties

- static global::System.Resources.ResourceManager ResourceManager `[get]`

  Returns the cached ResourceManager instance used by this class.
- static global::System.Globalization.CultureInfo Culture `[get, set]`

  Overrides the current thread's CurrentUICulture property for all resource lookups using this strongly typed resource class.
- static byte [ ] Prefabs `[get]`

  Looks up a localized resource of type System.Byte[].
- static byte [ ] Sprites `[get]`

  Looks up a localized resource of type System.Byte[].

Static Private Attributes

- static global::System.Resources.ResourceManager resourceMan
- static global::System.Globalization.CultureInfo resourceCulture

6.41.1 Detailed Description

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

Definition at line 26 of file Resources.Designer.cs.

6.41.2 Constructor & Destructor Documentation

6.41.2.1 Resources()

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

Definition at line 33 of file Resources.Designer.cs.

```
00033                              {
00034         }
```

6.41.3 Member Function Documentation

6.41.3.1 GetPrefabs()

```
static Dictionary<string, GameObject> BeeGame.Resources.Resources.GetPrefabs ( )  [static],
[package]
```

Definition at line 118 of file Resources.Designer.cs.

```
00119         {
00120             string[] splitCharacters = new string[] { "," };
00121             object obj = ResourceManager.GetObject("Prefabs",
      resourceCulture);
00122 
00123             string text = System.Text.Encoding.Default.GetString((byte[])obj);
00124             text = text.Remove(0, 3);
00125             string lineText = "";
00126             string[] splitText;
00127             Dictionary<string, GameObject> objects = new Dictionary<string, GameObject>();
00128 
00129             for (int i = 0; i < text.Length; i++)
00130             {
00131                 if(text[i] != '\n')
00132                 {
00133                     lineText += text[i];
00134                 }
00135                 else
00136                 {
00137                     splitText = lineText.Split(splitCharacters, StringSplitOptions.RemoveEmptyEntries);
00138                     lineText = "";
00139                     objects.Add(splitText[0], UnityEngine.Resources.Load("Prefabs/" + splitText[
      1].Remove(splitText[1].Length - 1, 1)) as GameObject);
00140                 }
00141             }
00142 
00143             splitText = lineText.Split(splitCharacters, StringSplitOptions.RemoveEmptyEntries);
00144             lineText = "";
00145             objects.Add(splitText[0], UnityEngine.Resources.Load("Prefabs/" + splitText[1]) as
      GameObject);
00146 
00147             return objects;
00148         }
```

6.41.3.2 GetSprites()

`static Dictionary<string, Sprite> BeeGame.Resources.Resources.GetSprites ( ) [static], [package]`

Definition at line 84 of file Resources.Designer.cs.

```
        {
            string[] splitCharacters = new string[] { "," };
            object obj = ResourceManager.GetObject("Sprites",
    resourceCulture);

            string text = System.Text.Encoding.Default.GetString((byte[])obj);
            string lineText = "";
            string[] splitText;
            Texture2D tex;
            Dictionary<string, Sprite> sprites = new Dictionary<string, Sprite>();

            for (int i = 0; i < text.Length; i++)
            {
                if (text[i] != '\n')
                {
                    lineText += text[i];
                }
                else
                {
                    splitText = lineText.Split(splitCharacters, StringSplitOptions.RemoveEmptyEntries);
                    lineText = "";
                    tex = UnityEngine.Resources.Load("Sprites/" + splitText[1].Remove(splitText[
    1].Length - 1, 1)) as Texture2D;
                    sprites.Add(splitText[0], Sprite.Create(tex, new UnityEngine.Rect(0, 0, tex.
    width, tex.height), Vector2.zero));
                }
            }

            splitText = lineText.Split(splitCharacters, StringSplitOptions.RemoveEmptyEntries);
            lineText = "";
            tex = UnityEngine.Resources.Load("Sprites/" + splitText[1]) as Texture2D;
            sprites.Add(splitText[0], Sprite.Create(tex, new UnityEngine.Rect(0, 0, tex.width,
    tex.height), Vector2.zero));

            return sprites;
        }
```

6.41.4 Member Data Documentation

6.41.4.1 resourceCulture

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

Definition at line 30 of file Resources.Designer.cs.

6.41.4.2 resourceMan

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

Definition at line 28 of file Resources.Designer.cs.

6.41.5 Property Documentation

6.41.5.1 Culture

```
global.System.Globalization.CultureInfo BeeGame.Resources.Resources.Culture  [static], [get],
[set], [package]
```

Overrides the current thread's CurrentUICulture property for all resource lookups using this strongly typed resource class.

Definition at line 55 of file Resources.Designer.cs.

6.41.5.2 Prefabs

```
byte [] BeeGame.Resources.Resources.Prefabs  [static], [get], [package]
```

Looks up a localized resource of type System.Byte[].

Definition at line 67 of file Resources.Designer.cs.

6.41.5.3 ResourceManager

```
global.System.Resources.ResourceManager BeeGame.Resources.Resources.ResourceManager  [static],
[get], [package]
```

Returns the cached ResourceManager instance used by this class.

Definition at line 40 of file Resources.Designer.cs.

6.41.5.4 Sprites

```
byte [] BeeGame.Resources.Resources.Sprites  [static], [get], [package]
```

Looks up a localized resource of type System.Byte[].

Definition at line 77 of file Resources.Designer.cs.

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

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Resources/Resources.↩
Designer.cs

6.42 BeeGame.Terrain.Chunks.SaveChunk Class Reference

Saves a Chunks modified Blocks for save optimisation

Public Member Functions

- SaveChunk (Block[„] blockArray)

    Will search all the the given Blocks for modified blocks

Public Attributes

- Dictionary< ChunkWorldPos, Block > blocks = new Dictionary<ChunkWorldPos, Block>()

    Blocks to be saved

6.42.1 Detailed Description

Saves a Chunks modified Blocks for save optimisation

Definition at line 12 of file SaveChunk.cs.

6.42.2 Constructor & Destructor Documentation

6.42.2.1 SaveChunk()

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

Will search all the the given Blocks for modified blocks

Parameters

blockArray	Chunks blocks (Must be [16, 16, 16])

Definition at line 23 of file SaveChunk.cs.

```
        {
            for (int x = 0; x < Chunk.chunkSize; x++)
            {
                for (int y = 0; y < Chunk.chunkSize; y++)
                {
                    for (int z = 0; z < Chunk.chunkSize; z++)
                    {
                        //* if the block has changed save it
                        if (blockArray[x, y, z].changed)
                            blocks.Add(new ChunkWorldPos(x, y, z), blockArray[x, y, z]);
                    }
                }
            }
        }
```

6.42.3 Member Data Documentation

6.42.3.1 blocks

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

Blocks to be saved

Definition at line 17 of file SaveChunk.cs.

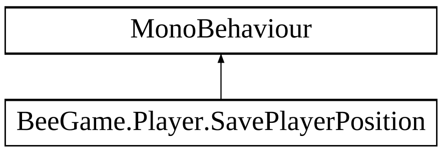
The documentation for this class was generated from the following file:

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/SaveChunk.cs

6.43 BeeGame.Player.SavePlayerPosition Class Reference

Saves the player postion

Inheritance diagram for BeeGame.Player.SavePlayerPosition:



Private Member Functions

- void Update ()

  Saves the player every 1000 frames

Private Attributes

- int counter = 0

  Timer for saveing the player

6.43.1 Detailed Description

Saves the player postion

Definition at line 9 of file SavePlayerPosition.cs.

6.43.2 Member Function Documentation

6.43.2.1 Update()

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

Saves the player every 1000 frames

Definition at line 19 of file SavePlayerPosition.cs.

```
        {
            if(counter == 0)
            {
                counter = 1000;
                Serialization.Serialization.SavePlayerPosition(transform);
                //print("saved player");
            }

            counter--;
        }
```

6.43.3 Member Data Documentation

6.43.3.1 counter

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

Timer for saveing the player

Definition at line 14 of file SavePlayerPosition.cs.

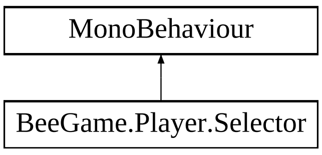
The documentation for this class was generated from the following file:

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

6.44 BeeGame.Player.Selector Class Reference

Moves the Block selector

Inheritance diagram for BeeGame.Player.Selector:



Public Attributes

- GameObject selector

  Selector
- PlayerInventory playerInventory

  Player Inventory
- LayerMask layers

  Layers for the selector to look at
- int selectedHotbarSlot = 27

  What slot in the hotbar is selected

Private Member Functions

- void Awake ()

  Make the selector
- void FixedUpdate ()

  Updates the selector if an inventory is not open
- void Update ()

  Breaks and places a Block if an inventory is no open
- void UpdateSelector ()

  Updates teh selectors position
- void SelectedSlot ()

  Chanages what slot in the hotbar is currently selected by the player
- void BreakBlock ()

  Breaks the Block in the selectors postion
- void PlaceBlock ()

  Places s Block in the selector postion

Private Attributes

- RaycastHit hit

  Where the raycast hit

6.44.1 Detailed Description

Moves the Block selector

Definition at line 15 of file Selector.cs.

6.44.2 Member Function Documentation

6.44.2.1 Awake()

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

Make the selector

Definition at line 47 of file Selector.cs.

```
        {
            selector = Instantiate(selector);
        }
```

6.44.2.2 BreakBlock()

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

Breaks the Block in the selectors postion

Definition at line 123 of file Selector.cs.

```
        {
            Chunk chunk = GetChunk(selector.transform.position);

            Block block = chunk.world.GetBlock((int)selector.transform.position.x, (int)selector.transform.position.y, (int)selector.transform.position.z);

            if (!block.breakable)
                return;

            chunk.world.SetBlock((int)selector.transform.position.x, (int)selector.transform.position.y, (int)selector.transform.position.z, new Air(), true);
            //* set to changed so when block is placed down again it will be saved
            block.changed = true;
            block.BreakBlock(selector.transform.position);
        }
```

6.44.2.3 FixedUpdate()

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

Updates the selector if an inventory is not open

Definition at line 55 of file Selector.cs.

```
        {
            if(!isAnotherInventoryOpen)
                UpdateSelector();
        }
```

6.44.2.4 PlaceBlock()

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

Places s Block in the selector postion

Definition at line 141 of file Selector.cs.

```
        {
            Chunk chunk = GetChunk(selector.transform.position);

            if (chunk == null)
                return;

            if (!chunk.GetBlock((int)selector.transform.position.x - chunk.chunkWorldPos.x, (int)selector.transform.position.y - chunk.chunkWorldPos.y, (int)selector.transform.position.z - chunk.chunkWorldPos.z).InteractWithBlock(playerInventory))
                //* gets the item in the hotbar and if the item is placeable place it
                if (transform.parent.GetComponentInChildren<PlayerInventory>().GetItemFromHotBar(selectedHotbarSlot, out Item blockToPlace))
                    chunk.world.SetBlock((int)(selector.transform.position.x + hit.normal.x), (int)(selector.transform.position.y + hit.normal.y), (int)(selector.transform.position.z + hit.normal.z), (Block)blockToPlace.CloneObject(), true);
        }
```

6.44.2.5 SelectedSlot()

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

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

Definition at line 98 of file Selector.cs.

```
        {
            //* adds 1 to the selected slot and if that is out of range set it to the first hotbar slot
            if(Input.GetAxis("Mouse ScrollWheel") > 0)
            {
                selectedHotbarSlot += 1;
                if (selectedHotbarSlot == 36)
                    selectedHotbarSlot = 27;
            }
            //* removes one from the hotbar selector and if the selector would be inside the inventory set it to the last slot in the hotbar
            else if (Input.GetAxis("Mouse ScrollWheel") < 0)
            {
                selectedHotbarSlot -= 1;
                if (selectedHotbarSlot == 26)
                    selectedHotbarSlot = 35;
            }

            transform.parent.GetComponentInChildren<PlayerInventory>().SelectedSlot(selectedHotbarSlot);
        }
```

6.44.2.6 Update()

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

Breaks and places a Block if an inventory is no open

Definition at line 64 of file Selector.cs.

```
        {
            if (!isAnotherInventoryOpen)
            {
                if (GetButtonDown("Break Block"))
                    BreakBlock();
                if (GetButtonDown("Place"))
                    PlaceBlock();
            }
        }
```

6.44.2.7 UpdateSelector()

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

Updates teh selectors position

Definition at line 80 of file Selector.cs.

```
        {
            if (Physics.Raycast(transform.position, transform.forward, out hit, 15, layers))
            {
                selector.SetActive(true);
                selector.transform.position = GetBlockPos(hit);
                //*selector.SetActive(BlockInPosition(GetBlockPos(hit), hit.collider.GetComponent<Chunk>()));
            }
            else
            {
                selector.SetActive(false);
            }
            SelectedSlot();
        }
```

6.44.3 Member Data Documentation

6.44.3.1 hit

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

Where the raycast hit

Definition at line 35 of file Selector.cs.

6.44.3.2 layers

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

Layers for the selector to look at

Definition at line 31 of file Selector.cs.

6.44.3.3 playerInventory

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

Player Inventory

Definition at line 26 of file Selector.cs.

6.44.3.4 selectedHotbarSlot

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

What slot in the hotbar is selected

Definition at line 40 of file Selector.cs.

6.44.3.5 selector

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

Selector

Definition at line 21 of file Selector.cs.

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

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

6.45 BeeGame.Serialization.Serialization Class Reference

Serializes and Deserialises things

Static Public Member Functions

- static void MakeDirectorys ()

  Sets the paths for the save files
- static void DeleteFile (string fileName)

  Deletes the given file if it exists, Starts in Application.dataPath
- static void SavePlayerPosition (Transform positon)

  Saves the player positon, rotation, and scale
- static void LoadPlayerPosition (Transform playerTransfom)

  Loads the players positon, roatation, and scale if it has previously been saved
- static void SerializeInventory (Inventory.Inventory inventory, string inventoryName)

  Serializes a given Inventory
- static void DeSerializeInventory (Inventory.Inventory inventory, string inventoryName)

  Deserializesd an Inventory from its name into a given inventory
- static void SaveChunk (Chunk chunk)

  Saves a given Chunk if a block in it has been changed
- static bool LoadChunk (Chunk chunk)

  Load a Chunk
- static string FileName (ChunkWorldPos pos)

  Sets the file name of the Chunk

Static Public Attributes

- static string worldName = "World"

  Name if the world. If multiple world are ever added
- static string saveFolderName = "Saves"

  Save folder

Static Private Member Functions

- static void SaveFile (object obj, string file)

  Saves the given data in the given file
- static object LoadFile (string file)

  Loads the file at the given path

Static Private Attributes

- static string savePath

  Path to save things

6.45.1 Detailed Description

Serializes and Deserialises things

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

Definition at line 19 of file Serialization.cs.

6.45.2 Member Function Documentation

6.45.2.1 DeleteFile()

```
static void BeeGame.Serialization.Serialization.DeleteFile (
              string fileName )  [static]
```

Deletes the given file if it exists, Starts in Application.dataPath

Parameters

fileName	File to delete

Definition at line 51 of file Serialization.cs.

```
        {
            string[] file = Directory.GetFiles(Application.dataPath + "/Saves", "*.dat", SearchOption.AllDirectories);

            string[] splitCharacters = { "/", "\\", ".dat" };

            for (int i = 0; i < file.Length; i++)
            {
                string[] temp = file[i].Split(splitCharacters, System.StringSplitOptions.RemoveEmptyEntries);

                if(temp[temp.Length - 1] == fileName)
                {
                    File.Delete(file[i]);

                    return;
                }
            }
        }
```

6.45.2.2 DeSerializeInventory()

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

Deserializesd an Inventory from its name into a given *inventory*

Parameters

inventory	Inventory to apply the data to
inventoryName	Inventory to deserialize

Definition at line 132 of file Serialization.cs.

```
        {
            //* make the path
            string inventorySavePath = $"{savePath}/Inventorys/{inventoryName}.dat";

            //* checks that the file exists
            if (!File.Exists(inventorySavePath))
            {
                for (int i = 0; i < inventory.items.itemsInInventory.Length; i++)
                {
                    inventory.items.itemsInInventory[i] = null;
                }

                SerializeInventory(inventory, inventoryName);

                return;
            }

            inventory.SetAllItems((ItemsInInventory)LoadFile($"{inventorySavePath}"
      ));
        }
```

6.45.2.3 FileName()

```
static string BeeGame.Serialization.Serialization.FileName (
              ChunkWorldPos pos )  [static]
```

Sets the file name of the Chunk

Parameters

pos	Position of teh Chunk

Returns

The string of pos

Definition at line 204 of file Serialization.cs.

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

6.45.2.4 LoadChunk()

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

Load a Chunk

Parameters

chunk	

Returns

Definition at line 179 of file Serialization.cs.

```
        {
            //* gets the save file
            string saveFile = $"{savePath}/{FileName(chunk.chunkWorldPos)}.dat";

            //* if the file does not exist return false
            if (!File.Exists(saveFile))
                return false;

            //* set all of the changed blocks in the chunk
            SaveChunk save = (SaveChunk)LoadFile(saveFile);

            foreach (var block in save.blocks)
            {
                chunk.blocks[block.Key.x, block.Key.y, block.Key.z] = block.Value;
            }

            return true;
        }
```

6.45.2.5 LoadFile()

```
static object BeeGame.Serialization.Serialization.LoadFile (
            string file )  [static], [private]
```

Loads the file at the given path

Parameters

file	File to load

Returns

returns the loaded file as an object

Definition at line 241 of file Serialization.cs.

```
        {
            BinaryFormatter bf = new BinaryFormatter();
            FileStream fs = new FileStream(file, FileMode.Open);

            try
            {
                return bf.Deserialize(fs);
            }
            catch(SerializationException e)
            {
                Debug.Log($"Deserialization Exception {e}");
                throw new SerializationException();
            }
            finally
            {
                fs.Close();
            }
        }
```

6.45.2.6 LoadPlayerPosition()

```
static void BeeGame.Serialization.Serialization.LoadPlayerPosition (
              Transform playerTransfom )  [static]
```

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

Parameters

playerTransfom	Transform to apply the data to

Definition at line 92 of file Serialization.cs.

```
        {
            string playerPosSavePath = $"{savePath}/player.dat";

            if (!File.Exists(playerPosSavePath))
                return;

            THVector3[] pos = (THVector3[])LoadFile(playerPosSavePath);

            playerTransfom.position = pos[0];
            playerTransfom.rotation = (Quaternion)pos[1];
            playerTransfom.localScale = pos[2];
        }
```

6.45.2.7 MakeDirectorys()

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

Sets the paths for the save files

Definition at line 39 of file Serialization.cs.

```
        {
            savePath = $"{Application.dataPath}/{saveFolderName}/{worldName}";

            if (!(Directory.Exists(savePath)))
                Directory.CreateDirectory(savePath);
        }
```

6.45.2.8 SaveChunk()

```
static void BeeGame.Serialization.Serialization.SaveChunk (
              Chunk chunk )  [static]
```

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

Parameters

chunk	

Definition at line 159 of file Serialization.cs.

```
        {
            //* saves the blocks
            SaveChunk save = new SaveChunk(chunk.blocks);

            //* if no block was changed return early
            if (save.blocks.Count == 0)
                return;

            //* otherwise save the file
            string saveFile = $"{savePath}/{FileName(chunk.chunkWorldPos)}.dat";

            SaveFile(save, saveFile);
        }
```

6.45.2.9 SaveFile()

```
static void BeeGame.Serialization.Serialization.SaveFile (
              object obj,
              string file )  [static], [private]
```

Saves the given data in the given file

Parameters

obj	Object to save
file	File path to save to

Definition at line 216 of file Serialization.cs.

```
        {
            BinaryFormatter bf = new BinaryFormatter();
            FileStream fs = new FileStream(file, FileMode.OpenOrCreate);

            try
            {
                bf.Serialize(fs, obj);
            }
            catch(SerializationException e)
            {
                Debug.Log($"Serialization Exception: {e}");
                throw new SerializationException();
            }
            finally
            {
                fs.Close();
            }
        }
```

6.45.2.10 SavePlayerPosition()

```
static void BeeGame.Serialization.Serialization.SavePlayerPosition (
              Transform positon )  [static]
```

Saves the player positon, rotation, and scale

Parameters

positon	Transform to get the data from

Definition at line 75 of file Serialization.cs.

```
        {
            THVector3[] playerTransform = new THVector3[3];

            playerTransform[0] = positon.position;
            playerTransform[1] = positon.rotation.eulerAngles;
            playerTransform[2] = positon.localScale;

            string playerPosSavePath = $"{savePath}/player.dat";

            SaveFile(playerTransform, playerPosSavePath);
        }
```

6.45.2.11 SerializeInventory()

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

Serializes a given Inventory

Parameters

inventory	Invenotry to Serialize
inventoryName	Name of the inventory

The name of the inventory for the player is "PlayerInventory".
For all other ivnetorys the name is the block type + its position eg, Apiay@0, 0, 0

Definition at line 117 of file Serialization.cs.

```
        {
            string inventorySavePath = $"{savePath}/Inventorys";

            if (!Directory.Exists(inventorySavePath))
                Directory.CreateDirectory(inventorySavePath);

            SaveFile(inventory.GetAllItems(), $"{inventorySavePath}/{inventoryName}.dat");
        }
```

6.45.3 Member Data Documentation

6.45.3.1 saveFolderName

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

Save folder

Definition at line 29 of file Serialization.cs.

6.45.3.2 savePath

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

Path to save things

Definition at line 33 of file Serialization.cs.

6.45.3.3 worldName

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

Name if the world. If multiple world are ever added

Definition at line 25 of file Serialization.cs.

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

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Serialization/Serialization.cs

6.46 BeeGame.Terrain.LandGeneration.Noise.SimplexNoise Class Reference

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

Static Public Member Functions

- static float Generate (float x)

  1D simplex noise
- static float Generate (float x, float y)

  2D simplex noise
- static float Generate (float x, float y, float z)

Static Public Attributes

- static byte [ ] perm

Static Private Member Functions

- static int FastFloor (float x)
- static int Mod (int x, int m)
- static float grad (int hash, float x)
- static float grad (int hash, float x, float y)
- static float grad (int hash, float x, float y, float z)
- static float grad (int hash, float x, float y, float z, float t)

6.46.1 Detailed Description

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

Definition at line 37 of file SimplexNoise.cs.

6.46.2 Member Function Documentation

6.46.2.1 FastFloor()

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

Definition at line 272 of file SimplexNoise.cs.

```
        {
            return (x > 0) ? ((int)x) : (((int)x) - 1);
        }
```

6.46.2.2 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.

```
        {
            int i0 = FastFloor(x);
            int i1 = i0 + 1;
            float x0 = x - i0;
            float x1 = x0 - 1.0f;

            float n0, n1;

            float t0 = 1.0f - x0 * x0;
            t0 *= t0;
            n0 = t0 * t0 * grad(perm[i0 & 0xff], x0);

            float t1 = 1.0f - x1 * x1;
            t1 *= t1;
            n1 = t1 * t1 * grad(perm[i1 & 0xff], x1);
            //* The maximum value of this noise is 8*(3/4)^4 = 2.53125
            //* A factor of 0.395 scales to fit exactly within [-1,1]
            return 0.395f * (n0 + n1);
        }
```

6.46.2.3 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.

```
        {
            const float F2 = 0.366025403f; //* F2 = 0.5*(sqrt(3.0)-1.0)
            const float G2 = 0.211324865f; //* G2 = (3.0-Math.sqrt(3.0))/6.0

            float n0, n1, n2; //* Noise contributions from the three corners

            //* Skew the input space to determine which simplex cell we're in
            float s = (x + y) * F2; //* Hairy factor for 2D
            float xs = x + s;
            float ys = y + s;
            int i = FastFloor(xs);
            int j = FastFloor(ys);

            float t = (float)(i + j) * G2;
            float X0 = i - t; //* Unskew the cell origin back to (x,y) space
            float Y0 = j - t;
            float x0 = x - X0; //* The x,y distances from the cell origin
            float y0 = y - Y0;

            //* For the 2D case, the simplex shape is an equilateral triangle.
            //* Determine which simplex we are in.
            int i1, j1; //* Offsets for second (middle) corner of simplex in (i,j) coords
            if (x0 > y0) { i1 = 1; j1 = 0; } //* lower triangle, XY order: (0,0)->(1,0)->(1,1)
            else { i1 = 0; j1 = 1; }      //* upper triangle, YX order: (0,0)->(0,1)->(1,1)

            //* A step of (1,0) in (i,j) means a step of (1-c,-c) in (x,y), and
            //* a step of (0,1) in (i,j) means a step of (-c,1-c) in (x,y), where
            //* c = (3-sqrt(3))/6

            float x1 = x0 - i1 + G2; //* Offsets for middle corner in (x,y) unskewed coords
            float y1 = y0 - j1 + G2;
            float x2 = x0 - 1.0f + 2.0f * G2; //* Offsets for last corner in (x,y) unskewed coords
            float y2 = y0 - 1.0f + 2.0f * G2;

            //* Wrap the integer indices at 256, to avoid indexing perm[] out of bounds
            int ii = i % 256;
            int jj = j % 256;

            //* Calculate the contribution from the three corners
            float t0 = 0.5f - x0 * x0 - y0 * y0;
            if (t0 < 0.0f) n0 = 0.0f;
            else
            {
                t0 *= t0;
                n0 = t0 * t0 * grad(perm[ii + perm[jj]], x0, y0);
            }

            float t1 = 0.5f - x1 * x1 - y1 * y1;
            if (t1 < 0.0f) n1 = 0.0f;
            else
            {
                t1 *= t1;
                n1 = t1 * t1 * grad(perm[ii + i1 + perm[jj + j1]], x1, y1);
            }

            float t2 = 0.5f - x2 * x2 - y2 * y2;
            if (t2 < 0.0f) n2 = 0.0f;
            else
            {
                t2 *= t2;
                n2 = t2 * t2 * grad(perm[ii + 1 + perm[jj + 1]], x2, y2);
            }

            //* Add contributions from each corner to get the final noise value.
            //* The result is scaled to return values in the interval [-1,1].
            return 40.0f * (n0 + n1 + n2); //* TODO: The scale factor is preliminary!
        }
```

6.46.2.4 Generate() [3/3]

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

Definition at line 141 of file SimplexNoise.cs.

```
        {
            //* Simple skewing factors for the 3D case
            const float F3 = 0.333333333f;
            const float G3 = 0.166666667f;

            float n0, n1, n2, n3; //* Noise contributions from the four corners

            //* Skew the input space to determine which simplex cell we're in
            float s = (x + y + z) * F3; //* Very nice and simple skew factor for 3D
            float xs = x + s;
            float ys = y + s;
            float zs = z + s;
            int i = FastFloor(xs);
            int j = FastFloor(ys);
            int k = FastFloor(zs);

            float t = (float)(i + j + k) * G3;
            float X0 = i - t; //* Unskew the cell origin back to (x,y,z) space
            float Y0 = j - t;
            float Z0 = k - t;
            float x0 = x - X0; //* The x,y,z distances from the cell origin
            float y0 = y - Y0;
            float z0 = z - Z0;

            //* For the 3D case, the simplex shape is a slightly irregular tetrahedron.
            //* Determine which simplex we are in.
            int i1, j1, k1; //* Offsets for second corner of simplex in (i,j,k) coords
            int i2, j2, k2; //* Offsets for third corner of simplex in (i,j,k) coords

            /* This code would benefit from a backport from the GLSL version! */
            if (x0 >= y0)
            {
                if (y0 >= z0)
                { i1 = 1; j1 = 0; k1 = 0; i2 = 1; j2 = 1; k2 = 0; } //* X Y Z order
                else if (x0 >= z0) { i1 = 1; j1 = 0; k1 = 0; i2 = 1; j2 = 0; k2 = 1; } //* X Z Y order
                else { i1 = 0; j1 = 0; k1 = 1; i2 = 1; j2 = 0; k2 = 1; } //* Z X Y order
            }
            else
            { //* x0<y0
                if (y0 < z0) { i1 = 0; j1 = 0; k1 = 1; i2 = 0; j2 = 1; k2 = 1; } //* Z Y X order
                else if (x0 < z0) { i1 = 0; j1 = 1; k1 = 0; i2 = 0; j2 = 1; k2 = 1; } //* Y Z X order
                else { i1 = 0; j1 = 1; k1 = 0; i2 = 1; j2 = 1; k2 = 0; } //∗ Y X Z order
            }

            //∗ A step of (1,0,0) in (i,j,k) means a step of (1-c,-c,-c) in (x,y,z),
            //∗ a step of (0,1,0) in (i,j,k) means a step of (-c,1-c,-c) in (x,y,z), and
            //∗ a step of (0,0,1) in (i,j,k) means a step of (-c,-c,1-c) in (x,y,z), where
            //∗ c = 1/6.

            float x1 = x0 - i1 + G3; //∗ Offsets for second corner in (x,y,z) coords
            float y1 = y0 - j1 + G3;
            float z1 = z0 - k1 + G3;
            float x2 = x0 - i2 + 2.0f ∗ G3; //∗ Offsets for third corner in (x,y,z) coords
            float y2 = y0 - j2 + 2.0f ∗ G3;
            float z2 = z0 - k2 + 2.0f ∗ G3;
            float x3 = x0 - 1.0f + 3.0f ∗ G3; //∗ Offsets for last corner in (x,y,z) coords
            float y3 = y0 - 1.0f + 3.0f ∗ G3;
            float z3 = z0 - 1.0f + 3.0f ∗ G3;

            //∗ Wrap the integer indices at 256, to avoid indexing perm[] out of bounds
            int ii = Mod(i, 256);
            int jj = Mod(j, 256);
            int kk = Mod(k, 256);

            //∗ Calculate the contribution from the four corners
            float t0 = 0.6f - x0 ∗ x0 - y0 ∗ y0 - z0 ∗ z0;
            if (t0 < 0.0f) n0 = 0.0f;
            else
            {
                t0 ∗= t0;
                n0 = t0 ∗ t0 ∗ grad(perm[ii + perm[jj + perm[kk]]], x0, y0, z0);
            }

            float t1 = 0.6f - x1 ∗ x1 - y1 ∗ y1 - z1 ∗ z1;
            if (t1 < 0.0f) n1 = 0.0f;
            else
            {
                t1 ∗= t1;
                n1 = t1 ∗ t1 ∗ grad(perm[ii + i1 + perm[jj + j1 + perm[kk + k1]]], x1, y1, z1);
            }

            float t2 = 0.6f - x2 ∗ x2 - y2 ∗ y2 - z2 ∗ z2;
            if (t2 < 0.0f) n2 = 0.0f;
            else
            {
                t2 ∗= t2;
                n2 = t2 ∗ t2 ∗ grad(perm[ii + i2 + perm[jj + j2 + perm[kk + k2]]], x2, y2, z2);
            }

            float t3 = 0.6f - x3 ∗ x3 - y3 ∗ y3 - z3 ∗ z3;
            if (t3 < 0.0f) n3 = 0.0f;
            else
            {
                t3 ∗= t3;
                n3 = t3 ∗ t3 ∗ grad(perm[ii + 1 + perm[jj + 1 + perm[kk + 1]]], x3, y3, z3);
            }

            //∗ Add contributions from each corner to get the final noise value.
            //∗ The result is scaled to stay just inside [-1,1]
            return 32.0f ∗ (n0 + n1 + n2 + n3); //∗ TODO: The scale factor is preliminary!
        }
```

6.46.2.5 grad() [1/4]

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

Definition at line 283 of file SimplexNoise.cs.

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

6.46.2.6 grad() [2/4]

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

Definition at line 291 of file SimplexNoise.cs.

```
        {
            int h = hash & 7;      //* Convert low 3 bits of hash code
            float u = h < 4 ? x : y;  //* into 8 simple gradient directions,
            float v = h < 4 ? y : x;  //* and compute the dot product with (x,y).
            return ((h & 1) != 0 ? -u : u) + ((h & 2) != 0 ? -2.0f * v : 2.0f * v);
        }
```

6.46.2.7 grad() [3/4]

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

Definition at line 299 of file SimplexNoise.cs.

```
        {
            int h = hash & 15;     //* Convert low 4 bits of hash code into 12 simple
            float u = h < 8 ? x : y; //* gradient directions, and compute dot product.
            float v = h < 4 ? y : h == 12 || h == 14 ? x : z; //* Fix repeats at h = 12 to 15
            return ((h & 1) != 0 ? -u : u) + ((h & 2) != 0 ? -v : v);
        }
```

6.46.2.8 grad() [4/4]

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

Definition at line 307 of file SimplexNoise.cs.

```
        {
            int h = hash & 31;      //* Convert low 5 bits of hash code into 32 simple
            float u = h < 24 ? x : y; //* gradient directions, and compute dot product.
            float v = h < 16 ? y : z;
            float w = h < 8 ? z : t;
            return ((h & 1) != 0 ? -u : u) + ((h & 2) != 0 ? -v : v) + ((h & 4) != 0 ? -w : w);
        }
```

6.46.2.9 Mod()

```
static int BeeGame.Terrain.LandGeneration.Noise.SimplexNoise.Mod (
              int x,
              int m )  [static], [private]
```

Definition at line 277 of file SimplexNoise.cs.

```
              {
                  int a = x % m;
                  return a < 0 ? a + m : a;
              }
```

6.46.3 Member Data Documentation

6.46.3.1 perm

```
byte [] BeeGame.Terrain.LandGeneration.Noise.SimplexNoise.perm  [static]
```

Initial value:

```
= new byte[512] { 151,160,137,91,90,15,
              131,13,201,95,96,53,194,233,7,225,140,36,103,30,69,142,8,99,37,240,21,10,23,
              190, 6,148,247,120,234,75,0,26,197,62,94,252,219,203,117,35,11,32,57,177,33,
              88,237,149,56,87,174,20,125,136,171,168, 68,175,74,165,71,134,139,48,27,166,
              77,146,158,231,83,111,229,122,60,211,133,230,220,105,92,41,55,46,245,40,244,
              102,143,54, 65,25,63,161, 1,216,80,73,209,76,132,187,208, 89,18,169,200,196,
              135,130,116,188,159,86,164,100,109,198,173,186, 3,64,52,217,226,250,124,123,
              5,202,38,147,118,126,255,82,85,212,207,206,59,227,47,16,58,17,182,189,28,42,
              223,183,170,213,119,248,152, 2,44,154,163, 70,221,153,101,155,167, 43,172,9,
              129,22,39,253, 19,98,108,110,79,113,224,232,178,185, 112,104,218,246,97,228,
              251,34,242,193,238,210,144,12,191,179,162,241, 81,51,145,235,249,14,239,107,
              49,192,214, 31,181,199,106,157,184, 84,204,176,115,121,50,45,127, 4,150,254,
              138,236,205,93,222,114,67,29,24,72,243,141,128,195,78,66,215,61,156,180,
              151,160,137,91,90,15,
              131,13,201,95,96,53,194,233,7,225,140,36,103,30,69,142,8,99,37,240,21,10,23,
              190, 6,148,247,120,234,75,0,26,197,62,94,252,219,203,117,35,11,32,57,177,33,
              88,237,149,56,87,174,20,125,136,171,168, 68,175,74,165,71,134,139,48,27,166,
              77,146,158,231,83,111,229,122,60,211,133,230,220,105,92,41,55,46,245,40,244,
              102,143,54, 65,25,63,161, 1,216,80,73,209,76,132,187,208, 89,18,169,200,196,
              135,130,116,188,159,86,164,100,109,198,173,186, 3,64,52,217,226,250,124,123,
              5,202,38,147,118,126,255,82,85,212,207,206,59,227,47,16,58,17,182,189,28,42,
              223,183,170,213,119,248,152, 2,44,154,163, 70,221,153,101,155,167, 43,172,9,
              129,22,39,253, 19,98,108,110,79,113,224,232,178,185, 112,104,218,246,97,228,
              251,34,242,193,238,210,144,12,191,179,162,241, 81,51,145,235,249,14,239,107,
              49,192,214, 31,181,199,106,157,184, 84,204,176,115,121,50,45,127, 4,150,254,
              138,236,205,93,222,114,67,29,24,72,243,141,128,195,78,66,215,61,156,180
            }
```

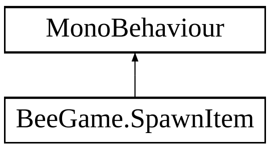
Definition at line 244 of file SimplexNoise.cs.

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

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/LandGeneration/↩
  Noise/SimplexNoise.cs

6.47 BeeGame.SpawnItem Class Reference

Inheritance diagram for BeeGame.SpawnItem:



Private Member Functions

- void Start ()
- void OnDrawGizmos ()

6.47.1 Detailed Description

Definition at line 12 of file SpawnITem.cs.

6.47.2 Member Function Documentation

6.47.2.1 OnDrawGizmos()

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

Definition at line 49 of file SpawnITem.cs.

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

6.47.2.2 Start()

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

Definition at line 14 of file SpawnITem.cs.

```
        {
            GameObject go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as
      GameObject, transform.position, Quaternion.identity) as GameObject;
            go.GetComponent<ItemGameObject>().item = new Bee(
      BeeType.DRONE, new NormalBee() { pSpecies = BeeSpecies.FOREST });

            go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
       transform.position, Quaternion.identity) as GameObject;
            go.GetComponent<ItemGameObject>().item = new Bee(
      BeeType.PRINCESS, new NormalBee() { pSpecies = BeeSpecies.FOREST });

            go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
       transform.position, Quaternion.identity) as GameObject;
            go.GetComponent<ItemGameObject>().item = new Bee(
      BeeType.DRONE, new NormalBee() { pSpecies = BeeSpecies.COMMON, sSpecies =
      BeeSpecies.COMMON });

            go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
       transform.position, Quaternion.identity) as GameObject;
            go.GetComponent<ItemGameObject>().item = new Bee(
      BeeType.PRINCESS, new NormalBee() { pSpecies = BeeSpecies.COMMON, sSpecies =
      BeeSpecies.COMMON });

            //go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
       transform.position, Quaternion.identity) as GameObject;
            //go.GetComponent<ItemGameObject>().item = new Bee(BeeType.QUEEN, new QueenBee());

            go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
       transform.position, Quaternion.identity) as GameObject;
            go.GetComponent<ItemGameObject>().item = new HoneyComb(
      HoneyCombType.ICEY);

            go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
       transform.position, Quaternion.identity) as GameObject;
            go.GetComponent<ItemGameObject>().item = new HoneyComb(
      HoneyCombType.HONEY);

            go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
       transform.position, Quaternion.identity) as GameObject;
            go.GetComponent<ItemGameObject>().item = new Chest();
            go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
       transform.position, Quaternion.identity) as GameObject;
            go.GetComponent<ItemGameObject>().item = new Chest();

            go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
       transform.position, Quaternion.identity) as GameObject;
            go.GetComponent<ItemGameObject>().item = new Apiary();

            go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
       transform.position, Quaternion.identity) as GameObject;
            go.GetComponent<ItemGameObject>().item = new
      CraftingTable();
        }
```

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

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/SpawnITem.cs

6.48 BeeGame.Core.Dictionarys.SpriteDictionary Class Reference

All of the sprites avaliable 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 avaliable to spawn in

6.48.1 Detailed Description

All of the sprites avaliable to the game

Definition at line 9 of file SpriteDictionary.cs.

6.48.2 Member Function Documentation

6.48.2.1 GetSprite()

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

Get a sprite of the given name

Parameters

spriteName	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.

```
        {
            itemSpriteDictionary.TryGetValue(spriteName, out Sprite sprite);

            if (sprite == null)
                return new Sprite();

            return sprite;
        }
```

6.48.2.2 LoadSprites()

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

Loads the sprites into the dictionary

Definition at line 34 of file SpriteDictionary.cs.

```
        {
            itemSpriteDictionary = Resources.Resources.GetSprites();
        }
```

6.48.3 Member Data Documentation

6.48.3.1 itemSpriteDictionary

```
Dictionary<string, Sprite> BeeGame.Core.Dictionarys.SpriteDictionary.itemSpriteDictionary =
new Dictionary<string, Sprite>()  [static], [private]
```

All of the sprites avaliable to spawn in

Definition at line 14 of file SpriteDictionary.cs.

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

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/Sprite↩ Dictionary.cs

6.49 BeeGame.Terrain.LandGeneration.Terrain Class Reference

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

Static Public Member Functions

- static ChunkWorldPos GetBlockPos (THVector3 pos)

  Gets a block postion from a THVector3
- static THVector3 GetBlockPos (RaycastHit hit)

  Returns the positon of the block hit as a THVector3
- static ChunkWorldPos GetBlockPosFromRayCast (RaycastHit hit)

  GetBlockPos(THVector3) does the same thing but returns a ChunkWorldPos
- static float Round (float pos, float norm, bool adjacent=false)

  Rounds the given pos to the correct position
- static ChunkWorldPos GetBlockPos (RaycastHit hit, bool adjacent=false)

  Gets a Chunks world positon
- static Block GetBlock (RaycastHit hit, bool adjacent=false)

  Get a Block at the given position
- static Block GetBlock (THVector3 pos)
- static bool BlockInPosition (THVector3 pos, Chunk chunk)
- static Chunk GetChunk (THVector3 vec3)
- static bool SetBlock (RaycastHit hit, Block block, bool adjacent=false)

  Sets the Block at the given point the given Block

Static Public Attributes

- static World world

Static Private Member Functions

- static float RoundXZ (float pos, float normal)

  Used to round the X/Z values when getting a block
- static float RoundY (float pos, float normal)

  Round the Y value of the given coord

6.49.1 Detailed Description

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

Definition at line 12 of file Terrain.cs.

6.49.2 Member Function Documentation

6.49.2.1 BlockInPosition()

```
static bool BeeGame.Terrain.LandGeneration.Terrain.BlockInPosition (
            THVector3 pos,
            Chunk chunk )  [static]
```

Definition at line 247 of file Terrain.cs.

```
        {
            if (chunk == null)
                return false;

            if (chunk.GetBlock((int)pos.x, (int)pos.y, (int)pos.z) != new Air())
                return true;

            return false;
        }
```

6.49.2.2 GetBlock() [1/2]

```
static Block BeeGame.Terrain.LandGeneration.Terrain.GetBlock (
            RaycastHit hit,
            bool adjacent = false )  [static]
```

Get a Block at the given position

Parameters

hit	Where to get the block from
adjacent	Should the adjacent Block be returned

Returns

Block at *hit.point* , Null if no block was found

Definition at line 221 of file Terrain.cs.

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

            if (chunk == null)
                return null;

            //* allignes the hit to the block grid and returns the block
            ChunkWorldPos pos = GetBlockPos(hit, adjacent);

            return chunk.world.GetBlock(pos.x, pos.y, pos.z);
        }
```

6.49.2.3 GetBlock() [2/2]

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

Definition at line 235 of file Terrain.cs.

```
        {
            Chunk chunk = GetChunk(pos);

            if (chunk == null)
                return new Air();

            chunk.world.GetBlock((int)pos.x, (int)pos.y, (int)pos.z);

            return new Block();
        }
```

6.49.2.4 GetBlockPos() [1/3]

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

Gets a block postion from a THVector3

Parameters

pos	Position of the block as a THVector3

Returns

ChunkWorldPos of the Block

Definition at line 22 of file Terrain.cs.

```
        {
            return new ChunkWorldPos()
            {
                x = Mathf.RoundToInt(pos.x),
                y = Mathf.RoundToInt(pos.y),
                z = Mathf.RoundToInt(pos.z)
            };
        }
```

6.49.2.5 GetBlockPos() [2/3]

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

Returns the positon of the block hit as a THVector3

Parameters

hit	RaycastHit
adjacent	Do you want the face adjecent to the block hit

Returns

THVector3 of the block you hit in world cordinates

Definition at line 38 of file Terrain.cs.

```
        {
            THVector3 vec3 = new THVector3()
            {
                x = RoundXZ(hit.point.x, hit.normal.x),
                y = RoundY(hit.point.y, hit.normal.y),
                z = RoundXZ(hit.point.z, hit.normal.z)
            };
            return (vec3);
        }
```

6.49.2.6 GetBlockPos() [3/3]

```
static ChunkWorldPos BeeGame.Terrain.LandGeneration.Terrain.GetBlockPos (
            RaycastHit hit,
            bool adjacent = false )  [static]
```

Gets a Chunks world positon

Parameters

hit	Where the raycast hit
adjacent	Should the adjacent Chunk position be returned?

Returns

ChunkWorldPos of the Chunk

Returns

Definition at line 204 of file Terrain.cs.

```
{
    return GetBlockPos(new THVector3()
    {
        //* rounds the hit to the correct position
        x = Round(hit.point.x, hit.normal.x, adjacent),
        y = Round(hit.point.y, hit.normal.y, adjacent),
        z = Round(hit.point.z, hit.normal.z, adjacent)
    });
}
```

6.49.2.7 GetBlockPosFromRayCast()

```
static ChunkWorldPos BeeGame.Terrain.LandGeneration.Terrain.GetBlockPosFromRayCast (
            RaycastHit hit )  [static]
```

GetBlockPos(THVector3) does the same thing but returns a ChunkWorldPos

Parameters

hit	

Returns

Definition at line 54 of file Terrain.cs.

```
{
    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));
}
```

6.49.2.8 GetChunk()

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

Definition at line 259 of file Terrain.cs.

```
{
    return world.GetChunk((int)vec3.x, (int)vec3.y, (int)vec3.
z);
}
```

6.49.2.9 Round()

```
static float BeeGame.Terrain.LandGeneration.Terrain.Round (
            float pos,
            float norm,
            bool adjacent = false )  [static]
```

Rounds the given pos to the correct position

Parameters

pos	Position that needs to be rounded
norm	Normal for the face
adjacent	Should the adjacent block be recived

Returns

rounded value of *pos* as a float

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

Definition at line 179 of file Terrain.cs.

```
        {
            if(pos - (int)pos == 0.5f || pos - (int)pos == -0.5f)
            {
                if(adjacent)
                {
                    pos += (norm / 2);
                }
                else
                {
                    pos -= (norm / 2);
                }
            }

            return pos;
        }
```

6.49.2.10 RoundXZ()

```
static float BeeGame.Terrain.LandGeneration.Terrain.RoundXZ (
            float pos,
            float normal )  [static], [private]
```

Used to round the X/Z values when getting a block

Parameters

pos	X/Y pos
normal	X/Y normal

Returns

rounded *pos*

Do I realy need to do all this?

Definition at line 68 of file Terrain.cs.

```
        {
            //* if we are looking at + x/z vlaues
            if (pos > 0)
            {
                if (normal > 0)
                {
                    pos = (int)pos;
                    return pos;
                }
                else if (normal < 0)
                {
                    pos = (int)pos;
                    return pos - -1;
                }
                else
                {
                    if ((pos - (int)pos) > 0.5)
                    {
                        return (int)pos + 1;
                    }
                    return (int)pos;
                }
            }
            //* if we are looking at - x/z values
            else
            {
                //* if poitive normal
                if (normal > 0)
                {
                    pos = (int)pos;
                    return pos - 1;
                }

                //* if negative nomrmal
                if (normal < 0)
                {
                    pos = (int)pos;
                    return pos;
                }
                //* if their is no normal

                //* if pos is greater than 0.5 we are in the next block so go to it
                if ((-pos - (int)-pos) > 0.5)
                {
                    return (int)pos - 1;
                }

                return (int)pos;
            }
        }
```

6.49.2.11 RoundY()

```
static float BeeGame.Terrain.LandGeneration.Terrain.RoundY (
            float pos,
            float normal )  [static], [private]
```

Round the Y value of the given coord

Parameters

pos	Y pos
normal	Y normal

Returns

pos rounded to 1 DP

Do I have to do this? or is their an easier way to do this

Definition at line 129 of file Terrain.cs.

```
        {
            pos = (float)Math.Round(pos, 1);
            if (pos >= 0)
            {
                if(normal > 0)
                {
                    if((int)pos % 2 == 0)
                        return Mathf.RoundToInt((float)Math.Round(pos, 1));

                    return Mathf.RoundToInt((float)Math.Round(pos, 1)) - normal;
                }

                if((int)pos % 2 == 0)
                    return Mathf.RoundToInt((float)Math.Round(pos, 1)) - normal;

                return Mathf.RoundToInt((float)Math.Round(pos, 1));
            }

            if(pos <= 0)
            {
                if (normal > 0)
                {
                    if ((int)pos % 2 == 0)
                        //* the Math.Round removes strange rounding errors shown with Mathf.Round eg
sometimes 0.5 would round to 0 not 1
                        return Mathf.RoundToInt((float)Math.Round(pos, 1)) - normal;

                    return Mathf.RoundToInt((float)Math.Round(pos, 1));// - normal;
                }

                if ((int)pos % 2 == 0)
                    return Mathf.RoundToInt((float)Math.Round(pos, 1));

                return Mathf.RoundToInt((float)Math.Round(pos, 1)) - normal;
            }


            return Mathf.RoundToInt((float)Math.Round(pos, 1));
        }
```

6.49.2.12 SetBlock()

```
static bool BeeGame.Terrain.LandGeneration.Terrain.SetBlock (
            RaycastHit hit,
            Block block,
            bool adjacent = false )  [static]
```

Sets the Block at the given point the given Block

Parameters

hit	Where the block should be set
block	Block to be set
adjacent	Should the adjacent Block be set

Returns

true if block was set

Definition at line 272 of file Terrain.cs.

```
        {
            //* checks that a chnk was hit
            Chunk chunk = hit.collider.GetComponent<Chunk>();

            if (chunk == null)
                return false;

            //* alligns the hit to the block grid
            ChunkWorldPos pos = GetBlockPosFromRayCast(hit);

            //* checks that the block tryign to be replaced can be replaced eg bedrock cannot be replaced
            if (GetBlock(hit, adjacent).breakable)
            {
                //* sets the position of the block and saves the chunk
                chunk.world.SetBlock(pos.x, pos.y, pos.z, block);
                Serialization.Serialization.SaveChunk(chunk);
            }

            return true;
        }
```

6.49.3 Member Data Documentation

6.49.3.1 world

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

Definition at line 14 of file Terrain.cs.

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

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/LandGeneration/Terrain.↩ cs

6.50 BeeGame.Terrain.LandGeneration.TerrainGeneration Class Reference

Generates the terrain for the game

Public Member Functions

- Chunk ChunkGen (Chunk chunk)

  Generates a Chunk in a new thread
- void ChunkGenThread (Chunk chunk, out Chunk outChunk)

  Generates a new Chunk
- Chunk GenChunkColum (Chunk chunk, int x, int z)

  Generates a colum of the Chunk

Static Public Member Functions

- static int GetNoise (int x, int y, int z, float scale, int max)

  Get a noise value
- static void SetBlock (int x, int y, int z, Blocks.Block block, Chunk chunk, bool replacesBlocks=false)

  Sets a Block in the position

Private Member Functions

- void CreateTree (int x, int y, int z, Chunk chunk)

  Makes a tree

Private Attributes

- float stoneBaseHeight = -24

  Base height of stone
- float stoneBaseNoise = 0.05f

  Base noise of stone
- float stoneBaseNoiseHeight = 4

  Base noise heigh for stone
- float stoneMountainHeight = 48

  Base height for a mountain
- float stoneMountainFrequency = 0.008f

  Frequency of mountains (larger value = more choppy terrain)
- float stoneMinHeight = -12

  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 treeFrequency = 0.2f

  Frequency of trees
- int treeDensity = 3

  Desity of trees
- float caveFrequency = 0.025f

  How often do caves happen
- int caveSize = 8

  Threashold for makeing a cave

6.50.1 Detailed Description

Generates the terrain for the game

Definition at line 13 of file TerrainGeneration.cs.

6.50.2 Member Function Documentation

6.50.2.1 ChunkGen()

```
Chunk BeeGame.Terrain.LandGeneration.TerrainGeneration.ChunkGen (
            Chunk chunk )
```

Generates a Chunk in a new thread

Parameters

chunk	Chunk to populate with Blocks

Returns

Chunk with Blocks generated

Definition at line 79 of file TerrainGeneration.cs.

```
        {
            Chunk outChunk = chunk;
            lock (chunk)
            {
                Thread thread = new Thread(() => ChunkGenThread(chunk, out outChunk)) { Name
 = $"Generate Chunk Thread @ {chunk.chunkWorldPos}"};

                thread.Start();
                return outChunk;
            }
        }
```

6.50.2.2 ChunkGenThread()

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

Generates a new Chunk

Parameters

chunk	Chunk to be generated
outChunk	Generated Chunk to return

Definition at line 96 of file TerrainGeneration.cs.

```
        {
            //* for each x and z position in teh chunk
            for (int x = chunk.chunkWorldPos.x-3; x < chunk.
 chunkWorldPos.x + Chunk.chunkSize + 3; x++)
            {
                for (int z = chunk.chunkWorldPos.z-3; z < chunk.
 chunkWorldPos.z + Chunk.chunkSize + 3; z++)
                {
                    chunk = GenChunkColum(chunk, x, z);
                }
            }

            chunk.SetBlocksUnmodified();
            outChunk = chunk;
        }
```

6.50.2.3 CreateTree()

```
void BeeGame.Terrain.LandGeneration.TerrainGeneration.CreateTree (
             int x,
             int y,
             int z,
             Chunk chunk )  [private]
```

Makes a tree

Parameters

x	X pos of the trunk
y	Y pos of the trunk
z	Z pos of the trunk
chunk	Chunk to make the tree in

Trees will always look the same, possibly add to leafs can have different shapes

Definition at line 210 of file TerrainGeneration.cs.

```
        {
            //* makes the leaves of teh tree
            for (int xi = -2; xi <= 2; xi++)
            {
                for (int yi = 4; yi <= 8; yi++)
                {
                    for (int zi = -2; zi <= 2; zi++)
                    {
                        SetBlock(xi + x, yi + y, zi + z, new Blocks.Leaves(), chunk, true);
                    }
                }
            }

            //* makes the trunk of the tree
            for (int i = 0; i < 6; i++)
            {
                SetBlock(x, y + i, z, new Blocks.Wood(), chunk, true);
            }
        }
```

6.50.2.4 GenChunkColum()

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

Generates a colum of the Chunk

Parameters

chunk	Chunk to generate a colum for
x	X pos to make the colum
z	Z pos to make the colum

Returns

Chunk with a new colum ob blocks generated

Definition at line 118 of file TerrainGeneration.cs.

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

            //* if the colum is currenly to low make it not so low
            if (stoneHeight < stoneMinHeight)
                stoneHeight = Mathf.FloorToInt(stoneMinHeight);

            //* add the height of normal stone on to the mountain
            stoneHeight += GetNoise(x, 0, -z, stoneBaseNoise, Mathf.RoundToInt(
stoneBaseNoiseHeight));

            //*put dirt on top
            int dirtHeight = stoneHeight + Mathf.FloorToInt(dirtBaseHeight);
            dirtHeight += GetNoise(x, 100, z, dirtNoise, Mathf.FloorToInt(
dirtNoiseHeight));

            //* set the colum to the correct blocks
            for (int y = chunk.chunkWorldPos.y - 8; y < chunk.
chunkWorldPos.y + Chunk.chunkSize; y ++)
            {
                int caveChance = GetNoise(x + 40, y + 100, z - 50,
caveFrequency, 200);

                //* puts a layer of bedrock at the botton the the world
                if (y <= (chunk.chunkWorldPos.y) && chunk.
chunkWorldPos.y == -16)
                {
                    SetBlock(x, y, z, new Blocks.Bedrock(), chunk);
                }
                else if (y <= stoneHeight && caveSize < caveChance)
                {
                    SetBlock(x, y, z, new Blocks.Block(), chunk);
                }
                else if (y <= dirtHeight && caveSize < caveChance)
                {
                    SetBlock(x, y, z, new Blocks.Grass(), chunk);
                    if (y == dirtHeight && GetNoise(x, 0, z,
treeFrequency, 100) < treeDensity)
                        CreateTree(x, y + 1, z, chunk);
                }
                else
                {
                    SetBlock(x, y, z, new Blocks.Air(), chunk);
                }
            }

                return chunk;
        }
```

6.50.2.5 GetNoise()

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

Get a noise value

Parameters

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

Returns

A noise value as an int

Definition at line 173 of file TerrainGeneration.cs.

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

6.50.2.6 SetBlock()

```
static void BeeGame.Terrain.LandGeneration.TerrainGeneration.SetBlock (
            int x,
            int y,
            int z,
            Blocks.Block block,
            Chunk chunk,
            bool replacesBlocks = false )  [static]
```

Sets a Block in the position

Parameters

x	X pos of the block
y	Y pos of the block
z	Z pos of the block
block	Block to set
chunk	Chunk to set the block in
replacesBlocks	Can it replace blocks

Definition at line 187 of file TerrainGeneration.cs.

```
        {
            //* corrects the x, y, z pos of the so that the block is placed in the correct position
            x -= chunk.chunkWorldPos.x;
            y -= chunk.chunkWorldPos.y;
            z -= chunk.chunkWorldPos.z;

            //* checks that the block is in the chunk and that no block is already their then sets it
            if (Chunk.InRange(x) && Chunk.InRange(y) &&
     Chunk.InRange(z))
                if (replacesBlocks || chunk.blocks[x, y, z] == null)
                    chunk.SetBlock(x, y, z, block, false);
        }
```

6.50.3 Member Data Documentation

6.50.3.1 caveFrequency

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

How often do caves happen

Definition at line 67 of file TerrainGeneration.cs.

6.50.3.2 caveSize

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

Threashold for makeing a cave

Definition at line 71 of file TerrainGeneration.cs.

6.50.3.3 dirtBaseHeight

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

Where does dirt start

Definition at line 45 of file TerrainGeneration.cs.

6.50.3.4 dirtNoise

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.dirtNoise = 0.04f  [private]
```

How much of the surface is dirt

Definition at line 49 of file TerrainGeneration.cs.

6.50.3.5 dirtNoiseHeight

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

How tall dirt can be

Definition at line 53 of file TerrainGeneration.cs.

6.50.3.6 stoneBaseHeight

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

Base height of stone

Definition at line 19 of file TerrainGeneration.cs.

6.50.3.7 stoneBaseNoise

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

Base noise of stone

Definition at line 23 of file TerrainGeneration.cs.

6.50.3.8 stoneBaseNoiseHeight

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

Base noise heigh for stone

Definition at line 27 of file TerrainGeneration.cs.

6.50.3.9 stoneMinHeight

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

Minimun height for stone

Definition at line 40 of file TerrainGeneration.cs.

6.50.3.10 stoneMountainFrequency

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.stoneMountainFrequency = 0.008f  [private]
```

Frequency of mountains (larger value = more choppy terrain)

Definition at line 36 of file TerrainGeneration.cs.

6.50.3.11 stoneMountainHeight

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

Base height for a mountain

Definition at line 32 of file TerrainGeneration.cs.

6.50.3.12 treeDensity

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

Desity of trees

Definition at line 62 of file TerrainGeneration.cs.

6.50.3.13 treeFrequency

```
float BeeGame.Terrain.LandGeneration.TerrainGeneration.treeFrequency = 0.2f  [private]
```

Frequency of trees

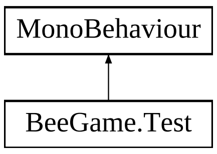
Definition at line 58 of file TerrainGeneration.cs.

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

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/LandGeneration/Terrain↵
Generation.cs

6.51 BeeGame.Test Class Reference

Inheritance diagram for BeeGame.Test:



Public Member Functions

- void Print (Item item)

Private Member Functions

- void Start ()

6.51.1 Detailed Description

Definition at line 14 of file test.cs.

6.51.2 Member Function Documentation

6.51.2.1 Print()

```
void BeeGame.Test.Print (
            Item item )
```

6.51.2.2 Start()

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

Definition at line 16 of file test.cs.

```
        {
            CraftingRecipies.AddShapedRecipie(new object[] { "   ", " X ", "   ", "X", Dirt.ID }, new Grass());
            CraftingRecipies.AddShaplessRecipie(new object[] { new Grass(), 1 }, new Dirt());

            Events.shapedRecipieCrafted += Print;
        }
```

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

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/test.cs

6.52 BeeGame.Core.THInput Class Reference

My implementation of the unity input system. Acts as a buffer layer to the unity system so that the input keys can be changed at runtime

Static Public Member Functions

- static bool GetButtonDown (string button)

  Has the given button been pressed this update
- static bool GetButton (string button)

  Is the given button currently being held down
- static bool GetButtonUp (string button)

  Has the given button been relesed this update
- static int GetAxis (string axis)

  Gets the axis of a button press

Static Public Attributes

- static bool isAnotherInventoryOpen

  If another inventory is open true, else false
- static bool blockInventoryJustClosed

  Was a Block inventory just closed

Static Package Attributes

- static bool chestOpen

  Stops the player from being able to open the BeeGame.Inventory.Player_Inventory.PlayerInventory whilst a block/item BeeGame.Inventory.Inventory is open

Static Private Attributes

- static Dictionary< string, object > inputButtons

  Button identifiers and KeyCode

6.52.1 Detailed Description

My implementation of the unity input system. Acts as a buffer layer to the unity system so that the input keys can be changed at runtime

Definition at line 11 of file THInput.cs.

6.52.2 Member Function Documentation

6.52.2.1 GetAxis()

```
static int BeeGame.Core.THInput.GetAxis (
              string axis )  [static]
```

Gets the axis of a button press

Parameters

axis	Axis to check, Horizontal or Vertical

Returns

+1 or -1

Definition at line 140 of file THInput.cs.

```
00141         {
```

```
            int returnAxis = 0;

            if (axis == "Horizontal")
            {
                if (GetButton("Right"))
                {
                    returnAxis += 1;
                }

                if (GetButton("Left"))
                {
                    returnAxis -= 1;
                }
            }
            else if (axis == "Vertical")
            {
                if (GetButton("Forward"))
                {
                    returnAxis += 1;
                }

                if (GetButton("Backward"))
                {
                    returnAxis -= 1;
                }
            }

            return returnAxis;
        }
```

6.52.2.2 GetButton()

```
static bool BeeGame.Core.THInput.GetButton (
              string button )  [static]
```

Is the given button currently being held down

Parameters

button	The button name eg "Forward"

Returns

true if the given button is currently being held down

Definition at line 80 of file THInput.cs.

```
        {
            if (!inputButtons.ContainsKey(button))
            {
                throw new InputException($"Key input name not defined: {button}");
            }

            switch (inputButtons[button])
            {
                case KeyCode[] arry:
                    //*for each posible key, check if it was pressed and if it was return that it was, if
      none of them was poressed return false
                    foreach (var item in arry)
                    {
                        if (Input.GetKey(item))
                        {
                            return true;
                        }
                    }

                    return false;
                default:
                    return Input.GetKey((KeyCode)inputButtons[button]);
            }
        }
```

6.52.2.3 GetButtonDown()

```
static bool BeeGame.Core.THInput.GetButtonDown (
             string button )  [static]
```

Has the given button been pressed this update

Parameters

button	The button name eg "Inventory"

Returns

true if the given button has been pressed this update

Definition at line 50 of file THInput.cs.

```
        {
            if (!inputButtons.ContainsKey(button))
            {
                throw new InputException($"Key input name not defined: {button}");
            }

            switch (inputButtons[button])
            {
                case KeyCode[] arry:
                    //*for each posible key, check if it was pressed and if it was return that it was, if none of them was poressed return false
                    foreach (var item in arry)
                    {
                        if (Input.GetKeyDown(item))
                        {
                            return true;
                        }
                    }

                    return false;
                default:
                    return Input.GetKeyDown((KeyCode)inputButtons[button]);
            }
        }
```

6.52.2.4 GetButtonUp()

```
static bool BeeGame.Core.THInput.GetButtonUp (
             string button )  [static]
```

Has the given button been relesed this update

Parameters

button	Button name eg "Inventory"

Returns

true if the button has been relesed during this update

Definition at line 110 of file THInput.cs.

```
        {
            if (!inputButtons.ContainsKey(button))
            {
                throw new InputException($"Key input name not defined: {button}");
            }

            switch (inputButtons[button])
            {
                case KeyCode[] arry:
                    //*for each posible key, check if it was pressed and if it was return that it was, if
none of them was poressed return false
                    foreach (var item in arry)
                    {
                        if (Input.GetKeyUp(item))
                        {
                            return true;
                        }
                    }

                    return false;
                default:
                    return Input.GetKeyUp((KeyCode)inputButtons[button]);
            }
        }
```

6.52.3 Member Data Documentation

6.52.3.1 blockInventoryJustClosed

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

Was a Block inventory just closed

Definition at line 39 of file THInput.cs.

6.52.3.2 chestOpen

`bool BeeGame.Core.THInput.chestOpen [static], [package]`

Stops the player from being able to open the BeeGame.Inventory.Player_Inventory.PlayerInventory whilst a block/item BeeGame.Inventory.Inventory is open

Definition at line 43 of file THInput.cs.

6.52.3.3 inputButtons

```
Dictionary<string, object> BeeGame.Core.THInput.inputButtons  [static], [private]
```

Initial value:

```
= new Dictionary<string, object>()
        {
            {"Forward" , KeyCode.W},
            {"Backward", KeyCode.S },
            {"Right", KeyCode.D },
            {"Left", KeyCode.A },
            {"Player Inventory", KeyCode.E },
            {"Quest Book", KeyCode.Mouse1 },
            {"Interact", KeyCode.Mouse1 },
            {"Place", KeyCode.Mouse1 },
            {"Break Block", KeyCode.Mouse0 },
            {"Close Menu/Inventory", new KeyCode[2] { KeyCode.Escape, KeyCode.E } },
            {"Jump", KeyCode.Space }
        }
```

Button identifiers and KeyCode

Definition at line 16 of file THInput.cs.

6.52.3.4 isAnotherInventoryOpen

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

If another inventory is open true, else false

Definition at line 34 of file THInput.cs.

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

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityTypeReplacements/T↩ HInput.cs

6.53 BeeGame.Core.UnityTypeReplacements.THQuaternion Struct Reference

Public Attributes

- float x
- float y
- float z
- float w

6.53.1 Detailed Description

Definition at line 8 of file THQuaternion.cs.

6.53.2 Member Data Documentation

6.53.2.1 w

```
float BeeGame.Core.UnityTypeReplacements.THQuaternion.w
```

Definition at line 13 of file THQuaternion.cs.

6.53.2.2 x

```
float BeeGame.Core.UnityTypeReplacements.THQuaternion.x
```

Definition at line 10 of file THQuaternion.cs.

6.53.2.3 y

```
float BeeGame.Core.UnityTypeReplacements.THQuaternion.y
```

Definition at line 11 of file THQuaternion.cs.

6.53.2.4 z

```
float BeeGame.Core.UnityTypeReplacements.THQuaternion.z
```

Definition at line 12 of file THQuaternion.cs.

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

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityTypeReplacements/T↩ HQuaternion.cs

6.54 BeeGame.Core.THVector2 Struct Reference

Serilializable version of Vector2

Public Member Functions

- THVector2 (float x, float y)

  Constructor from 2 floats
- THVector2 (THVector2 vec2)

  Constructor from another THVector2
- THVector2 (Vector2 vec2)

  Constructor from Vector2
- override bool Equals (object obj)
- override int GetHashCode ()
- override string ToString ()

Static Public Member Functions

- static bool operator== (THVector2 a, THVector2 b)
- static bool operator!= (THVector2 a, THVector2 b)
- static THVector2 operator+ (THVector2 a, THVector2 b)
- static THVector2 operator+ (THVector2 a, float b)
- static THVector2 operator+ (float a, THVector2 b)
- static THVector2 operator- (THVector2 a, THVector2 b)
- static THVector2 operator- (THVector2 a, float b)
- static THVector2 operator- (float a, THVector2 b)
- static THVector2 operator∗ (THVector2 a, THVector2 b)
- static THVector2 operator∗ (THVector2 a, float b)
- static THVector2 operator∗ (float a, THVector2 b)
- static THVector2 operator/ (THVector2 a, THVector2 b)
- static THVector2 operator/ (THVector2 a, float b)
- static THVector2 operator/ (float a, THVector2 b)
- static implicit operator Vector2 (THVector2 vec2)
- static implicit operator THVector2 (Vector2 vec2)

Public Attributes

- float x

  X position
- float y

  Y position

6.54.1 Detailed Description

Serilializable version of Vector2

Definition at line 10 of file THVector2.cs.

6.54.2 Constructor & Destructor Documentation

6.54.2.1 THVector2() [1/3]

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

Constructor from 2 floats

Parameters

x	X position
y	Y position

Definition at line 29 of file THVector2.cs.

```
        {
            this.x = x;
            this.y = y;
        }
```

6.54.2.2 THVector2() [2/3]

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

Constructor from another THVector2

Parameters

vec2	Vector to make this from

Definition at line 39 of file THVector2.cs.

```
        {
            this = vec2;
        }
```

6.54.2.3 THVector2() [3/3]

```
BeeGame.Core.THVector2.THVector2 (
            Vector2 vec2 )
```

Constructor from Vector2

Parameters

vec2	Vector to make this from

Definition at line 48 of file THVector2.cs.

```
        {
            this = vec2;
        }
```

6.54.3 Member Function Documentation

6.54.3.1 Equals()

```
override bool BeeGame.Core.THVector2.Equals (
              object obj )
```

Definition at line 55 of file THVector2.cs.

```
        {
            if (!(obj is THVector2))
                return false;
            if (obj.GetHashCode() == GetHashCode())
                return true;
            return false;
        }
```

6.54.3.2 GetHashCode()

```
override int BeeGame.Core.THVector2.GetHashCode ( )
```

Definition at line 64 of file THVector2.cs.

```
        {
            unchecked
            {
                int hash = 13;

                hash *= 443 * x.GetHashCode();
                hash *= 373 * y.GetHashCode();

                return hash;
            }
        }
```

6.54.3.3 operator THVector2()

```
static implicit BeeGame.Core.THVector2.operator THVector2 (
              Vector2 vec2 )  [static]
```

Definition at line 171 of file THVector2.cs.

```
        {
            return new THVector2(vec2.x, vec2.y);
        }
```

6.54.3.4 operator Vector2()

```
static implicit BeeGame.Core.THVector2.operator Vector2 (
              THVector2 vec2 )  [static]
```

Definition at line 166 of file THVector2.cs.

```
        {
            return new Vector2(vec2.x, vec2.y);
        }
```

6.54.3.5 operator"!=()

```
static bool BeeGame.Core.THVector2.operator!= (
            THVector2 a,
            THVector2 b )  [static]
```

Definition at line 86 of file THVector2.cs.

```
        {
            return !(a == b);
        }
```

6.54.3.6 operator*() [1/3]

```
static THVector2 BeeGame.Core.THVector2.operator* (
            THVector2 a,
            THVector2 b )  [static]
```

Definition at line 127 of file THVector2.cs.

```
        {
            a.x *= b.x;
            a.y *= b.y;

            return a;
        }
```

6.54.3.7 operator*() [2/3]

```
static THVector2 BeeGame.Core.THVector2.operator* (
            THVector2 a,
            float b )  [static]
```

Definition at line 134 of file THVector2.cs.

```
        {
            a.x *= b;
            a.y *= b;

            return a;
        }
```

6.54.3.8 operator*() [3/3]

```
static THVector2 BeeGame.Core.THVector2.operator* (
            float a,
            THVector2 b )  [static]
```

Definition at line 141 of file THVector2.cs.

```
        {
            return new THVector2(a * b.x, a * b.y);
        }
```

6.54.3.9 operator+() [1/3]

```
static THVector2 BeeGame.Core.THVector2.operator+ (
            THVector2 a,
            THVector2 b ) [static]
```

Definition at line 91 of file THVector2.cs.

```
        {
            a.x += b.x;
            a.y += b.y;

            return a;
        }
```

6.54.3.10 operator+() [2/3]

```
static THVector2 BeeGame.Core.THVector2.operator+ (
            THVector2 a,
            float b ) [static]
```

Definition at line 98 of file THVector2.cs.

```
        {
            a.x += b;
            a.y += b;

            return a;
        }
```

6.54.3.11 operator+() [3/3]

```
static THVector2 BeeGame.Core.THVector2.operator+ (
            float a,
            THVector2 b ) [static]
```

Definition at line 105 of file THVector2.cs.

```
        {
            return new THVector2(a + b.x, a + b.y);
        }
```

6.54.3.12 operator-() [1/3]

```
static THVector2 BeeGame.Core.THVector2.operator- (
            THVector2 a,
            THVector2 b ) [static]
```

Definition at line 109 of file THVector2.cs.

```
        {
            a.x -= b.x;
            a.y -= b.y;

            return a;
        }
```

6.54.3.13 operator-() [2/3]

```
static THVector2 BeeGame.Core.THVector2.operator- (
            THVector2 a,
            float b )  [static]
```

Definition at line 116 of file THVector2.cs.

```
        {
            a.x += b;
            a.y += b;

            return a;
        }
```

6.54.3.14 operator-() [3/3]

```
static THVector2 BeeGame.Core.THVector2.operator- (
            float a,
            THVector2 b )  [static]
```

Definition at line 123 of file THVector2.cs.

```
        {
            return new THVector2(a - b.x, a - b.y);
        }
```

6.54.3.15 operator/() [1/3]

```
static THVector2 BeeGame.Core.THVector2.operator/ (
            THVector2 a,
            THVector2 b )  [static]
```

Definition at line 145 of file THVector2.cs.

```
        {
            a.x /= b.x;
            a.y /= b.y;

            return a;
        }
```

6.54.3.16 operator/() [2/3]

```
static THVector2 BeeGame.Core.THVector2.operator/ (
            THVector2 a,
            float b )  [static]
```

Definition at line 152 of file THVector2.cs.

```
        {
            a.x /= b;
            a.y /= b;

            return a;
        }
```

6.54.3.17 operator/() [3/3]

```
static THVector2 BeeGame.Core.THVector2.operator/ (
             float a,
             THVector2 b ) [static]
```

Definition at line 159 of file THVector2.cs.

```
        {
            return new THVector2(a / b.x, a / b.y);
        }
```

6.54.3.18 operator==()

```
static bool BeeGame.Core.THVector2.operator== (
             THVector2 a,
             THVector2 b ) [static]
```

Definition at line 82 of file THVector2.cs.

```
        {
            return a.Equals(b);
        }
```

6.54.3.19 ToString()

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

Definition at line 77 of file THVector2.cs.

```
        {
            return $"{x}, {y}";
        }
```

6.54.4 Member Data Documentation

6.54.4.1 x

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

X position

Definition at line 16 of file THVector2.cs.

6.54.4.2 y

```
float BeeGame.Core.THVector2.y
```

Y position

Definition at line 20 of file THVector2.cs.

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

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityTypeReplacements/T↩ HVector2.cs

6.55 BeeGame.Core.THVector3 Struct Reference

Serializable version of Vector3

Public Member Functions

- THVector3 (float x, float y, float z)

  Constructor from 3 floats
- THVector3 (THVector3 vec3)

  Constructor from another THVector3
- THVector3 (Vector3 vec3)

  Constructor from another Vector3
- THVector3 (Terrain.ChunkWorldPos vec3)

  Constructor from another Terrain.ChunkWorldPos
- override bool Equals (object obj)

  This this vector == to another
- override int GetHashCode ()

  Gets the hascode for the vector
- override string ToString ()

  Formats the vector as a nice string

Static Public Member Functions

- static float Distance (THVector3 a, THVector3 b)

  Distance between 2 vectors
- static bool operator== (THVector3 a, THVector3 b)

  Checks if a == b
- static bool operator!= (THVector3 a, THVector3 b)

  Inverse of ==
- static THVector3 operator+ (THVector3 a, THVector3 b)

  Adds vector a and b
- static THVector3 operator+ (THVector3 a, float b)

  Adds b to vector a
- static THVector3 operator+ (float a, THVector3 b)

  Adds a to vector b
- static THVector3 operator- (THVector3 a, THVector3 b)

Subtracs 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)

  Multiples b to vector a
- static THVector3 operator∗ (float a, THVector3 b)

  Multiples a to vector b
- static THVector3 operator/ (THVector3 a, THVector3 b)

  Divides vector a 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 implicetly
- static implicit operator THVector3 (Vector3 vec3)

  Converts Vector3 to THVector3 implicetly
- static operator Quaternion (THVector3 vec3)

  Converts a THVector3 to a Quaternion

Public Attributes

- float x

  X position
- float y

  Y postion
- float z

  Z position

6.55.1 Detailed Description

Serializable version of Vector3

Definition at line 10 of file THVector3.cs.

6.55.2 Constructor & Destructor Documentation

6.55.2.1 THVector3() [1/4]

```
BeeGame.Core.THVector3.THVector3 (
            float x,
            float y,
            float z )
```

Constructor from 3 floats

Parameters

x	X position
y	Y position
z	Z position

Definition at line 34 of file THVector3.cs.

```
        {
            this.x = x;
            this.y = y;
            this.z = z;
        }
```

6.55.2.2 THVector3() [2/4]

```
BeeGame.Core.THVector3.THVector3 (
            THVector3 vec3 )
```

Constructor from another THVector3

Parameters

vec3	Vector to make this from

Definition at line 45 of file THVector3.cs.

```
        {
            this = vec3;
        }
```

6.55.2.3 THVector3() [3/4]

```
BeeGame.Core.THVector3.THVector3 (
            Vector3 vec3 )
```

Constructor from another Vector3

Parameters

vec3	Vector to make this from

Definition at line 54 of file THVector3.cs.

```
        {
            this = vec3;
        }
```

6.55.2.4 THVector3() [4/4]

```
BeeGame.Core.THVector3.THVector3 (
            Terrain.ChunkWorldPos vec3 )
```

Constructor from another Terrain.ChunkWorldPos

Parameters

vec3	Vector to make this from

Definition at line 63 of file THVector3.cs.

```
        {
            this = vec3;
        }
```

6.55.3 Member Function Documentation

6.55.3.1 Distance()

```
static float BeeGame.Core.THVector3.Distance (
            THVector3 a,
            THVector3 b )  [static]
```

Distance between 2 vectors

Parameters

a	First Vector
b	Second Vector

Returns

Distance between *a* and *b*

Definition at line 76 of file THVector3.cs.

```
        {
            return (float)Math.Sqrt(Math.Pow((a.x - b.x), 2) + Math.Pow((a.y - b.y), 2) + Math.Pow((a.z - b
    .z), 2));
        }
```

6.55.3.2 Equals()

```
override bool BeeGame.Core.THVector3.Equals (
            object obj )
```

This this vector == to another

Parameters

obj	object to check against

Returns

Definition at line 88 of file THVector3.cs.

```
        {
            if (!(obj is THVector3))
                return false;
            if (obj.GetHashCode() == GetHashCode())
                return true;
            return false;
        }
```

6.55.3.3 GetHashCode()

```
override int BeeGame.Core.THVector3.GetHashCode ( )
```

Gets the hascode for the vector

Returns

Definition at line 101 of file THVector3.cs.

```
        {
            unchecked
            {
                int hash = 13;

                hash *= 443 * x.GetHashCode();
                hash *= 373 * y.GetHashCode();
                hash *= 127 * z.GetHashCode();

                return hash;
            }
        }
```

6.55.3.4 operator Quaternion()

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

Converts a THVector3 to a Quaternion

Parameters

vec3	Vector to convert to Quaternion

Explicit as conversion is not exact

Definition at line 327 of file THVector3.cs.

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

6.55.3.5 operator THVector3()

```
static implicit BeeGame.Core.THVector3.operator THVector3 (
            Vector3 vec3 ) [static]
```

Converts Vector3 to THVector3 implicetly

Parameters

vec3	Vector to convert

Definition at line 313 of file THVector3.cs.

```
        {
            return new THVector3(vec3.x, vec3.y, vec3.z);
        }
```

6.55.3.6 operator Vector3()

```
static implicit BeeGame.Core.THVector3.operator Vector3 (
            THVector3 vec3 ) [static]
```

Converts THVector3 to Vector3 implicetly

Parameters

vec3	Vector to convert

Definition at line 304 of file THVector3.cs.

```
        {
            return new Vector3(vec3.x, vec3.y, vec3.z);
        }
```

6.55.3.7 operator"!=()

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

Inverse of ==

Parameters

a	First vector
b	Second vector

Returns

true if *a* != *b*

Definition at line 140 of file THVector3.cs.

```
        {
            return !(a == b);
        }
```

6.55.3.8 operator*() [1/3]

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

Multiplies vector a and b

Parameters

a	Vector a
b	Vector b

Returns

returns new vector that is the product of a and b

Definition at line 227 of file THVector3.cs.

```
        {
            a.x *= b.x;
            a.y *= b.y;
            a.z *= b.z;

            return a;
        }
```

6.55.3.9 operator*() [2/3]

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

Multiples b to vector a

Parameters

a	Vector a
b	float b

Returns

returns new vector that is the product of a and b

Definition at line 241 of file THVector3.cs.

```
        {
            a.x *= b;
            a.y *= b;
            a.z *= b;

            return a;
        }
```

6.55.3.10 operator*() [3/3]

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

Multiples a to vector b

Parameters

a	Vector a
b	float b

Returns

returns new vector that is the product of a and b

Definition at line 255 of file THVector3.cs.

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

6.55.3.11 operator+() [1/3]

```
static THVector3 BeeGame.Core.THVector3.operator+ (
            THVector3 a,
            THVector3 b ) [static]
```

Adds vector a and b

Parameters

a	Vector a
b	Vector b

Returns

returns new vector that is the sum of a and b

Definition at line 151 of file THVector3.cs.

```
        {
            a.x += b.x;
            a.y += b.y;
            a.z += b.z;

            return a;
        }
```

6.55.3.12 operator+() [2/3]

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

Adds b to vector a

Parameters

a	Vector a
b	float b

Returns

returns new vector that is the sum of a and b

Definition at line 165 of file THVector3.cs.

```
        {
            a.x += b;
            a.y += b;
            a.z += b;

            return a;
        }
```

6.55.3.13 operator+() [3/3]

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

Adds a to vector b

Parameters

a	Vector a
b	float b

Returns

returns new vector that is the sum of a and b

Definition at line 179 of file THVector3.cs.

```
        {
            return new THVector3(a + b.x, a + b.y, a + b.z);
        }
```

6.55.3.14 operator-() [1/3]

```
static THVector3 BeeGame.Core.THVector3.operator- (
            THVector3 a,
            THVector3 b ) [static]
```

Subtracs vector a and b

Parameters

a	Vector a
b	Vector b

Returns

returns new vector that is the subtraction of a and b

Definition at line 189 of file THVector3.cs.

```
        {
            a.x -= b.x;
            a.y -= b.y;
            a.z -= b.z;

            return a;
        }
```

6.55.3.15 operator-() [2/3]

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

Subtracts b from vector a

Parameters

a	Vector a
b	float b

Returns

returns new vector that is the subtraction of a and b

Definition at line 203 of file THVector3.cs.

```
        {
            a.x += b;
            a.y += b;
            a.z += b;

            return a;
        }
```

6.55.3.16 operator-() [3/3]

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

Subtracts a from vector b

Parameters

a	Vector a
b	float b

Returns

returns new vector that is the subtraction of a and b

Definition at line 217 of file THVector3.cs.

```
        {
            return new THVector3(a - b.x, a - b.y, a - b.z);
        }
```

6.55.3.17 operator/() [1/3]

```
static THVector3 BeeGame.Core.THVector3.operator/ (
            THVector3 a,
            THVector3 b )  [static]
```

Divides vector a and b

Parameters

a	Vector a
b	Vector b

Returns

returns new vector that is the division of a and b

Definition at line 265 of file THVector3.cs.

```
        {
            a.x /= b.x;
            a.y /= b.y;
            a.z /= b.z;

            return a;
        }
```

6.55.3.18 operator/() [2/3]

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

Divides a by b

Parameters

a	Vector a
b	float b

Returns

returns new vector that is the division of a and b

Definition at line 279 of file THVector3.cs.

```
        {
            a.x /= b;
            a.y /= b;
            a.z /= b;

            return a;
        }
```

6.55.3.19 operator/() [3/3]

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

Divides b by a

Parameters

a	Vector a
b	float b

Returns

returns new vector that is the division of a and b

Definition at line 293 of file THVector3.cs.

```
        {
            return new THVector3(a / b.x, a / b.y, a / b.z);
        }
```

6.55.3.20 operator==()

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

Checks if *a* == *b*

Parameters

a	First vector
b	Second vector

Returns

true if *a* == *b*

Definition at line 130 of file THVector3.cs.

```
        {
            return a.Equals(b);
        }
```

6.55.3.21 ToString()

```
override string BeeGame.Core.THVector3.ToString ( )
```

Formats the vector as a nice string

Returns

The vector as a nice string

Definition at line 119 of file THVector3.cs.

```
        {
            return $"{x}, {y}, {z}";
        }
```

6.55.4 Member Data Documentation

6.55.4.1 x

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

X position

Definition at line 16 of file THVector3.cs.

6.55.4.2 y

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

Y postion

Definition at line 20 of file THVector3.cs.

6.55.4.3 z

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

Z position

Definition at line 24 of file THVector3.cs.

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

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityTypeReplacements/T↩
HVector3.cs

6.56 BeeGame.Items.Tile Struct Reference

Position of the items texture

Public Attributes

- int x

  X pos of the texture
- int y

  Y pos of the texture

6.56.1 Detailed Description

Position of the items texture

Definition at line 411 of file Item.cs.

6.56.2 Member Data Documentation

6.56.2.1 x

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

X pos of the texture

Definition at line 416 of file Item.cs.

6.56.2.2 y

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

Y pos of the texture

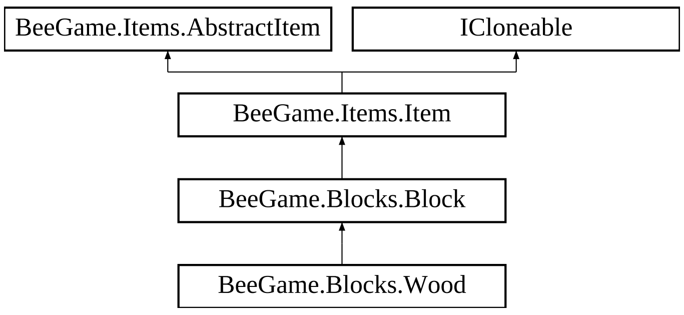
Definition at line 420 of file Item.cs.

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

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

6.57 BeeGame.Blocks.Wood Class Reference

Inheritance diagram for BeeGame.Blocks.Wood:



Public Member Functions

- Wood ()
- override Sprite GetItemSprite ()

    Returns the sprite for the item
- override Tile TexturePosition (Direction direction)

    Texture postion of the items texture
- override int GetHashCode ()

    Base ID of the block
- override string ToString ()

    Returns the name and ID of the block as a string

Static Public Attributes

- static new int ID => 5

Additional Inherited Members

6.57.1 Detailed Description

Definition at line 13 of file Wood.cs.

6.57.2 Constructor & Destructor Documentation

6.57.2.1 Wood()

```
BeeGame.Blocks.Wood.Wood ( )
```

Definition at line 17 of file Wood.cs.

```
                            : base("Wood")
          {
          }
```

6.57.3 Member Function Documentation

6.57.3.1 GetHashCode()

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

Base ID of the block

Returns

5

Reimplemented from BeeGame.Blocks.Block.

Definition at line 39 of file Wood.cs.

```
        {
            return ID;
        }
```

6.57.3.2 GetItemSprite()

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

Returns the sprite for the item

Returns

Sprite for this item

Reimplemented from BeeGame.Blocks.Block.

Definition at line 23 of file Wood.cs.

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

6.57.3.3 TexturePosition()

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

Texture postion of the items texture

Parameters

direction	Direction for the texture

Returns

Position of the texture

Reimplemented from BeeGame.Items.Item.

Definition at line 29 of file Wood.cs.

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

6.57.3.4 ToString()

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

Returns the name and ID of the block as a string

Returns

A nicely formatted string

Definition at line 48 of file Wood.cs.

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

6.57.4 Member Data Documentation

6.57.4.1 ID

```
new int BeeGame.Blocks.Wood.ID => 5  [static]
```

Definition at line 15 of file Wood.cs.

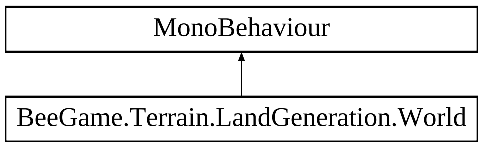
The documentation for this class was generated from the following file:

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

6.58 BeeGame.Terrain.LandGeneration.World Class Reference

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

Inheritance diagram for BeeGame.Terrain.LandGeneration.World:



Public Member Functions

- void CreateChunk (int x, int y, int z)

  Creates a chunk at the given x, y, z
- void DestroyChunk (int x, int y, int z)

  Destroys a Chunk st the given x, y, z postion
- void SetBlock (int x, int y, int z, Block block, bool saveChunk=false)

  Sets a Block at the given position
- Chunk GetChunk (int x, int y, int z)

  Gets a chunk at eh given x, y, z
- Block GetBlock (int x, int y, int z)

  Gets a Block at the given position

Public Attributes

- Dictionary< ChunkWorldPos, Chunk > chunks = new Dictionary<ChunkWorldPos, Chunk>()

  All of the currently loaded chunks
- GameObject chunkPrefab

  The chunk prefab
- bool chunkHasMadeCollisionMesh = false

  Has a Chunk made a collision mesh?

Private Member Functions

- void UpdateIfEqual (int value1, int value2, ChunkWorldPos pos)

  Updates a chunk if value1 and value2 are equal

6.58.1 Detailed Description

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

Definition at line 14 of file World.cs.

6.58.2 Member Function Documentation

6.58.2.1 CreateChunk()

```
void BeeGame.Terrain.LandGeneration.World.CreateChunk (
                int x,
                int y,
                int z )
```

Creates a chunk at the given x, y, z

Parameters

x	X pos to make the new chunk
y	Y pos to make the new chunk
z	Z pos to make the new chunk

Definition at line 41 of file World.cs.

```
        {
            //* pos of the chunk
            ChunkWorldPos pos = new ChunkWorldPos(x, y, z);

            //* makes the chunk at the given position
            GameObject newChunk = Instantiate(chunkPrefab, new Vector3(x, y, z), Quaternion.
    identity);

            Chunk chunk = newChunk.GetComponent<Chunk>();

            //* setting the chunks pos and a reference to this
            chunk.chunkWorldPos = pos;
            chunk.world = this;

            //* adds the nwe chunk to the dictionary
            chunks.Add(pos, chunk);

            //* generates the new chunks blocks
            chunk = new TerrainGeneration().ChunkGen(chunk);

            //loads any blocks that the chunk has had modified
            Serialization.Serialization.LoadChunk(chunk);

            //* updates all chunks around this one to reduce drawing of unecisary faces
            chunks.TryGetValue(new ChunkWorldPos(x, y - 16, z), out chunk);
            if (chunk != null)
                chunk.update = true;

            chunks.TryGetValue(new ChunkWorldPos(x, y, z - 16), out chunk);
            if (chunk != null)
                chunk.update = true;

            chunks.TryGetValue(new ChunkWorldPos(x - 16, y, z), out chunk);
            if (chunk != null)
                chunk.update = true;

            chunks.TryGetValue(new ChunkWorldPos(x, y + 16, z), out chunk);
            if (chunk != null)
                chunk.update = true;

            chunks.TryGetValue(new ChunkWorldPos(x, y, z + 16), out chunk);
            if (chunk != null)
                chunk.update = true;

            chunks.TryGetValue(new ChunkWorldPos(x + 16, y, z), out chunk);
            if (chunk != null)
                chunk.update = true;
            //* the chunk will then make its meshes
        }
```

6.58.2.2 DestroyChunk()

```
void BeeGame.Terrain.LandGeneration.World.DestroyChunk (
            int x,
            int y,
            int z )
```

Destroys a Chunk st the given x, y, z postion

Parameters

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

Definition at line 97 of file World.cs.

```
        {
            //* if teh chnks exists destroy it
            if (chunks.TryGetValue(new ChunkWorldPos(x, y, z), out Chunk chunk))
            {
                //* saves the chunk before destroying it incase any block were changed in it
                Serialization.Serialization.SaveChunk(chunk);
                Destroy(chunk.gameObject);
                chunks.Remove(new ChunkWorldPos(x, y, z));
            }
        }
```

6.58.2.3 GetBlock()

```
Block BeeGame.Terrain.LandGeneration.World.GetBlock (
                int x,
                int y,
                int z )
```

Gets a Block at the given position

Parameters

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

Returns

Block at given x, y, z position

Definition at line 184 of file World.cs.

```
        {
            //* gets the chunk that the block is in
            Chunk chunk = GetChunk(x, y, z);

            if(chunk != null)
            {
                //* gets the block in the chunk
                return chunk.GetBlock(x - chunk.chunkWorldPos.
      x, y - chunk.chunkWorldPos.y, z - chunk.chunkWorldPos.
      z) ?? new Air();
            }

            //* returns an empty block is the chunk was not found
            return new Air();
        }
```

6.58.2.4 GetChunk()

```
Chunk BeeGame.Terrain.LandGeneration.World.GetChunk (
            int x,
            int y,
            int z )
```

Gets a chunk at eh given x, y, z

Parameters

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.

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

            //* gets the chunk if it exists
            chunks.TryGetValue(pos, out Chunk chunk);
            //* if the chunk does not exist will return null
            return chunk;
        }
```

6.58.2.5 SetBlock()

```
void BeeGame.Terrain.LandGeneration.World.SetBlock (
            int x,
            int y,
            int z,
            Block block,
            bool saveChunk = false )
```

Sets a Block at the given position

Parameters

x	X pos of the block
y	Y pos of the block
z	Z pos of the block
block	Block to be placed

Definition at line 118 of file World.cs.

```
        {
            //*gets the chunk for the block to be placed in
            Chunk chunk = GetChunk(x, y, z);

            //*if the chunk is not null and the block trying to be replaced is replaceable, replace it
            if(chunk != null && chunk.blocks[x - chunk.chunkWorldPos.
      x, y - chunk.chunkWorldPos.y, z - chunk.chunkWorldPos.
      z].breakable)
            {

                chunk.SetBlock(x - chunk.chunkWorldPos.x, y - chunk.
      chunkWorldPos.y, z - chunk.chunkWorldPos.z, block);
                chunk.update = true;

                //*updates the nebouring chunks as when a block is broken it may be in the edje of the
       chunk so their meshes also need to be updated
                //*only updates chunks that need to be updated as not every chunk will need to be and
       sometines none of them will need to be

                //*checks if the block chaged is in the edge if the x value for the chunk
                UpdateIfEqual(x - chunk.chunkWorldPos.
      x, 0, new ChunkWorldPos(x - 1, y, z));
                UpdateIfEqual(x - chunk.chunkWorldPos.
      x, Chunk.chunkSize - 1, new ChunkWorldPos(x + 1, y, z));

                //*checks if the block chaged is in the edge if the y value for the chunk
                UpdateIfEqual(y - chunk.chunkWorldPos.
      y, 0, new ChunkWorldPos(x, y - 1, z));
                UpdateIfEqual(y - chunk.chunkWorldPos.
      y, Chunk.chunkSize - 1, new ChunkWorldPos(x, y + 1, z));

                //*checks if the block chaged is in the edge if the z value for the chunk
                UpdateIfEqual(z - chunk.chunkWorldPos.
      z, 0, new ChunkWorldPos(x, y, z - 1));
                UpdateIfEqual(z - chunk.chunkWorldPos.
      z, Chunk.chunkSize - 1, new ChunkWorldPos(x, y, z + 1));

                if (saveChunk)
                    Serialization.Serialization.SaveChunk(chunk);
            }
        }
```

6.58.2.6 UpdateIfEqual()

```
void BeeGame.Terrain.LandGeneration.World.UpdateIfEqual (
            int value1,
            int value2,
            ChunkWorldPos pos ) [private]
```

Updates a chunk if *value1* and *value2* are equal

Parameters

value1	First value to check
value2	Second value to check
pos	Position of chunk to update if values are equal

Definition at line 206 of file World.cs.

```
        {
            if(value1 == value2)
            {
                Chunk chunk = GetChunk(pos.x, pos.y, pos.z);

                if (chunk != null)
                    chunk.update = true;
            }
        }
```

6.58.3 Member Data Documentation

6.58.3.1 chunkHasMadeCollisionMesh

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

Has a Chunk made a collision mesh?

Definition at line 30 of file World.cs.

6.58.3.2 chunkPrefab

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

The chunk prefab

Definition at line 25 of file World.cs.

6.58.3.3 chunks

```
Dictionary<ChunkWorldPos, Chunk> BeeGame.Terrain.LandGeneration.World.chunks = new Dictionary<Chunk↵
WorldPos, Chunk>()
```

All of the currently loaded chunks

Definition at line 20 of file World.cs.

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

- C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/LandGeneration/World.↵ cs

7 File Documentation

7.1 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Air.cs File Reference

Classes

- class BeeGame.Blocks.Air

  Air Block is an empty block that does not render and has no collider

Namespaces

- namespace BeeGame.Blocks

7.2 Air.cs

```
using System;
using BeeGame.Core.Enums;
using BeeGame.Terrain.Chunks;
using BeeGame.Core;

namespace BeeGame.Blocks
{
    [Serializable]
    public class Air : Block
    {
        public new static int ID => 0;

        public Air() : base("Air")
        {
        }

        public override void BreakBlock(THVector3 pos)
        {
            return;
        }

        public override MeshData BlockData(Chunk chunk, int x, int y, int z,
 MeshData meshData, bool addRoRenderMesh = true)
        {
            return meshData;
        }

        public override bool IsSolid(Direction direction)
        {
            return false;
        }

        public override int GetHashCode()
        {
            return ID;
        }

        public override string ToString()
        {
            return $"{itemName} \nID: {GetItemID()}";
        }
    }
}
```

7.3 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Apiary.cs File Reference

Classes

- class BeeGame.Blocks.Apiary

    Apiary Block

Namespaces

- namespace BeeGame.Blocks

7.4 Apiary.cs

```
using System;
using System.Linq;
using UnityEngine;
using BeeGame.Core;
using BeeGame.Items;
using BeeGame.Inventory;
using BeeGame.Core.Enums;
using BeeGame.Terrain.Chunks;
using BeeGame.Core.Dictionarys;

```

```
namespace BeeGame.Blocks
{
    [Serializable]
    public class Apiary : Block
    {
        [NonSerialized]
        private GameObject myGameobject;

        public int mutationMultiplyer;

        public new static int ID => 10;

        #region Constructor
        public Apiary() : base("Apiary")
        {
            usesGameObject = true;
        }
        #endregion

        #region Block Overrides
        public override GameObject GetGameObject()
        {
            return PrefabDictionary.GetPrefab("Apiary");
        }

        public override Tile TexturePosition(Direction direction)
        {
            return new Tile() { x = 0, y = 9 };
        }

        public override MeshData BlockData(Chunk chunk, int x, int y, int z, MeshData meshData, bool addToRenderMesh = true)
        {
            if (myGameobject == null)
            {
                myGameobject = UnityEngine.Object.Instantiate(PrefabDictionary.GetPrefab("Apiary"), new THVector3(x, y, z) + chunk.chunkWorldPos, Quaternion.identity, chunk.transform);
                myGameobject.GetComponent<ChestInventory>().inventoryPosition = new THVector3(x, y, z) + chunk.chunkWorldPos;
                myGameobject.GetComponent<ChestInventory>().SetChestInventory();
            }
            return base.BlockData(chunk, x, y, z, meshData, true);
        }

        public override void BreakBlock(THVector3 pos)
        {
            //* removes the blocks blocks inventory save file and destroys the game object
            Serialization.Serialization.DeleteFile(myGameobject.GetComponent<ApiaryInventory>().inventoryName);
            UnityEngine.Object.Destroy(myGameobject);
            //* removes the collision mesh from the chunk
            base.BreakBlock(pos);
        }
        #endregion

        #region Overrides
        public override int GetHashCode()
        {
            return ID;
        }

        public override string ToString()
        {
            return $"{itemName} \nID: {GetItemID()}";
        }
        #endregion

        public override bool InteractWithBlock(Inventory.Inventory inv)
        {
            myGameobject.GetComponent<ApiaryInventory>().myblock = this;
            myGameobject.GetComponent<ApiaryInventory>().ToggleInventory(inv);
            return true;
        }

        #region Bee Combineing Stuff
        public void MakeBees(Bee queen, ref Item[] inventory)
        {
            Item[] producedItems = new Item[9];

            //* will always return a new princess and drone
            producedItems[0] = MakeBee(BeeType.PRINCESS, queen.queenBee);
            producedItems[1] = MakeBee(BeeType.DRONE, queen.queenBee);

            var repeats = UnityEngine.Random.Range(0, queen.queenBee.queen.pFertility);

            //* produces as many other children as the bee staats will allow
            for (int i = 0; i < repeats; i++)
            {
                producedItems[i + 2] = MakeBee(queen.queenBee.queen.pFertility > 6 ? (BeeType)UnityEngine.Random.Range(1, 3) : BeeType.DRONE, queen.queenBee);

                if (producedItems[i + 2] is Bee b && b.beeType != BeeType.PRINCESS)
                    producedItems[i + 2].itemStackCount = UnityEngine.Random.Range(1, (int)queen.queenBee.queen.pFertility + 1);
            }

            //* gets the produced items
            var beeProduce = BeeDictionarys.GetBeeProduce(queen.queenBee.queen.pSpecies);

            //* chnages the stack count of the produced items to the correct number
            for (int i = 0; i < beeProduce.Length; i++)
            {
                beeProduce[i].itemStackCount += UnityEngine.Random.Range(1, (int)queen.queenBee.queen.sProdSpeed + 1);
            }

            //* adds the itmes that the bee species produces into the procued item array
            for (int i = (int)queen.queenBee.queen.pFertility + 2, prod = 0; prod < beeProduce.Length; i++, prod++)
            {
                producedItems[i] = beeProduce[prod];
            }

            //* puts the items into the inventory
            for (int i = 0; i < 9; i++)
            {
                if (inventory[i + 2] != null)
                {
                    //* if the slot has the same item in it and it wont be more than the max stack ount but the new item into it
                    if (producedItems[i] == inventory[i + 2] && inventory[i + 2].itemStackCount + 1 <= inventory[i + 2].maxStackCount)
                        inventory[i + 2].itemStackCount++;
                    else
                        //* otherwise find a new slot to put the item into
                        for (int j = i; j < (9 - i); j++)
                        {
                            if (inventory[j + 2] == null)
                            {
                                inventory[j + 2] = producedItems[i];
                                break;
                            }
                            else if (producedItems[i] == inventory[j + 2] && inventory[j + 2].itemStackCount + 1 <= inventory[j + 2].maxStackCount)
                            {
                                inventory[j + 2].itemStackCount++;
                                break;
                            }
                        }
                }
                //* if the slot is empty put the item into it
                else
                    inventory[i + 2] = producedItems[i];
            }
        }

        public Bee MakeBee(BeeType beeType, QueenBee queen)
        {
            //* gives all of the primary and secondary stats to the bee
            NormalBee nb = new NormalBee()
            {
                pSpecies = CombineSpecies(queen.queen.sSpecies, queen.drone.sSpecies),
                sSpecies = CombineSpecies(queen.queen.sSpecies, queen.drone.sSpecies),

                pEffect = CombineEffect(queen.queen.sEffect, queen.drone.sEffect),
                sEffect = CombineEffect(queen.queen.sEffect, queen.drone.sEffect),

                pFertility = CombineFertility(queen.queen.sFertility, queen.drone.sFertility),
                sFertility = CombineFertility(queen.queen.sFertility, queen.drone.sFertility),

                pLifespan = CombineLifespan(queen.queen.sLifespan, queen.drone.sLifespan),
                sLifespan = CombineLifespan(queen.queen.sLifespan, queen.drone.sLifespan),

                pProdSpeed = CombineProductionSpeed(queen.queen.sProdSpeed, queen.drone.sProdSpeed),
                sProdSpeed = CombineProductionSpeed(queen.queen.sProdSpeed, queen.drone.sProdSpeed)
            };

            //* returns the new bee
            return new Bee(beeType, nb);
        }

        private BeeSpecies CombineSpecies(BeeSpecies s1, BeeSpecies s2)
        {
            BeeSpecies[] possibleSpecies = BeeDictionarys.GetCombinations(s1, s2);
            float[] weights = possibleSpecies.Length > 2 ? BeeDictionarys.GetWeights(possibleSpecies) : new float[] { 0.5f, 0.5f };

            var randomNum = Rand(weights);
            var weightsSum = 0f;

            //* when the rumber generated is less than the current sum of the weights return that bee
            for (int i = 0; i < weights.Length; i++)
            {
                if(randomNum <= weightsSum)
                {
                    return possibleSpecies[i];
                }

                weightsSum += weights[i];
            }

            //* if for some reason the weights cannot work return the first bee in the combination list
            return possibleSpecies[0];
        }

        private float Rand(float[] weights)
        {
            var totalWeights = 0f;

            //* sums the weights
            for (int i = 0; i < weights.Length; i++)
            {
                totalWeights += weights[i];
            }

            return (float)Math.Round(UnityEngine.Random.Range(0, totalWeights), 2);
        }

        private BeeLifeSpan CombineLifespan(BeeLifeSpan b1, BeeLifeSpan b2)
        {
            return (BeeLifeSpan)ReturnChange((int)b1, (int)b2, (int)BeeLifeSpan.SEATURTLE);
        }

        private uint CombineFertility(uint b1, uint b2)
        {
            return (uint)ReturnChange((int)b1, (int)b2, 5, 1);
        }

        private BeeEffect CombineEffect(BeeEffect b1, BeeEffect b2)
        {
            return (BeeEffect)ReturnChange((int)b1, (int)b2, (int)BeeEffect.POSION);
        }

        public  BeeProductionSpeed CombineProductionSpeed(BeeProductionSpeed b1, BeeProductionSpeed b2)
        {
            return (BeeProductionSpeed)ReturnChange((int)b1, (int)b2, (int)BeeProductionSpeed.FAST);
        }

        private int ReturnChange(int b1, int b2, int maxChange, int minChange = 0)
        {
            //* b1 and b2 are checked for which one is bigger than the other here as the
            //* queen my have a lower stat the an the drone and the drone is always passed in second
            var change = UnityEngine.Random.Range(b1 < b2 ? b1 : b2, (b2 > b1 ? b2 : b1) + 2);

            //* this will make it possible for the bees to mutate during combination of the stats are the same
            //* it will also cause more random mutation more mimicing nature
            change += UnityEngine.Random.Range(-mutationMultiplyer, mutationMultiplyer);

            //* as all but on ef the stats are enums they have a min/max value so need to check that this
 is not exceded
            if (change > maxChange)
                change = maxChange;
            else if (minChange > change)
                change = minChange;

            return change;

        }
        #endregion
    }
}
```

7.5 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Bedrock.cs File Reference

Classes

- class BeeGame.Blocks.Bedrock

  Bedrock Block

Namespaces

- namespace BeeGame.Blocks

7.6 Bedrock.cs

```
using System;
using BeeGame.Core.Enums;
using BeeGame.Items;
using BeeGame.Core;

namespace BeeGame.Blocks
{
    [Serializable]
    public class Bedrock : Block
    {
        #region Data
        public new static int ID => -1;
        #endregion

        #region Constructor
        public Bedrock() : base("Bedrock")
        {
            breakable = false;
        }
        #endregion

        #region Break Block
        public override void BreakBlock(THVector3 pos)
        {
            return;
        }
        #endregion

        #region Mesh
        public override Tile TexturePosition(Direction direction)
        {
            return new Tile() { x = 0, y = 0};
        }
        #endregion

        #region Overrides
        public override int GetHashCode()
        {
            return ID;
        }

        public override string ToString()
        {
            return $"{itemName} \nID: {GetItemID()}";
        }
        #endregion
    }
}
```

7.7 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Block.cs File Reference

Classes

- class BeeGame.Blocks.Block

  Base class for blocks

Namespaces

- namespace BeeGame.Blocks

7.8 Block.cs

```
using UnityEngine;
using BeeGame.Terrain.Chunks;
using BeeGame.Core.Enums;
using BeeGame.Items;
using BeeGame.Core;
using BeeGame.Core.Dictionarys;

namespace BeeGame.Blocks
{
    [System.Serializable]
    public class Block : Item
    {
        #region Data
        public new static int ID = 1;
        public bool breakable = true;
        public bool changed = true;
        public override bool placeable => true;
        #endregion

        #region Constructor
        public Block() : base()
        {
            itemName = "Stone";
        }

        public Block(string name) : base(name)
        {
        }
        #endregion

        #region Item Stuff
        public override Sprite GetItemSprite()
        {
            return SpriteDictionary.GetSprite("Stone");
        }
        #endregion

        #region Update/Break Block
        public virtual void BreakBlock(THVector3 pos)
        {
            GameObject go = Object.Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject, pos, Quaternion.identity) as GameObject;
            go.GetComponent<ItemGameObject>().item = this;
        }

        public virtual void UpdateBlock(int x, int y, int z, Chunk chunk) { }

        public virtual bool InteractWithBlock(BeeGame.Inventory.Inventory inv)
        {
            return false;
        }
        #endregion

        #region Mesh
        public virtual MeshData BlockData(Chunk chunk, int x, int y, int z, MeshData meshData, bool addToRenderMesh = true)
        {
            //* Adds the Top face of the block
            if (!chunk.GetBlock(x, y + 1, z, false).IsSolid(Direction.DOWN))
            {
                meshData = FaceDataUp(x, y, z, meshData, addToRenderMesh);
            }

            //* Adds the Bottom face of the block
            if (!chunk.GetBlock(x, y - 1, z, false).IsSolid(Direction.UP))
            {
                meshData = FaceDataDown(x, y, z, meshData, addToRenderMesh);
            }

            //* Adds the North face of the block
            if (!chunk.GetBlock(x, y, z + 1, false).IsSolid(Direction.SOUTH))
            {
                meshData = FaceDataNorth(x, y, z, meshData, addToRenderMesh);
            }

            //* Adds the South face of the block
            if (!chunk.GetBlock(x, y, z - 1, false).IsSolid(Direction.NORTH))
            {
                meshData = FaceDataSouth(x, y, z, meshData, addToRenderMesh);
            }

            //* Adds the East face of the block
            if (!chunk.GetBlock(x + 1, y, z, false).IsSolid(Direction.WEST))
            {
                meshData = FaceDataEast(x, y, z, meshData, addToRenderMesh);
            }

            //* Adds the West face of the block
            if (!chunk.GetBlock(x - 1, y, z, false).IsSolid(Direction.EAST))
            {
                meshData = FaceDataWest(x, y, z, meshData, addToRenderMesh);
            }

            return meshData;
        }

        public virtual bool IsSolid(Direction direction)
        {
            return true;
        }
        #endregion

        #region Overrides
        public override int GetHashCode()
        {
            return ID;
        }

        public override string ToString()
        {
            return $"{itemName} \nID: {GetHashCode()}";
        }
        #endregion
    }
}
```

7.9 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Chest.cs File Reference

Classes

- class BeeGame.Blocks.Chest

  Chest Block

Namespaces

- namespace BeeGame.Blocks

7.10 Chest.cs

```
using System;
using UnityEngine;
using BeeGame.Core;
using BeeGame.Terrain.Chunks;
using BeeGame.Core.Enums;
using BeeGame.Items;
using BeeGame.Inventory;
using BeeGame.Core.Dictionarys;

namespace BeeGame.Blocks
{
    [Serializable]
    public class Chest : Block
    {
        #region Data
        [NonSerialized]
        private GameObject myGameobject;

        public new static int ID => 8;
        #endregion

        #region Constructors
        public Chest() : base("Chest")
        {
            usesGameObject = true;
        }
        #endregion

        #region Block Overrides
        public override GameObject GetGameObject()
        {
            return PrefabDictionary.GetPrefab("Chest");
        }

        public override Tile TexturePosition(Direction direction)
        {
            return new Tile() { x = 0, y = 9 };
        }

        public override MeshData BlockData(Chunk chunk, int x, int y, int z, MeshData meshData, bool addToRenderMesh = true)
        {
            if (myGameobject == null)
            {
                myGameobject = UnityEngine.Object.Instantiate(PrefabDictionary.GetPrefab("Chest"), new THVector3(x, y, z) + chunk.chunkWorldPos, Quaternion.identity, chunk.transform);
                myGameobject.GetComponent<ChestInventory>().inventoryPosition = new THVector3(x, y, z) + chunk.chunkWorldPos;
                myGameobject.GetComponent<ChestInventory>().SetChestInventory();
            }
            return base.BlockData(chunk, x, y, z, meshData, true);
        }

        public override void BreakBlock(THVector3 pos)
        {
            //* removes the blocks blocks inventory save file and destroys the game object
            Serialization.Serialization.DeleteFile(myGameobject.GetComponent<ChestInventory>().inventoryName);
            UnityEngine.Object.Destroy(myGameobject);
            //* removes the collision mesh from the chunk
            base.BreakBlock(pos);
        }
        #endregion

        #region Inventory Suff
        public override bool InteractWithBlock(BeeGame.Inventory.Inventory inv)
        {
            myGameobject.GetComponent<ChestInventory>().ToggleInventory(inv);
            return true;
        }
        #endregion

        #region Overrides
        public override int GetHashCode()
        {
            return ID;
        }

        public override string ToString()
        {
            return $"{itemName}\nID{GetItemID()}";
        }
        #endregion
    }
}
```

7.11 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/CraftingTable.cs File Reference

Classes

- class BeeGame.Blocks.CraftingTable

    The Workbanch Block class

Namespaces

- namespace BeeGame.Blocks

7.12 CraftingTable.cs

```
using System;
using UnityEngine;
using BeeGame.Core;
using BeeGame.Items;
using BeeGame.Core.Enums;
using BeeGame.Terrain.Chunks;
using BeeGame.Core.Dictionarys;

namespace BeeGame.Blocks
{
    [Serializable]
    public class CraftingTable : Block
    {
        #region Data
        [NonSerialized]
        private GameObject myGameobject;

        public new static int ID => 9;
        #endregion

        #region Constructor
        public CraftingTable() : base("Workbench")
        {
            usesGameObject = true;
        }
        #endregion

        #region Crafting
        public Item ReturnShapedRecipieItem(Item[] items)
        {
            var recipie = "";

            for (int i = 0; i < items.Length; i++)
            {
                if (items[i] == null)
                {
                    recipie += "0:";
                    continue;
                }

                recipie += $"{items[i].GetItemID()}:";
            }

            return ReturnShapedRecipieItem(recipie);
        }

        public virtual Item ReturnShapelessRecipieItem(
Item[] items)
        {
            return CraftingRecipies.GetShaplessRecipieResult(items)
;
        }

        public virtual Item ReturnShapedRecipieItem(string recipie)
        {
            return BeeGame.Core.Dictionarys.
CraftingRecipies.GetShapedRecipeItem(recipie);
        }
        #endregion

        #region Block Overrides
        public override bool InteractWithBlock(Inventory.Inventory inv)
        {
            myGameobject.GetComponent<Inventory.BlockInventory.CraftingTableInventory>().myblock = this;
            myGameobject.GetComponent<Inventory.BlockInventory.CraftingTableInventory>().ToggleInventory(
inv);
            return true;
        }

        public override GameObject GetGameObject()
        {
            return PrefabDictionary.GetPrefab("CraftingTable");
        }

        public override MeshData BlockData(Chunk chunk, int x, int y, int z,
MeshData meshData, bool addToRenderMesh = true)
        {
            if (myGameobject == null)
            {
                myGameobject = UnityEngine.Object.Instantiate(
PrefabDictionary.GetPrefab("CraftingTable"), new
THVector3(x, y, z) + chunk.chunkWorldPos, Quaternion.identity, chunk.transform);
            }
            return base.BlockData(chunk, x, y, z, meshData, true);
        }

        public override void BreakBlock(THVector3 pos)
        {
            //* removes the game object
            UnityEngine.Object.Destroy(myGameobject);
            //* removes the collision mesh from the chunk
            base.BreakBlock(pos);
        }

        public override Sprite GetItemSprite()
        {
            return SpriteDictionary.GetSprite("TestSprite");
        }

        public override Tile TexturePosition(Direction direction)
        {
            return new Tile() { x = 0, y = 9 };
        }
        #endregion

        #region Overrides
        public override int GetHashCode()
        {
            return ID;
        }
        #endregion
    }
}
```

7.13 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Dirt.cs File Reference

Classes

- class BeeGame.Blocks.Dirt

  Dirt Block

Namespaces

- namespace BeeGame.Blocks

7.14 Dirt.cs

```
using System;
using BeeGame.Core.Enums;
using BeeGame.Items;
using BeeGame.Core.Dictionarys;
using UnityEngine;

namespace BeeGame.Blocks
{
    [Serializable]
    public class Dirt : Block
    {
        public new static int ID => 3;

        #region Constructor
        public Dirt() : base("Dirt"){}
        #endregion

        #region Item Stuff
        public override Sprite GetItemSprite()
        {
            return SpriteDictionary.GetSprite("Dirt");
        }
        #endregion

        #region Mesh
        public override Tile TexturePosition(Direction direction)
        {
            return new Tile { x = 2, y = 9 };
        }
        #endregion

        #region Overrides
        public override int GetHashCode()
        {
            return ID;
        }

        public override string ToString()
        {
            return $"{itemName} \nID: {GetItemID()}";
        }
        #endregion
    }
}
```

7.15 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Grass.cs File Reference

Classes

- class BeeGame.Blocks.Grass

    Grass Block

Namespaces

- namespace BeeGame.Blocks

7.16 Grass.cs

```
using System;
using UnityEngine;
using BeeGame.Core.Enums;
using BeeGame.Terrain.Chunks;
using BeeGame.Core.Dictionarys;
using BeeGame.Items;

namespace BeeGame.Blocks
{
    [Serializable]
    public class Grass : Block
    {
        public new static int ID => 4;

        #region Constructor
        public Grass() : base("Grass"){}
        #endregion

        #region Item Stuff
        public override Sprite GetItemSprite()
        {
            return SpriteDictionary.GetSprite("Grass");
        }
        #endregion

        #region Mesh
        public override void UpdateBlock(int x, int y, int z, Chunk chunk)
        {
            if (chunk.GetBlock(x, y + 1, z, false).IsSolid(Direction.DOWN))
                chunk.blocks[x, y, z] = new Dirt() { changed = changed };
        }

        public override Tile TexturePosition(Direction direction)
        {
            //All textures are on the dame Y value for the texture atlas so Y can be set
            Tile tile = new Tile()
            {
                y = 9
            };

            switch (direction)
            {
                //if we want the top face return the full grass texture
                case Direction.UP:
                    tile.x = 3;
                    return tile;
                //if we want the bottom face return the dirt texture
                case Direction.DOWN:
                    tile.x = 2;
                    return tile;
                //return the 1/2 grass testure if a side face is wanted
                default:
                    tile.x = 4;
                    return tile;
            }
        }
        #endregion

        #region Overrides
        public override int GetHashCode()
        {
            return ID;
        }

        public override string ToString()
        {
            return $"{itemName} \nID: {GetItemID()}";
        }
        #endregion
    }
}
```

7.17 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Leaves.cs File Reference

Classes

- class BeeGame.Blocks.Leaves

Namespaces

- namespace BeeGame.Blocks

7.18 Leaves.cs

```
using System;
using UnityEngine;
using BeeGame.Core.Dictionarys;
using BeeGame.Core.Enums;
using BeeGame.Items;

namespace BeeGame.Blocks
{
    [Serializable]
    public class Leaves : Block
    {
        public new static int ID => 6;

        public Leaves() : base("Leaves")
        {

        }

        #region Item Stuff
        public override Sprite GetItemSprite()
        {
            return SpriteDictionary.GetSprite("Leaves");
        }
        #endregion

        public override Tile TexturePosition(Direction direction)
        {
            return new Tile() { x = 5, y = 9 };
        }

        public override bool IsSolid(Direction direction)
        {
            return false;
        }

        #region Overrides
        public override int GetHashCode()
        {
            return ID;
        }

        public override string ToString()
        {
            return $"{itemName} \nID: {GetItemID()}";
        }
        #endregion
    }
}
```

7.19 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Blocks/Wood.cs File Reference

Classes

- class BeeGame.Blocks.Wood

Namespaces

- namespace BeeGame.Blocks

7.20 Wood.cs

```
using System;
using UnityEngine;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using BeeGame.Core.Dictionarys;
using BeeGame.Core.Enums;
using BeeGame.Items;

namespace BeeGame.Blocks
{
    [Serializable]
    public class Wood : Block
    {
        public new static int ID => 5;

        public Wood() : base("Wood")
        {

        }

        #region Item Stuff
        public override Sprite GetItemSprite()
        {
            return SpriteDictionary.GetSprite("Wood");
        }
        #endregion

        public override Tile TexturePosition(Direction direction)
        {
            return new Tile() { x = 7, y = 9 };
        }

        #region Overrides
        public override int GetHashCode()
        {
            return ID;
        }

        public override string ToString()
        {
            return $"{itemName} \nID: {GetItemID()}";
        }
        #endregion
    }
}
```

7.21 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/Bee↩ Dictionarys.cs File Reference

Classes

- class BeeGame.Core.Dictionarys.BeeDictionarys

Namespaces

- namespace BeeGame.Core.Dictionarys

7.22 BeeDictionarys.cs

```
using System.Collections.Generic;
using System.Linq;
using BeeGame.Core.Enums;
using UnityEngine;
using BeeGame.Core.Dictionarys;

namespace BeeGame.Core.Dictionarys
{
    public static class BeeDictionarys
    {
        #region Bee Combination Weights
        private static Dictionary<BeeSpecies, float> beeCombinationWeights = new Dictionary<BeeSpecies,
  float>()
        {
            {BeeSpecies.COMMON, 0.15f },
            {BeeSpecies.HEROIC, 0.06f }
        };

        public static float[] GetWeights(BeeSpecies[] species)
        {
            var returnArray = new float[species.Length];

            for (int i = 0; i < species.Length; i++)
            {
                if(beeCombinationWeights.ContainsKey(species[i]))
                    returnArray[i] = beeCombinationWeights[species[i]];
                else
                    returnArray[i] = 0.5f;
            }

            return returnArray;
        }
        #endregion

        #region Bee Combinations
        public static Dictionary<BeeSpecies[], BeeSpecies[]> beeCombinations = new Dictionary<BeeSpecies[], BeeSpecies[]>(new BeeCombinationDictionaryEqualityComparer())
        {
             { new BeeSpecies[6] { BeeSpecies.FOREST, BeeSpecies.MEADOWS, BeeSpecies.TROPICAL, BeeSpecies.WINTRY, BeeSpecies.MODEST, BeeSpecies.MARSHY }, new BeeSpecies[1] { BeeSpecies.COMMON } }
        };

        public static BeeSpecies[] GetCombinations(BeeSpecies s1, BeeSpecies s2)
        {
            var beeSpecies = new BeeSpecies[2] { s1, s2 };
            var returnBeeList = new List<BeeSpecies>();

            var keys = beeCombinations.Keys.ToArray();
            var comparor = new BeeCombinationDictionaryEqualityComparer();

            for (int i = 0; i < keys.Length; i++)
            {
                if(comparor.Equals(keys[i], beeSpecies))
                {
                    var temp = beeCombinations[keys[i]];

                    for (int j = 0; j < temp.Length; j++)
                    {
                        returnBeeList.Add(temp[i]);
                    }
                }
            }

            returnBeeList.Add(s1);
            returnBeeList.Add(s2);

            return returnBeeList.ToArray();
        }
        #endregion

        #region Bee Produce
        private static Dictionary<BeeSpecies, Items.Item[]> beeProduce = new Dictionary<BeeSpecies, Items.Item[]>()
        {
            {BeeSpecies.FOREST, new Items.Item[]{new Items.HoneyComb(HoneyCombType.HONEY) } },
            {BeeSpecies.COMMON, new Items.Item[]{new Items.HoneyComb(HoneyCombType.HONEY) } }
        };

        public static Items.Item[] GetBeeProduce(BeeSpecies species)
        {
            beeProduce.TryGetValue(species, out Items.Item[] produce);

            //* of the produce cant be found then return a honey comb as it is probly a bug
            return produce ?? new Items.Item[1] { new Items.HoneyComb(HoneyCombType.HONEY) };
        }
        #endregion

        #region Bee Colours
        private static Dictionary<BeeSpecies, Color> beeColour = new Dictionary<BeeSpecies, Color>()
        {
            {BeeSpecies.FOREST, CombColour(0, 255, 0) },
            {BeeSpecies.COMMON, CombColour(255, 0, 0) }
        };

        public static Color GetBeeColour(BeeSpecies species)
        {
            beeColour.TryGetValue(species, out Color colour);

            return colour != null ? colour : new Color();
        }
        #endregion

        #region Comb Colours
        private static Dictionary<HoneyCombType, Color> honeyCoumbColour = new Dictionary<HoneyCombType,
    Color>()
        {
            {HoneyCombType.HONEY, CombColour(255, 164, 56) },
            {HoneyCombType.ICEY, CombColour(78, 231, 231) }
        };

        private static Color CombColour(float r, float g, float b, float a = 255f)
        {
            return new Color(r / 255f, g / 255f, b / 255f);
        }

        public static Color GetCombColour(HoneyCombType type)
        {
            honeyCoumbColour.TryGetValue(type, out var temp);

            if (temp == null)
                return new Color(1, 0, 0);

            return temp;
        }
        #endregion
    }
}
```

7.23 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/↩ CraftingRecipies.cs File Reference

Classes

- class BeeGame.Core.Dictionarys.CraftingRecipies

Namespaces

- namespace BeeGame.Core.Dictionarys

7.24 CraftingRecipies.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using BeeGame.Items;
using BeeGame.Exceptipns;

namespace BeeGame.Core.Dictionarys
{
    public static class CraftingRecipies
    {
        #region Shaped Crafting
        private static Dictionary<string, Item> shapedCraftingRecipies = new Dictionary<string, Item>();

        public static void AddShapedRecipie(object[] recipie,
  Item result)
        {
            //* converts the given blocks of 3 haracters to a 9 character string
            var stringRecipie = "";

            for (int i = 0; i < 3; i++)
            {
                stringRecipie += recipie[i] as string;
            }

            //* gets what character represents which item
            for (int i = 3; i < recipie.Length; i += 2)
            {
                var character = (string)recipie[i];
                var itemID = (int)recipie[i + 1];

                //* replaces the character with the items id
                stringRecipie = stringRecipie.Replace(character, $"{itemID.ToString()}:");
            }

            //* converts empty sots " " into "0:"
            stringRecipie = stringRecipie.Replace(" ", "0:");

            //* if the recipe exists an exception is thrown as two recipies cannot be the same
            if (shapedCraftingRecipies.ContainsKey(stringRecipie))
                throw new CraftingRecipieAdditionException($"Shaped Recipie
 already exists: {stringRecipie}");

            //* adds the recipie to the dictionary
            shapedCraftingRecipies.Add(stringRecipie, result);
        }

        public static Item GetShapedRecipeItem(string recipie)
        {
            shapedCraftingRecipies.TryGetValue(recipie, out var item);

            return item;
        }
        #endregion

        #region Shapless Crafting
        private static Dictionary<string, Item> shaplessRecipies = new Dictionary<string, Item>()
        {

        };

        public static void AddShaplessRecipie(object[] recipie,
Item result)
        {
            var itemList = new List<int>();
            var stringRecpie = "";

            for (int i = 0; i < recipie.Length; i+=2)
            {
                for (int j = 0; j < (int)recipie[i+1]; j++)
                {
                    itemList.Add(int.Parse(((Item)recipie[i]).GetItemID()));
                }
            }

            itemList.Sort();

            for (int i = 0; i < itemList.Count; i++)
            {
                stringRecpie += $"{itemList[i]}:";
            }

            if (shaplessRecipies.ContainsKey(stringRecpie))
                throw new CraftingRecipieAdditionException($"Shaped Recipie
 already exists: {stringRecpie}");

            shaplessRecipies.Add(stringRecpie, result);
        }

        public static string GetShaplessRecipieString(
Item[] recipie)
        {
            var IDList = new List<int>();
            var stringRecipe = "";

            //* converts tthe given item list to an ID list so it can be sorted
            for (int i = 0; i < recipie.Length; i++)
            {
                if(recipie[i] != null)
                    IDList.Add(recipie[i].GetHashCode());
            }

            IDList.Sort();

            //* converts the sorted ID list to a string so can be used as a dictionary key
            for (int i = 0; i < IDList.Count; i++)
            {
                //* : after each ID as it is possible for ID clashes without eg ID: 11 can be seen as 2 *
 ID: 1
                stringRecipe += $"{IDList[i]}:";
            }

            return stringRecipe;
        }

        public static Item GetShaplessRecipieResult(int[] recipie)
        {
            var list = recipie.ToList();
            list.Sort();

            var stringRecipe = "";

            for (int i = 0; i < list.Count; i++)
            {
                stringRecipe += $"{list[i]}:";
            }

            return GetShaplessRecipieResult(stringRecipe);
        }

        public static Item GetShaplessRecipieResult(string recipie)
        {
            shaplessRecipies.TryGetValue(recipie, out var item);

            return item;
        }

        public static Item GetShaplessRecipieResult(
      Item[] recipie)
        {
            shaplessRecipies.TryGetValue(GetShaplessRecipieString(recipie), out var item);

            return item;
        }
        #endregion
    }
}
```

7.25 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/↩ EqualityComperors.cs File Reference

Classes

- class BeeGame.Core.Dictionarys.BeeCombinationDictionaryEqualityComparer

Namespaces

- namespace BeeGame.Core.Dictionarys

7.26 EqualityComperors.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using BeeGame.Core.Enums;

namespace BeeGame.Core.Dictionarys
{
    public class BeeCombinationDictionaryEqualityComparer :
      IEqualityComparer<BeeSpecies[]>
    {
        public bool Equals(BeeSpecies[] x, BeeSpecies[] y)
        {
            if (x.Contains(y[0]) && x.Contains(y[1]))
            {
                //* if the x length is greater than 2 this means that the combination can have duplicate
       bees for a product
                if (x.Length > 2)
                    return true;

                //* if 1 means both y elements are the same so no combination has been found
                if(y.Intersect(x).Count() <= 1)
                    return false;

                return true;
            }

            return false;
        }

        public int GetHashCode(BeeSpecies[] obj)
        {
            unchecked
            {
                int hashcode = 13;

                for (int i = 0; i < obj.Length; i++)
                {
                    hashcode += (int)obj[i];
                }

                return hashcode;
            }
        }
    }
}
```

7.27 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/PrefabDictionary.cs File Reference

Classes

- class BeeGame.Core.Dictionarys.PrefabDictionary

  The prefabs avaliable to the game

Namespaces

- namespace BeeGame.Core.Dictionarys

7.28 PrefabDictionary.cs

```
using System.Collections.Generic;
using UnityEngine;

namespace BeeGame.Core.Dictionarys
{
    public static class PrefabDictionary
    {
        private static Dictionary<string, GameObject> prefabDictionary = new Dictionary<string, GameObject>
();

        public static void LoadPrefabs()
        {
            prefabDictionary = Resources.Resources.GetPrefabs();
        }

        public static GameObject GetPrefab(string prefab)
        {
            return prefabDictionary[prefab];
        }
    }
}
```

7.29 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Dictionarys/SpriteDictionary.cs File Reference

Classes

- class BeeGame.Core.Dictionarys.SpriteDictionary

  All of the sprites avaliable to the game

Namespaces

- namespace BeeGame.Core.Dictionarys

7.30 SpriteDictionary.cs

```
using System.Collections.Generic;
using UnityEngine;

namespace BeeGame.Core.Dictionarys
{
    public static class SpriteDictionary
    {
        private static Dictionary<string, Sprite> itemSpriteDictionary = new Dictionary<string, Sprite>();

        public static Sprite GetSprite(string spriteName)
        {
            itemSpriteDictionary.TryGetValue(spriteName, out Sprite sprite);

            if (sprite == null)
                return new Sprite();

            return sprite;
        }

        public static void LoadSprites()
        {
            itemSpriteDictionary = Resources.Resources.GetSprites();
        }
    }
}
```

7.31 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Enums/Enums.cs File Reference

Namespaces

- namespace BeeGame.Core.Enums

Enumerations

- enum BeeGame.Core.Enums.HoneyCombType { BeeGame.Core.Enums.HoneyCombType.HONEY, Bee←Game.Core.Enums.HoneyCombType.ICEY }

  Honey Comb Types
- enum BeeGame.Core.Enums.BeeSpecies {
  BeeGame.Core.Enums.BeeSpecies.FOREST, BeeGame.Core.Enums.BeeSpecies.MEADOWS, Bee←Game.Core.Enums.BeeSpecies.TROPICAL, BeeGame.Core.Enums.BeeSpecies.WINTRY,
  BeeGame.Core.Enums.BeeSpecies.MODEST, BeeGame.Core.Enums.BeeSpecies.MARSHY, BeeGame.←Core.Enums.BeeSpecies.ENDER, BeeGame.Core.Enums.BeeSpecies.MONASTIC,
  BeeGame.Core.Enums.BeeSpecies.STEADFAST, BeeGame.Core.Enums.BeeSpecies.VALIANT, Bee←Game.Core.Enums.BeeSpecies.COMMON, BeeGame.Core.Enums.BeeSpecies.CULTIVATED,
  BeeGame.Core.Enums.BeeSpecies.DILIGENT, BeeGame.Core.Enums.BeeSpecies.RURAL, BeeGame.←Core.Enums.BeeSpecies.FARMERLY, BeeGame.Core.Enums.BeeSpecies.AGRARIAN,
  BeeGame.Core.Enums.BeeSpecies.UNWEARY, BeeGame.Core.Enums.BeeSpecies.INDUSTRIOUS,
  BeeGame.Core.Enums.BeeSpecies.ICY, BeeGame.Core.Enums.BeeSpecies.GLACIAL,
  BeeGame.Core.Enums.BeeSpecies.NOBLE, BeeGame.Core.Enums.BeeSpecies.IMPERIAL, BeeGame.←Core.Enums.BeeSpecies.MAJESTIC, BeeGame.Core.Enums.BeeSpecies.MIRY,
  BeeGame.Core.Enums.BeeSpecies.BOGGY, BeeGame.Core.Enums.BeeSpecies.HERIOC, BeeGame.←Core.Enums.BeeSpecies.PHANTASMAL, BeeGame.Core.Enums.BeeSpecies.SPECTRAL,
  BeeGame.Core.Enums.BeeSpecies.HERMETIC, BeeGame.Core.Enums.BeeSpecies.SECLUDED, Bee←Game.Core.Enums.BeeSpecies.SINISTER, BeeGame.Core.Enums.BeeSpecies.FIENDISH,

BeeGame.Core.Enums.BeeSpecies.DEMONIC, BeeGame.Core.Enums.BeeSpecies.FRUGAL, Bee↩
Game.Core.Enums.BeeSpecies.AUSTER, BeeGame.Core.Enums.BeeSpecies.VINDICTIVE,
BeeGame.Core.Enums.BeeSpecies.EXOTIC, BeeGame.Core.Enums.BeeSpecies.ENDEMIC, BeeGame.↩
Core.Enums.BeeSpecies.VENGEFUL, BeeGame.Core.Enums.BeeSpecies.AVENGING,
BeeGame.Core.Enums.BeeSpecies.SETADFAST, BeeGame.Core.Enums.BeeSpecies.HEROIC }

The different possible bee Species

- enum BeeGame.Core.Enums.BeeType { BeeGame.Core.Enums.BeeType.QUEEN, BeeGame.Core.↩
Enums.BeeType.DRONE, BeeGame.Core.Enums.BeeType.PRINCESS }

  The different bee types
- enum BeeGame.Core.Enums.BeeTempPreferance {
BeeGame.Core.Enums.BeeTempPreferance.FROZEN, BeeGame.Core.Enums.BeeTempPreferance.COLD,
BeeGame.Core.Enums.BeeTempPreferance.TEMPERATE, BeeGame.Core.Enums.BeeTempPreferance.↩
HOT,
BeeGame.Core.Enums.BeeTempPreferance.HELL }

  The different bee temp preferences
- enum BeeGame.Core.Enums.BeeLifeSpan {
BeeGame.Core.Enums.BeeLifeSpan.HUMMINGBIRD, BeeGame.Core.Enums.BeeLifeSpan.SHORTEST,
BeeGame.Core.Enums.BeeLifeSpan.SHORT, BeeGame.Core.Enums.BeeLifeSpan.NORMAL,
BeeGame.Core.Enums.BeeLifeSpan.LONG, BeeGame.Core.Enums.BeeLifeSpan.LONGEST, BeeGame.↩
Core.Enums.BeeLifeSpan.SEATURTLE }

  The lifespan of the bee
- enum BeeGame.Core.Enums.BeeProductionSpeed { BeeGame.Core.Enums.BeeProductionSpeed.SLOW,
BeeGame.Core.Enums.BeeProductionSpeed.NORMAL, BeeGame.Core.Enums.BeeProductionSpeed.FA↩
ST }

  How fast the bee produces items
- enum BeeGame.Core.Enums.BeeEffect { BeeGame.Core.Enums.BeeEffect.NONE, BeeGame.Core.↩
Enums.BeeEffect.POSION }

  Any effects of the bee
- enum BeeGame.Core.Enums.BeeHumidityPreferance {
BeeGame.Core.Enums.BeeHumidityPreferance.ARID, BeeGame.Core.Enums.BeeHumidityPreferance.D↩
RY, BeeGame.Core.Enums.BeeHumidityPreferance.TEMPERATE, BeeGame.Core.Enums.BeeHumidity↩
Preferance.MOIST,
BeeGame.Core.Enums.BeeHumidityPreferance.HUMID }

  Humidity preferences of the bee
- enum BeeGame.Core.Enums.Direction {
BeeGame.Core.Enums.Direction.NORTH, BeeGame.Core.Enums.Direction.EAST, BeeGame.Core.↩
Enums.Direction.SOUTH, BeeGame.Core.Enums.Direction.WEST,
BeeGame.Core.Enums.Direction.UP, BeeGame.Core.Enums.Direction.DOWN }

  Direction in the game

7.32 Enums.cs

```
namespace BeeGame.Core.Enums
{
    public enum HoneyCombType
    {
        HONEY, ICEY
    };

    #region BeeStuff
    public enum BeeSpecies
    {
        FOREST, MEADOWS, TROPICAL, WINTRY, MODEST,
MARSHY, ENDER, MONASTIC, STEADFAST, VALIANT,
COMMON, CULTIVATED, DILIGENT, RURAL, FARMERLY,
AGRARIAN, UNWEARY, INDUSTRIOUS, ICY, GLACIAL,
NOBLE, IMPERIAL, MAJESTIC, MIRY, BOGGY, HERIOC,
PHANTASMAL, SPECTRAL, HERMETIC, SECLUDED,
SINISTER, FIENDISH, DEMONIC, FRUGAL, AUSTER,
VINDICTIVE, EXOTIC, ENDEMIC, VENGEFUL, AVENGING,
SETADFAST, HEROIC
    };

    public enum BeeType
    {
        QUEEN, DRONE, PRINCESS
    };

    public enum BeeTempPreferance
    {
        FROZEN, COLD, TEMPERATE, HOT, HELL
    };

    public enum BeeLifeSpan
    {
        HUMMINGBIRD, SHORTEST, SHORT, NORMAL, LONG,
LONGEST, SEATURTLE
    };

    public enum BeeProductionSpeed
    {
        SLOW, NORMAL, FAST
    };

    public enum BeeEffect
    {
        NONE, POSION
    }

    public enum BeeHumidityPreferance
    {
        ARID, DRY, TEMPERATE, MOIST, HUMID
    };
    #endregion BeeStuff

    public enum Direction
    {
        NORTH, EAST, SOUTH, WEST, UP, DOWN
    };
}
```

7.33 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/Events.cs File Reference

Classes

- class BeeGame.Core.Events

Namespaces

- namespace BeeGame.Core

7.34 Events.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using BeeGame.Items;
using BeeGame.Blocks;

namespace BeeGame.Core
{
    public static class Events
    {
        public delegate void ItemCraftedEvent(Item item);
        public static ItemCraftedEvent shapedRecipieCrafted;
        public static ItemCraftedEvent shaplessRecipieCrafted;
        public static ItemCraftedEvent beeCraftedEvent;

        public static void CallShapedRecipieCraftedEvent(Item item) => shapedRecipieCrafted?.Invoke(
item);
        public static void CallShaplessRecipirCraftedEvent(Item item) => shaplessRecipieCrafted?.Invoke
(item);
        public static void CallBeeCraftedEvent(Item item) => beeCraftedEvent?.Invoke(item);
    }
}
```

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

Classes

- class BeeGame.Core.Extensions

Namespaces

- namespace BeeGame.Core

7.36 Extensions.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reflection;
using System.Text;
using UnityEngine;
using System.Threading;
using BeeGame.Items;

namespace BeeGame.Core
{
    public static class Extensions
    {
        public static T CloneObject<T>(this T obj)
        {
            //* gets the tyoe of the given object
            Type typeSource = obj.GetType();

            //* makes a new object of type T
            T objTarget = (T)Activator.CreateInstance(typeSource);

            //* gets the properties in T
            PropertyInfo[] propertyInfo = typeSource.GetProperties(BindingFlags.Public | BindingFlags.NonPublic | BindingFlags.Instance);

            //* applies the properties in T to the new type T object
            foreach (var property in propertyInfo)
            {
                if (property.CanWrite)
                {
                    //* if the propertly is a value just set it
                    if (property.PropertyType.IsValueType || property.PropertyType.IsEnum || property.PropertyType.Equals(typeof(string)))
                    {
                        property.SetValue(objTarget, property.GetValue(obj, null), null);
                    }
                    else
                    {
                        //* if the propertly is not a value type this function will need to be called recursivly as it could also have non value type veriables
                        object propertyValue = property.GetValue(obj, null);

                        if (propertyValue == null)
                        {
                            property.SetValue(objTarget, null, null);
                        }
                        else
                        {
                            property.SetValue(objTarget, propertyValue.CloneObject(), null);
                        }
                    }
                }
            }

            //* gets all of the field in T
            FieldInfo[] fieldInfo = typeSource.GetFields();

            //* applies all of the fiels of T to the new object if type T in the same manor that the properites are applied
            foreach (var field in fieldInfo)
            {
                if(field.FieldType.IsValueType || field.FieldType.IsEnum || field.FieldType.Equals(typeof(
string)))
                {
                    field.SetValue(objTarget, field.GetValue(obj));
                }
                else
                {
                    object fieldValue = field.GetValue(obj);

                    if(fieldValue == null)
                    {
                        field.SetValue(objTarget, null);
                    }
                    else
                    {
                        field.SetValue(objTarget, field.CloneObject());
                    }
                }
            }

            return objTarget;
        }

        public static Sprite ColourSprite(this Sprite sprite, Color colour, Color[] coloursToAvoid = null, bool setTransparentToWhite = false)
        {
            Texture2D tex = new Texture2D((int)sprite.rect.width, (int)sprite.rect.height)
            {
                filterMode = FilterMode.Point,
                wrapMode = TextureWrapMode.Clamp
            };

            //* sets the teture pixels to the pixels of teh sprite so the original sprite is not modified
            tex.SetPixels(sprite.texture.GetPixels());

            for (int x = 0; x < tex.width; x++)
            {
                for (int y = 0; y < tex.height; y++)
                {
                    //* if we dont have to avoid any colours set the pixel
                    if (coloursToAvoid == null)
                    {
                        tex.SetPixel(x, y, tex.GetPixel(x, y) * colour);
                    }
                    else
                    {
                        for (int i = 0; i < coloursToAvoid.Length; i++)
                        {
                            //* if this colour should be avoided skip this iteration of the loop and move on
                            if (tex.GetPixel(x, y) == coloursToAvoid[i])
                                goto Skip;
                        }

                        tex.SetPixel(x, y, tex.GetPixel(x, y) * colour);
                    }

                    //* if transparent pixels should be set to white do that
                    if (setTransparentToWhite && tex.GetPixel(x, y).a == 0)
                        tex.SetPixel(x, y, Color.white);

                    Skip:
                        continue;
                }
            }

            //* apply the new texture with its colours
            tex.Apply();

            //* return the Texture2D as a sprite
            return Sprite.Create(tex, new Rect(0, 0, tex.width, tex.height), new THVector2(0.5f, 0.5f));
        }

        public static void SpawnItem(this Item item, THVector3 position, Quaternion rotation = new Quaternion())
        {
            GameObject go = MonoBehaviour.Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject, position, rotation) as GameObject;
            go.GetComponent<ItemGameObject>().item = item;
        }
    }
}
```

7.37 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityTypeReplacements/↩ THInput.cs File Reference

Classes

- class BeeGame.Core.THInput

  My implementation of the unity input system. Acts as a buffer layer to the unity system so that the input keys can be changed at runtime

Namespaces

- namespace BeeGame.Core

7.38 THInput.cs

```
using System;
using System.Collections.Generic;
using UnityEngine;
using BeeGame.Exceptipns;

namespace BeeGame.Core
{
    public static class THInput
    {
        private static Dictionary<string, object> inputButtons = 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 }
        };

        public static bool isAnotherInventoryOpen;

        public static bool blockInventoryJustClosed;
        internal static bool chestOpen;

        public static bool GetButtonDown(string button)
        {
            if (!inputButtons.ContainsKey(button))
            {
                throw new InputException($"Key input name not defined: {button}");
            }

            switch (inputButtons[button])
            {
                case KeyCode[] arry:
                    //*for each posible key, check if it was pressed and if it was return that it was, if
 none of them was poressed return false
                    foreach (var item in arry)
                    {
                        if (Input.GetKeyDown(item))
                        {
                            return true;
                        }
                    }

                    return false;
                default:
                    return Input.GetKeyDown((KeyCode)inputButtons[button]);
            }
        }

        public static bool GetButton(string button)
        {
            if (!inputButtons.ContainsKey(button))
            {
                throw new InputException($"Key input name not defined: {button}");
            }

            switch (inputButtons[button])
            {
                case KeyCode[] arry:
                    //*for each posible key, check if it was pressed and if it was return that it was, if
 none of them was poressed return false
                    foreach (var item in arry)
                    {
                        if (Input.GetKey(item))
                        {
                            return true;
                        }
                    }

                    return false;
                default:
                    return Input.GetKey((KeyCode)inputButtons[button]);
            }
        }

        public static bool GetButtonUp(string button)
        {
            if (!inputButtons.ContainsKey(button))
            {
                throw new InputException($"Key input name not defined: {button}");
            }

            switch (inputButtons[button])
            {
                case KeyCode[] arry:
                    //*for each posible key, check if it was pressed and if it was return that it was, if
 none of them was poressed return false
                    foreach (var item in arry)
                    {
                        if (Input.GetKeyUp(item))
                        {
                            return true;
                        }
                    }

                    return false;
                default:
                    return Input.GetKeyUp((KeyCode)inputButtons[button]);
            }
        }

        public static int GetAxis(string axis)
        {
            int returnAxis = 0;

            if (axis == "Horizontal")
            {
                if (GetButton("Right"))
                {
                    returnAxis += 1;
                }

                if (GetButton("Left"))
                {
                    returnAxis -= 1;
                }
            }
            else if (axis == "Vertical")
            {
                if (GetButton("Forward"))
                {
                    returnAxis += 1;
                }

                if (GetButton("Backward"))
                {
                    returnAxis -= 1;
                }
            }

            return returnAxis;
        }
    }
}
```

7.39 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityTypeReplacements/↩ THQuaternion.cs File Reference

Classes

- struct BeeGame.Core.UnityTypeReplacements.THQuaternion

Namespaces

- namespace BeeGame.Core.UnityTypeReplacements

7.40 THQuaternion.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace BeeGame.Core.UnityTypeReplacements
{
    public struct THQuaternion
    {
        public float x;
        public float y;
        public float z;
        public float w;
    }
}
```

7.41 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityTypeReplacements/↩ THVector2.cs File Reference

Classes

- struct BeeGame.Core.THVector2

    Serilializable version of Vector2

Namespaces

- namespace BeeGame.Core

7.42 THVector2.cs

```
using System;
using UnityEngine;

namespace BeeGame.Core
{
    [Serializable]
    public struct THVector2
    {
        #region Data
        public float x;
        public float y;
        #endregion

        #region Constructor
        public THVector2(float x, float y)
        {
            this.x = x;
            this.y = y;
        }

        public THVector2(THVector2 vec2)
        {
            this = vec2;
        }

        public THVector2(Vector2 vec2)
        {
            this = vec2;
        }
        #endregion

        #region Overrides
        public override bool Equals(object obj)
        {
            if (!(obj is THVector2))
                return false;
            if (obj.GetHashCode() == GetHashCode())
                return true;
            return false;
        }

        public override int GetHashCode()
        {
            unchecked
            {
                int hash = 13;

                hash *= 443 * x.GetHashCode();
                hash *= 373 * y.GetHashCode();

                return hash;
            }
        }

        public override string ToString()
        {
            return $"{x}, {y}";
        }

        public static bool operator ==(THVector2 a, THVector2 b)
        {
            return a.Equals(b);
        }
        public static bool operator !=(THVector2 a, THVector2 b)
        {
            return !(a == b);
        }

        public static THVector2 operator +(THVector2 a, THVector2 b)
        {
            a.x += b.x;
            a.y += b.y;

            return a;
        }
        public static THVector2 operator +(THVector2 a, float b)
        {
            a.x += b;
            a.y += b;

            return a;
        }
        public static THVector2 operator +(float a, THVector2 b)
        {
            return new THVector2(a + b.x, a + b.y);
        }
        public static THVector2 operator -(THVector2 a, THVector2 b)
        {
            a.x -= b.x;
            a.y -= b.y;

            return a;
        }
        public static THVector2 operator -(THVector2 a, float b)
        {
            a.x += b;
            a.y += b;

            return a;
        }
        public static THVector2 operator -(float a, THVector2 b)
        {
            return new THVector2(a - b.x, a - b.y);
        }
        public static THVector2 operator *(THVector2 a,
 THVector2 b)
        {
            a.x *= b.x;
            a.y *= b.y;

            return a;
        }
        public static THVector2 operator *(THVector2 a, float b)
        {
            a.x *= b;
            a.y *= b;

            return a;
        }
        public static THVector2 operator *(float a, THVector2 b)
        {
            return new THVector2(a * b.x, a * b.y);
        }
        public static THVector2 operator /(THVector2 a,
 THVector2 b)
        {
            a.x /= b.x;
            a.y /= b.y;

            return a;
        }
        public static THVector2 operator /(THVector2 a, float b)
        {
            a.x /= b;
            a.y /= b;

            return a;
        }
        public static THVector2 operator /(float a, THVector2 b)
        {
            return new THVector2(a / b.x, a / b.y);
        }
        #endregion

        #region Implicit Operators
        public static implicit operator Vector2(THVector2 vec2)
        {
            return new Vector2(vec2.x, vec2.y);
        }

        public static implicit operator THVector2(Vector2 vec2)
        {
            return new THVector2(vec2.x, vec2.y);
        }
        #endregion
    }
}
```

7.43 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Core/UnityTypeReplacements/↩ THVector3.cs File Reference

Classes

- struct BeeGame.Core.THVector3

  Serializable version of Vector3

Namespaces

- namespace BeeGame.Core

7.44 THVector3.cs

```
using System;
using UnityEngine;

namespace BeeGame.Core
{
    [Serializable]
    public struct THVector3
    {
        #region Data
        public float x;
        public float y;
        public float z;
        #endregion

        #region Constructors
        public THVector3(float x, float y, float z)
        {
            this.x = x;
            this.y = y;
            this.z = z;
        }

        public THVector3(THVector3 vec3)
        {
            this = vec3;
        }

        public THVector3(Vector3 vec3)
        {
            this = vec3;
        }

        public THVector3(Terrain.ChunkWorldPos vec3)
        {
            this = vec3;
        }
        #endregion

        #region Methods
        public static float Distance(THVector3 a, THVector3 b)
        {
            return (float)Math.Sqrt(Math.Pow((a.x - b.x), 2) + Math.Pow((a.y - b.
y), 2) + Math.Pow((a.z - b.z), 2));
        }
        #endregion

        #region Overrides
        public override bool Equals(object obj)
        {
            if (!(obj is THVector3))
                return false;
            if (obj.GetHashCode() == GetHashCode())
                return true;
            return false;
        }

        public override int GetHashCode()
        {
            unchecked
            {
                int hash = 13;

                hash *= 443 * x.GetHashCode();
                hash *= 373 * y.GetHashCode();
                hash *= 127 * z.GetHashCode();

                return hash;
            }
        }

        public override string ToString()
        {
            return $"{x}, {y}, {z}";
        }

        public static bool operator ==(THVector3 a, THVector3 b)
        {
            return a.Equals(b);
        }
        public static bool operator !=(THVector3 a, THVector3 b)
        {
            return !(a == b);
        }

        public static THVector3 operator +(THVector3 a,
THVector3 b)
        {
            a.x += b.x;
            a.y += b.y;
            a.z += b.z;

            return a;
        }
        public static THVector3 operator +(THVector3 a, float b)
        {
            a.x += b;
            a.y += b;
            a.z += b;

            return a;
        }
        public static THVector3 operator +(float a, THVector3 b)
        {
            return new THVector3(a + b.x, a + b.y, a + b.z);
        }
        public static THVector3 operator -(THVector3 a,
THVector3 b)
        {
            a.x -= b.x;
            a.y -= b.y;
            a.z -= b.z;

            return a;
        }
        public static THVector3 operator -(THVector3 a, float b)
        {
            a.x += b;
            a.y += b;
            a.z += b;

            return a;
        }
        public static THVector3 operator -(float a, THVector3 b)
        {
            return new THVector3(a - b.x, a - b.y, a - b.z);
        }
        public static THVector3 operator *(THVector3 a,
THVector3 b)
        {
            a.x *= b.x;
            a.y *= b.y;
            a.z *= b.z;

            return a;
        }
        public static THVector3 operator *(THVector3 a, float b)
        {
            a.x *= b;
            a.y *= b;
            a.z *= b;

            return a;
        }
        public static THVector3 operator *(float a, THVector3 b)
        {
            return new THVector3(a * b.x, a * b.y, a * b.z);
        }
        public static THVector3 operator /(THVector3 a,
THVector3 b)
        {
            a.x /= b.x;
            a.y /= b.y;
            a.z /= b.z;

            return a;
        }
        public static THVector3 operator /(THVector3 a, float b)
        {
            a.x /= b;
            a.y /= b;
            a.z /= b;

            return a;
        }
        public static THVector3 operator /(float a, THVector3 b)
        {
            return new THVector3(a / b.x, a / b.y, a / b.z);
        }
        #endregion

        #region Implicit Operators
        public static implicit operator Vector3(THVector3 vec3)
        {
            return new Vector3(vec3.x, vec3.y, vec3.z);
        }

        public static implicit operator THVector3(Vector3 vec3)
        {
            return new THVector3(vec3.x, vec3.y, vec3.z);
        }
        #endregion

        #region Explicit Operators
        public static explicit operator Quaternion(THVector3 vec3)
        {
            return new Quaternion(vec3.x, vec3.y, vec3.z, 0);
        }
        #endregion
    }
}
```

7.45 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Exceptipns/Crafting↩ RecipieAdditionException.cs File Reference

Classes

- class BeeGame.Exceptipns.CraftingRecipieAdditionException

Namespaces

- namespace BeeGame.Exceptipns

7.46 CraftingRecipieAdditionException.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace BeeGame.Exceptipns
{
    public class CraftingRecipieAdditionException : Exception
    {
        public CraftingRecipieAdditionException() : base()
        {

        }

        public CraftingRecipieAdditionException(string message) : base(
message)
        {

        }

        public CraftingRecipieAdditionException(string message, Exception
innerException) : base(message, innerException)
        {

        }
    }
}
```

7.47 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Exceptipns/Input↩ Exception.cs File Reference

Classes

- class BeeGame.Exceptipns.InputException

Namespaces

- namespace BeeGame.Exceptipns

7.48 InputException.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace BeeGame.Exceptipns
{
    public class InputException : Exception
    {
        public InputException() : base()
        {

        }

        public InputException(string message) : base(message)
        {

        }

        public InputException(string message, Exception innerException) : base(message,
innerException)
        {

        }
    }
}
```

7.49 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/BlockInventory/↩ ApiaryInventory.cs File Reference

Classes

- class BeeGame.Inventory.ApiaryInventory

  Inventory for Apiarys Apiary

Namespaces

- namespace BeeGame.Inventory

7.50 ApiaryInventory.cs

```
using UnityEngine;
using UnityEngine.UI;
using BeeGame.Blocks;

namespace BeeGame.Inventory
{
    public class ApiaryInventory : ChestInventory
    {
        #region Data
        private bool beesCombineing;

        public float combinationTime = 0;

        public Slider timerSlideer;
        #endregion

        #region Unity Methods
        private void Update()
        {
            //* Updates the base class as unity Update function does not run on parent classes
            UpdateChestInventory();

            //* if the apiary is not an item on the ground and bees are not currently combineing check is
 bees should be combineing
            if(items.itemsInInventory.Length > 0 && !beesCombineing)
                CheckforBees();

            //* if the currently combineing bees has finished combineing
            if (combinationTime < 0 && beesCombineing)
            {
                //* make the items that the bees should make and destroy the spent queen
                ((Apiary)myblock).MakeBees(items.itemsInInventory[0] as Items.Bee, ref items.
itemsInInventory);
                beesCombineing = false;
                items.itemsInInventory[0] = null;

                //* save the channges to the inventory
                SaveInv();
            }
        }

        private void FixedUpdate()
        {
            //* if bees are combineing reduce the combination time
            if (beesCombineing)
                timerSlideer.value = combinationTime -= 0.1f;
        }
        #endregion

        #region Apiary Stuff
        private void CheckforBees()
        {
            Items.Item posOneItem = items.itemsInInventory[0];
            Items.Item posTwoItem = items.itemsInInventory[1];

            //* the item is checkd if it is a bee and if it is then a new variable is made for convenience
            //* if it is a queen then just set the combination time and go
            if (posOneItem is Items.Bee b && b.beeType == Core.Enums.BeeType.QUEEN)
            {
                combinationTime = ((float)b.queenBee.queen.pLifespan + 1) * 2;
                beesCombineing = true;
                SaveInv();

                timerSlideer.maxValue = combinationTime;

                return;
            }

            //* of one bee is a princess and another is a drone in the correct slots combine them
            if(posOneItem is Items.Bee b1 && posTwoItem is Items.Bee b2 && b1.beeType == Core.Enums.BeeType
.PRINCESS && b2.beeType == Core.Enums.BeeType.DRONE)
            {
                //* comvert the princess to a queen with the paired drone
                Items.Bee.ConvertToQueen(ref b1, b2.normalBee);

                //* reduce number of drones in slot by 1 and check it is a valid stack number
                items.itemsInInventory[1].itemStackCount -= 1;
                slots[0].item = b1;

                if (items.itemsInInventory[1].itemStackCount <= 0)
                    items.itemsInInventory[1] = null;

                //* set the combination time
                combinationTime = ((float)b1.queenBee.queen.pLifespan + 1) * 2;
                beesCombineing = true;

                SaveInv();

                //* set the slider max to the combination time
                timerSlideer.maxValue = combinationTime;
            }
        }
        #endregion

        #region Overrides
        public override void SetChestInventory(string invName = "Apiary")
        {
            base.SetChestInventory("Apiary" );
        }
        #endregion
    }
}
```

7.51 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/BlockInventory/↩ ChestInventory.cs File Reference

Classes

- class BeeGame.Inventory.ChestInventory

  Incentory for the chests

Namespaces

- namespace BeeGame.Inventory

7.52 ChestInventory.cs

```
using BeeGame.Core;
using BeeGame.Terrain;
using UnityEngine;
using static BeeGame.Core.THInput;

namespace BeeGame.Inventory
{
    public class ChestInventory : Inventory
    {
        #region Data
        public THVector3 inventoryPosition;
        public Inventory playerinventory;
        public GameObject inventory;

        public int inventorySize;
        #endregion

        #region Unity Methods
        void Update()
        {
            UpdateChestInventory();
        }

        public void UpdateChestInventory()
        {
            //* the chest should always have a player inventory when it does this but checks just in case
            if (playerinventory != null)
                UpdateBase();

            //* checks if the inventory should be closed
            if (GetButtonDown("Player Inventory") && thisInventoryOpen && floatingItem == null)
                ToggleInventory(playerinventory);
        }
        #endregion

        public virtual void SetChestInventory(string invName = "Chest")
        {
            SetInventorySize(inventorySize);
            //* sets the UI to not be seen as inventorys cannot start open
            inventory.SetActive(false);

            //* sets the name and postion if this inventory used during serialization and deserialization
            inventoryName = $"{invName} @ {(ChunkWorldPos)inventoryPosition}";

            //* loads the inventory if it had had items put in it last time it existed
            Serialization.Serialization.DeSerializeInventory(this, inventoryName);
        }

        #region Player Inventory
        void SetPlayerItems()
        {
            for (int i = 0; i < playerinventory.items.itemsInInventory.Length; i++)
            {
                items.itemsInInventory[i + (inventorySize - 36)] = playerinventory.
items.itemsInInventory[i];
            }
        }

        void ApplyPlayerItems()
        {
            for (int i = 0; i < playerinventory.items.itemsInInventory.Length; i++)
            {
                playerinventory.items.itemsInInventory[i] = items.itemsInInventory[i + (inventorySize - 36)];
            }

            playerinventory.SaveInv();
        }
        #endregion

        public override void ToggleInventory(Inventory inv)
        {
            //* sets the player inventory
            playerinventory = inv;

            thisInventoryOpen = !thisInventoryOpen;

            isAnotherInventoryOpen = thisInventoryOpen;

            inventory.SetActive(!inventory.activeInHierarchy);

            if (inventory.activeInHierarchy)
            {
                chestOpen = true;

                //* stops the player invnetory from being opened immidiatly after this is closed
                blockInventoryJustClosed = true;
                SetPlayerItems();
                //* hides and locks the cursor
                Cursor.lockState = CursorLockMode.None;
                Cursor.visible = true;
            }
            else
            {
                chestOpen = false;

                //* puts the items into the chest
                //* shows and unlocks the cursor
                ApplyPlayerItems();
                Cursor.lockState = CursorLockMode.Locked;
                Cursor.visible = false;
            }
        }
    }
}
```

7.53 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/BlockInventory/↩ CraftingTableInventory.cs File Reference

Classes

- class BeeGame.Inventory.BlockInventory.CraftingTableInventory

  Invnetory for the CraftingTable Block

Namespaces

- namespace BeeGame.Inventory.BlockInventory

7.54 CraftingTableInventory.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using BeeGame.Core;
using BeeGame.Blocks;
using BeeGame.Items;

namespace BeeGame.Inventory.BlockInventory
{
    public class CraftingTableInventory : ChestInventory
    {
        #region Data
        public delegate void ItemRemovedFromResult();
        public ItemRemovedFromResult result;
        #endregion

        #region Unity Methods
        protected void Start()
        {
            SetChestInventory();
            result = CraftedItemRemoved;
        }


        protected void Update()
        {
            UpdateChestInventory();

            if (inventory.activeInHierarchy)
            {
                CheckShapedRecipie();

                //* checks for shapless recipies second
                if(items.itemsInInventory[9] == null)
                    CheckShapelessRecipie();
            }
        }

        protected void OnDestroy()
        {
            //* just ensures no memory leaks occur
            result -= CraftedItemRemoved;
        }
        #endregion

        #region Crafting Stuff
        public virtual void CheckShapedRecipie()
        {
            var items = new Item[9];

            for (int i = 0; i < items.Length; i++)
            {
                items[i] = base.items.itemsInInventory[i];
            }

            //* if it is a recipie put the result into the crafting result slot
            Item item = ((CraftingTable)myblock).ReturnShapedRecipieItem(items);
            if (item != base.items.itemsInInventory[9])
                base.items.itemsInInventory[9] = item;
        }

        public virtual void CheckShapelessRecipie()
        {
            var items = new Item[9];

            for (int i = 0; i < items.Length; i++)
            {
                items[i] = base.items.itemsInInventory[i];
            }

            Item item = ((CraftingTable)myblock).ReturnShapelessRecipieItem(items);
            if (item != base.items.itemsInInventory[9])
                base.items.itemsInInventory[9] = item;
        }

        public void CraftedItemRemoved()
        {
            if (items.itemsInInventory[9] != null)
            {
                Events.CallShapedRecipieCraftedEvent(items.
itemsInInventory[9]);
                for (int i = 0; i < 9; i++)
                {
                    if (items.itemsInInventory[i] != null)
                        items.itemsInInventory[i].itemStackCount -= 1;
                }
            }
        }
        #endregion

        #region Inventory Stuff
        public virtual void DropItemsFromInventory()
        {
            //* looks at every item in the crafting grid
            for (int i = 0; i < 9; i++)
            {
                if (items.itemsInInventory[i] != null)
                {
                    //* spwns it and removes it from the inventory if an items exists within
                    for (int j = 0; j < items.itemsInInventory[i].itemStackCount; j++)
                    {
                        items.itemsInInventory[i].SpawnItem((THVector3)this.transform.position +
new THVector3(0, 1, 0));
                    }
                    items.itemsInInventory[i] = null;
                }
            }
        }
        #endregion

        #region Overrides
        public override void ToggleInventory(Inventory inv)
        {
            base.ToggleInventory(inv);

            //* if the inventory was closed drop the items within
            if (!inventory.activeInHierarchy)
               DropItemsFromInventory();
        }

        public override void SetChestInventory(string invName = "Workbench")
        {
            SetInventorySize(inventorySize);
            //* sets the UI to not be seen as inventorys cannot start open
            inventory.SetActive(false);
        }

        public override void AddItemToSlots(int slotIndex, Item item)
        {
            items.AddItem(slotIndex, item);
        }

        public override void SaveInv()
        {
            //* does not need to be saved so overrided to do nothing
        }
        #endregion
    }
}
```

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

Classes

- class BeeGame.Inventory.Inventory

  Base class for all inventorys in the game

Namespaces

- namespace BeeGame.Inventory

7.56 Inventory.cs

```
using System;
using UnityEngine;
using BeeGame.Items;
using BeeGame.Core.Dictionarys;

namespace BeeGame.Inventory
{
    public class Inventory : MonoBehaviour
    {
        #region Data
        public ItemsInInventory items;
        public InventorySlot[] slots;
        internal Item floatingItem;
        public string inventoryName = "";
        protected bool thisInventoryOpen = false;
        private GameObject spriteAtCursor;
        public Blocks.Block myblock;
        #endregion

        #region Init
        public bool InventorySet()
        {
            if (items == null)
                return true;

            return false;
        }

        public void SetInventorySize(int inventorySize)
        {
            items = new ItemsInInventory(slots.Length);
        }

        public void SetAllItems(ItemsInInventory items)
        {
            this.items = items;
        }
        #endregion

        #region Update
        public void UpdateBase()
        {
            PutItemsInSlots();
            DrawItemAtCursor();
        }

        private void DrawItemAtCursor()
        {
            if(floatingItem != null)
            {
                if (spriteAtCursor == null)
                {
                    spriteAtCursor = Instantiate(PrefabDictionary.GetPrefab("ItemIcon"));
                    spriteAtCursor.GetComponentInChildren<UnityEngine.UI.Image>().sprite = floatingItem.GetItemSprite();
                }
                //* will update a the sprite of in item is swapped between a slot and teh floating item if the previous item wasnt put into a slot first
                else if(spriteAtCursor != null)
                {
                    spriteAtCursor.GetComponentInChildren<UnityEngine.UI.Image>().sprite = floatingItem.GetItemSprite();
                }

                spriteAtCursor.transform.GetChild(0).position = Input.mousePosition;
            }
            else
            {
                Destroy(spriteAtCursor);
            }
        }
        #endregion

        #region Edit Inventory
        public virtual void ToggleInventory(Inventory inv)
        {
            throw new NotImplementedException();
        }

        public virtual void SaveInv()
        {
            Serialization.Serialization.SerializeInventory(this, inventoryName);
        }

        void PutItemsInSlots()
        {
            //* goes through all of the items in the array setting then all to a slot
            for (int i = 0; i < slots.Length; i++)
            {
                slots[i].slotIndex = i;
                slots[i].myInventory = this;
                slots[i].item = items.itemsInInventory[i];
            }
        }

        public ItemsInInventory GetAllItems()
        {
            return items;
        }

        public virtual void AddItemToSlots(int slotIndex, Item item)
        {
            items.AddItem(slotIndex, item);
            //* saves the inventory changes
            Serialization.Serialization.SerializeInventory(this, inventoryName);
        }

        public bool AddItemToInventory(Item item)
        {
            return items.AddItem(item);
        }
        #endregion
    }
}
```

7.57 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Inventory←Slot.cs File Reference

Classes

- class BeeGame.Inventory.InventorySlot

Namespaces

- namespace BeeGame.Inventory

7.58 InventorySlot.cs

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;
using BeeGame.Items;
using BeeGame.Core;
using BeeGame.Core.Dictionarys;

namespace BeeGame.Inventory
{
    public class InventorySlot : MonoBehaviour, IPointerClickHandler, IPointerEnterHandler,
IPointerExitHandler
    {
        #region Data
        internal int slotIndex;
        public Item item;
        public Inventory myInventory;
        public GameObject itemText;
        public bool selectedSlot = false;
        public bool itemsCanBeInserted = true;
        #endregion

        private void Update()
        {
            CheckItem();
            UpdateIcon();
        }


        void UpdateIcon()
        {
            if(item == null)
            {
                GetComponent<Image>().sprite = null;
            }
            else
            {
                if(!item.Equals(new Item()))
                    GetComponent<Image>().sprite = item.GetItemSprite();
            }

            //* if the slot is selected in the hotbar give the player some indication by colouring it grey
            if (selectedSlot)
            {
                GetComponent<Image>().color = Color.gray;
            }
            else
            {
                GetComponent<Image>().color = Color.white;
            }
        }

        #region Interact With Slot
        public void OnPointerClick(PointerEventData eventData)
        {
            if (myInventory.floatingItem != null)
            {
                //* Left click moves whole stacks of items
                if (eventData.button == PointerEventData.InputButton.Left)
                {
                    //* If the item in the slot is empty put the floating item into it then clear it and the slot can have items inserted
                    if (item == null && itemsCanBeInserted)
                    {
                        item = myInventory.floatingItem;
                        myInventory.floatingItem = null;
                        myInventory.AddItemToSlots(slotIndex, item);
                        return;
                    }
                    //* if the items are the same
                    if(myInventory.floatingItem == item && itemsCanBeInserted)
                    {
                        //* if the item in the inventoys stack count + the floating items stack count is less than the max stack count
                        if (myInventory.floatingItem.itemStackCount + item.itemStackCount <= item.maxStackCount)
                        {
                            AddToSlot(myInventory.floatingItem.itemStackCount);
                            return;
                        }
                        //* if the item stack added is larger than the max count add as many as you can and move on
                        else
                        {
                            AddToSlot(item.maxStackCount - item.itemStackCount);
                            return;
                        }
                    }
                    //* if the tiems are the same but items cannot be inserted into the slot add as many items as you
                    //* can from the slot to the floating item
                    else if(myInventory.floatingItem == item && !itemsCanBeInserted)
                    {
                        AddToFloatingItem();
                        {
                            if (myInventory is BlockInventory.CraftingTableInventory c)
                                c.result.Invoke();
                        }
                        return;
                    }
                    //* If the items were not == swap them
                    else
                    {
                        //* only if items can be inserted into the slot
                        if(itemsCanBeInserted)
                            SwapItems();
                        return;
                    }
                }
                else if(eventData.button == PointerEventData.InputButton.Right)
                {
                    //* if the item in slot is null add 1 from the floating item to it
                    if(item == null && itemsCanBeInserted)
                    {
                        AddToSlot(1);
                        return;
                    }
                    //* if the items are the same add 1 from the floating item to this item
                    else if(item == myInventory.floatingItem && itemsCanBeInserted)
                    {
                        AddToSlot(1);
                        return;
                    }
                }
            }
            //* if the floating item is null
            else
            {
                //* add 1/2 of the stack into the floating item if right click was pressed
                if(eventData.button == PointerEventData.InputButton.Right)
                {
                    SplitStack();

                    //* blocks removed some weird name confliction
                    {
                        if (myInventory is BlockInventory.CraftingTableInventory c)
                            c.result.Invoke();
                    }

                    return;
                }

                //* otherwie add the items into the floating item slot
                SwapItems();
                //* ^ does not need to check that the slot cannot be inserted into as null be being inserted because the floating item is null

                {
                    if (myInventory is BlockInventory.CraftingTableInventory c)
                        c.result.Invoke();
                }

                return;
            }

        }

        void AddToFloatingItem()
        {
            //* if the whole stack can be added do it and move on
            if(myInventory.floatingItem.itemStackCount + item.itemStackCount <= item.maxStackCount)
            {
                myInventory.floatingItem.itemStackCount += item.itemStackCount;

                item = null;

                myInventory.AddItemToSlots(slotIndex, item);

                return;
            }

            //* if the whole stack cannot be added calculate how many need to be removed from the slots item stack
            item.itemStackCount -= (item.maxStackCount - myInventory.floatingItem.itemStackCount);
            //* set the floating item to the max stack count
            myInventory.floatingItem.itemStackCount = item.maxStackCount;

            myInventory.AddItemToSlots(slotIndex, item);
        }

        void AddToSlot(int numerToAdd)
        {
            //* if the item in the slot is null create it
            if (item == null)
            {
                item = myInventory.floatingItem.CloneObject();
                item.itemStackCount = 0;
            }

            //* add to number to add to the stack count
            item.itemStackCount += numerToAdd;

            //* if the stack count is now larger than it should be dont let it be
            if (item.itemStackCount > item.maxStackCount)
            {
                item.itemStackCount = item.maxStackCount;
            }

            //* remove the numebr if items form the floating item then check the floating item is not null
            myInventory.floatingItem.itemStackCount -= numerToAdd;
            CheckFloatingItem();
            //* save the inventory changes
            myInventory.AddItemToSlots(slotIndex, item);
        }

        void SplitStack()
        {
            myInventory.floatingItem = item.CloneObject();
            int give = (item.itemStackCount + 1) / 2;
            myInventory.floatingItem.itemStackCount = give;
            item.itemStackCount -= give;

            if (item.itemStackCount <= 0)
                item = null;

            myInventory.AddItemToSlots(slotIndex, item);
            Destroy(itemText);
        }

        void SwapItems()
        {
            //* temp copy of the item
            Item temp = myInventory.floatingItem;
            //* sets the floating item
            myInventory.floatingItem = item;
            //* sets the item that was in the floating item to the item in the the slot
            item = temp;
            //* Saves the changes to the inventory
            myInventory.AddItemToSlots(slotIndex, item);
            //* destroys the text as it is not needed anymore
            Destroy(itemText);
        }

        void CheckFloatingItem()
        {
            if(myInventory.floatingItem.itemStackCount <= 0)
            {
                myInventory.floatingItem = null;
            }
        }
        #endregion

        private void CheckItem()
        {
            if (item != null && myInventory != null)
            {
                if (item.itemStackCount == 0 || item.itemName == "TestItem")
                {
                    myInventory.items.itemsInInventory[slotIndex] = null;
                    Destroy(itemText);
                }
            }
        }

        #region Display Item On Hover
        public void OnPointerEnter(PointerEventData eventData)
        {
            //* if the item is null or the floating item has something in it dont display the item text as it is not necissary
            if (item != null && myInventory.floatingItem == null)
            {
                itemText = Instantiate(PrefabDictionary.GetPrefab("ItemDetails"));
                //* sets the text to the correct postion
                itemText.transform.GetChild(0).position = Input.mousePosition;
                //* puts the correct text in the box
                itemText.transform.GetChild(0).GetChild(0).GetComponent<Text>().text = $"{item.GetItemName()}\nStack: {item.itemStackCount}";
            }
        }

        public void OnPointerExit(PointerEventData eventData)
        {
            Destroy(itemText);
        }

        void OnDisable()
        {
            Destroy(itemText);
        }
        #endregion
    }
}
```

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

Classes

- class BeeGame.Inventory.ItemsInInventory

  Class that holds all of the items in the inventory. Can be serialized so inventory may be saved

Namespaces

- namespace BeeGame.Inventory

7.60 ItemsInInventory.cs

```
using System;
using BeeGame.Items;

namespace BeeGame.Inventory
{
    [Serializable]
    public class ItemsInInventory
    {
        public Item[] itemsInInventory;

        public ItemsInInventory(int numberOfInventorySlots)
        {
            itemsInInventory = new Item[numberOfInventorySlots];
        }

        public void AddItem(int index, Item item)
        {
            itemsInInventory[index] = item;
        }

        public bool AddItem(Item item)
        {
            for (int i = 0; i < itemsInInventory.Length; i++)
            {
                if (itemsInInventory[i] == null)
                {
                    itemsInInventory[i] = item;
                    return true;
                }
                if (itemsInInventory[i] == item && itemsInInventory[i].itemStackCount + 1 <= itemsInInventory[i].maxStackCount)
                {
                    itemsInInventory[i].itemStackCount++;
                    return true;
                }
            }

            return false;
        }
    }
}
```

7.61 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Inventory/Player Inventory/PlayerInventory.cs File Reference

Classes

- class BeeGame.Inventory.Player_Inventory.PlayerInventory

  Controlls the player inventory

Namespaces

- namespace BeeGame.Inventory.Player_Inventory

7.62 PlayerInventory.cs

```
using UnityEngine;
using BeeGame.Items;
using BeeGame.Core;

namespace BeeGame.Inventory.Player_Inventory
{
    public class PlayerInventory : Inventory
    {
        #region Data
        public GameObject playerInventory;
        #endregion

        #region Init
        void Awake()
        {
            SetPlayerInventory();
            inventoryName = "PlayerInventory";
            Serialization.Serialization.DeSerializeInventory(this, inventoryName);
        }

        void SetPlayerInventory()
        {
            if (!InventorySet())
                SetInventorySize(36);
        }
        #endregion

        void Update()
        {
            UpdateBase();

            //* checks if the inventory should be opened/closed
            if ((thisInventoryOpen || !playerInventory.activeInHierarchy) && !THInput.chestOpen && THInput.GetButtonDown("Player Inventory"))
            {
                if (THInput.blockInventoryJustClosed)
                {
                    THInput.blockInventoryJustClosed = false;
                    return;
                }
                else
                {
                    OpenPlayerInventory();
                }
            }

            //* dont pickup items if the inventory is open
            if (THInput.isAnotherInventoryOpen)
                return;

            //* checks if somethig should be picked up and put into the inventory
            RaycastHit[] hit = Physics.SphereCastAll(transform.position, 1f, transform.forward);

            for (int i = hit.Length - 1; i >= 0; i--)
            {
                if (hit[i].collider.GetComponent<ItemGameObject>())
                    PickupItem(hit[i].collider.GetComponent<ItemGameObject>());
            }

        }

        #region Hotbar
        public void SelectedSlot(int index)
        {
            for (int i = 0; i < slots.Length; i++)
            {
                slots[i].selectedSlot = false;
            }

            slots[index].selectedSlot = true;
        }

        public bool GetItemFromHotBar(int slotIndex, out Item outItem)
        {
            //* get the item
            outItem = GetAllItems().itemsInInventory[slotIndex];

            if (outItem == null)
                return false;

            //* if the item is placebale and is not null remove 1 from the inventory as it is assumed it is about to be placed in the world
            if(outItem.placeable)
                RemoveItemFromInventory(slotIndex);

            return outItem.placeable;
        }
        #endregion

        #region Interact With Inventory
        void OpenPlayerInventory()
        {
            if (floatingItem != null)
                return;
            thisInventoryOpen = !thisInventoryOpen;
            playerInventory.SetActive(!playerInventory.activeInHierarchy);
            THInput.isAnotherInventoryOpen = !THInput.isAnotherInventoryOpen;

            //* hides/shows the mouse depending on if te inventory is open or not
            if (playerInventory.activeInHierarchy)
            {
                Cursor.lockState = CursorLockMode.None;
                Cursor.visible = true;
            }
            else
            {
                Cursor.visible = false;
                Cursor.lockState = CursorLockMode.Locked;
            }
        }

        public void RemoveItemFromInventory(int index)
        {
            //* if the item is already null nothign needs to be removed
            if (GetAllItems().itemsInInventory[index] != null)
            {
                //* remove 1 item and if that was the last in the stack remove the item from the inventory
                GetAllItems().itemsInInventory[index].itemStackCount -= 1;

                if (GetAllItems().itemsInInventory[index].itemStackCount <= 0)
                    GetAllItems().itemsInInventory[index] = null;

                Serialization.Serialization.SerializeInventory(this, inventoryName);
            }
        }

        void PickupItem(ItemGameObject item)
        {
            item.item.itemStackCount = 1;

            //* if the item can be added to the inventory do that
            if (AddItemToInventory(item.item))
            {
                //* if the item was added destroyits gameobject and save the inventory
                Destroy(item.gameObject);
                Serialization.Serialization.SerializeInventory(this, inventoryName);
            }
        }
        #endregion
    }
}
```

7.63 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/AbstractItem.cs File Reference

Classes

- class BeeGame.Items.AbstractItem

  Does this need to exist?

Namespaces

- namespace BeeGame.Items

7.64 AbstractItem.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace BeeGame.Items
{
    [Serializable]
    public abstract class AbstractItem
    {
        public abstract string GetItemName();
        public abstract string GetItemID();
        public abstract override int GetHashCode();
    }
}
```

7.65 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/ApplyColour.cs File Reference

Classes

- class BeeGame.Items.ApplyColour

    Applies a given colour to a gameobject

Namespaces

- namespace BeeGame.Items

7.66 ApplyColour.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using UnityEngine;

namespace BeeGame.Items
{
    public class ApplyColour : MonoBehaviour
    {
        #region Data
        public Color colour;
        public GameObject[] objects;
        #endregion

        #region Unity Methods
        private void Start()
        {
            //* applies the correct colour to each object in the array
            for (int i = 0; i < objects.Length; i++)
            {
                objects[i].GetComponent<Renderer>().material.SetColor("_OverlayColour", colour);
            }
        }
        #endregion
    }
}
```

7.67 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/Bee.cs File Reference

Classes

- class BeeGame.Items.Bee

    The bee item
- class BeeGame.Items.QueenBee
- class BeeGame.Items.NormalBee

Namespaces

- namespace BeeGame.Items

7.68 Bee.cs

```
using System;
using System.Globalization;
using UnityEngine;
using BeeGame.Core;
using BeeGame.Core.Enums;
using BeeGame.Core.Dictionarys;

namespace BeeGame.Items
{
    [Serializable]
    public class Bee : Item
    {
        #region Data
        public bool canSeeBeeData = false;

        public BeeType beeType { get; set; }
        private BeeType previousBeeType { get; set; }
        public override int maxStackCount { get { return maxStack; } }
        private int maxStack = 64;

        [NonSerialized]
        private Sprite itemSprite;

        public QueenBee queenBee { get; set; }
        public NormalBee normalBee { get; set; }

        public new static int ID => 11;
        #endregion

        #region Constructors
        public Bee()
        {
            normalBee = new NormalBee();
        }


        public Bee(BeeType beeType, NormalBee normalBee) : base(new CultureInfo("en-US", false).TextInfo.ToTitleCase($"{normalBee.pSpecies} {beeType}".ToLower()))
        {
            if (beeType == BeeType.PRINCESS || beeType == BeeType.QUEEN)
                maxStack = 1;
            this.beeType = beeType;
            this.normalBee = normalBee;
        }

        public Bee(BeeType beeType, QueenBee queenBee) : base(new CultureInfo("en-US", false).TextInfo.ToTitleCase($"{queenBee.queen.pSpecies} {beeType}".ToLower()))
        {
            if (beeType == BeeType.PRINCESS || beeType == BeeType.QUEEN)
                maxStack = 1;
            this.beeType = beeType;
            this.queenBee = queenBee;
        }
        #endregion

        #region Item Overrides
        public override Sprite GetItemSprite()
        {
            //* if the bee has not change in any way dont rebuild the sprite as that takes time
            if(previousBeeType == beeType && itemSprite != null)
            {
                return itemSprite;
            }

            previousBeeType = beeType;

            //* set the correct sprite and colour
            if (beeType == BeeType.QUEEN)
            {
                //* avoids the crown, black body, yellow body, and both colours of the wings
                Color[] colorsToAvoid = { new Color(0, 0, 0), new Color(232f, 200f, 42f, 255f) / 255f, new Color(232f, 213f, 106f, 255f) / 255f, new Color(156f, 146f, 130f, 255f) / 255f, new Color(225f, 223f, 219f, 255f) / 255f };
                return itemSprite = SpriteDictionary.GetSprite("Queen").ColourSprite(BeeDictionarys.GetBeeColour((BeeSpecies)(queenBee?.queen.pSpecies)), coloursToAvoid: colorsToAvoid);
            }
            else if (beeType == BeeType.PRINCESS)
            {
                //* avoids the tiara, black body, yellow body, and both colours of the wings
                Color[] colorsToAvoid = { new Color(0, 0, 0), new Color(191f, 195f, 45f, 255f) / 255f, new Color(191f, 195f, 44f, 255f) / 255f, new Color(156f, 146f, 130f, 255f) / 255f, new Color(225f, 223f, 219f, 255f) / 255f, new Color(232f, 200, 42, 255f) / 255f };
                return itemSprite = SpriteDictionary.GetSprite("Princess").
ColourSprite(BeeDictionarys.GetBeeColour((BeeSpecies)(normalBee?.pSpecies)),
coloursToAvoid: colorsToAvoid);
            }
            else
            {
                //* avoids the block body, yellow body, and both wing colours
                Color[] colorsToAvoid = { new Color(0, 0, 0), new Color(156f, 146f, 130f, 255f) / 255f, new
 Color(225f, 223f, 219f, 255f) / 255f, new Color(232f, 200, 42, 255f) / 255f };
                return itemSprite = SpriteDictionary.GetSprite("Drone").
ColourSprite(BeeDictionarys.GetBeeColour((BeeSpecies)normalBee?.pSpecies),
coloursToAvoid: colorsToAvoid);
            }
        }

        public override string GetItemID()
        {
            return $"{GetHashCode()}\\{(int)beeType}{queenBee?.GetHashCode() ?? normalBee?.GetHashCode()}";
        }
        #endregion

        #region Bee Stuff
        public static void ConvertToQueen(Bee princess, NormalBee drone)
        {
            ConvertToQueen(ref princess, drone);
        }

        public static void ConvertToQueen(ref Bee princess,
NormalBee drone)
        {
            princess.beeType = BeeType.QUEEN;
            princess.queenBee = new QueenBee(princess.normalBee, drone);
            princess.normalBee = null;

            princess.itemName = new CultureInfo("en-US", false).TextInfo.ToTitleCase($"
{princess.queenBee.queen.pSpecies} {princess.beeType}".ToLower());
        }

        public Bee MakeBeeWithStats(BeeType beeType =
BeeType.DRONE, BeeSpecies species = BeeSpecies.FOREST,
BeeLifeSpan lifespan = BeeLifeSpan.NORMAL, uint fertility = 2,
BeeEffect effect = BeeEffect.NONE, BeeProductionSpeed prodSpeed =
BeeProductionSpeed.NORMAL)
        {
            NormalBee normBee = new NormalBee()
            {
                pSpecies = species,
                pLifespan = lifespan,
                pFertility = fertility,
                pProdSpeed = prodSpeed,
                pEffect = effect,
                sEffect = effect,
                sFertility = fertility,
                sLifespan = lifespan,
                sProdSpeed = prodSpeed,
                sSpecies = species
            };

            switch (beeType)
            {
                case BeeType.QUEEN:
                    return new Bee(beeType, new QueenBee(normBee, normBee));
                default:
                    return new Bee(beeType, normBee);
            }
        }
        #endregion

        #region Overrides
        public override int GetHashCode()
        {
            return ID;
        }
        #endregion
    }

    [Serializable]
    public class QueenBee
    {
        //* Properties so that they can be copied by reflection as it does not copy variables only
 properties
        public NormalBee queen { get; set; }
        public NormalBee drone { get; set; }

        public QueenBee() { }

        public QueenBee(NormalBee princess, NormalBee drone)
        {
            this.queen = princess;
            this.drone = drone;
        }

        public override int GetHashCode()
        {
            unchecked
            {
                return (int)Int64.Parse($"{queen.GetHashCode()}{drone.GetHashCode()}");
            }
        }
    }

    [Serializable]
    public class NormalBee
    {
        #region Phenotype
        //* Currently shown traits of the bee

        public BeeSpecies pSpecies;
        public BeeLifeSpan pLifespan;
        public uint pFertility;
        public BeeEffect pEffect;
        public BeeProductionSpeed pProdSpeed;
        #endregion

        #region Secondary
        //* Traits of the bee used in the bees combination

        public BeeSpecies sSpecies;
        public BeeLifeSpan sLifespan;
        public uint sFertility;
        public BeeEffect sEffect;
        public BeeProductionSpeed sProdSpeed;
        #endregion Secondary

        public override int GetHashCode()
        {
            unchecked
            {
                //int hashcode = 13;

                var temp = $"
{(int)pSpecies}{(int)sSpecies}{(int)pLifespan}{(int)sLifespan}{(int)pFertility}{(int)sFertility}{(int)pEffect}{(int)sEf

                var hashcode = (int)(Int64.Parse(temp) ^ (127 * 13) / 159);

                //hashcode += ((int)pSpecies ^ (int)pLifespan ^ (int)pFertility ^ (int)pEffect ^
 (int)pProdSpeed) * 127;
                //hashcode += ((int)sSpecies ^ (int)sLifespan ^ (int)sFertility ^ (int)sEffect ^
 (int)sProdSpeed) * 307;

                return hashcode;
            }
        }
    }
}
```

7.69 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/HoneyComb.cs File Reference

Classes

- class BeeGame.Items.HoneyComb

    Honey comb item produced by bees

Namespaces

- namespace BeeGame.Items

7.70 HoneyComb.cs

```
using System;
using System.Globalization;
using BeeGame.Core;
using BeeGame.Core.Enums;
using BeeGame.Core.Dictionarys;
using UnityEngine;

namespace BeeGame.Items
{
    [Serializable]
    public class HoneyComb : Item
    {
        #region Data
        public HoneyCombType type { get; set; }

        public Color CombColour
        {
            get
            {
                return BeeDictionarys.GetCombColour(type);
            }
        }

        [NonSerialized]
        private Sprite itemSprite;

        public new static int ID => 12;
        #endregion

        #region Constructors
        public HoneyComb() : base(new CultureInfo("en-US", false).TextInfo.ToTitleCase($"
{HoneyCombType.HONEY} Comb".ToLower()))
        {
            usesGameObject = true;
            type = HoneyCombType.HONEY;
        }

        public HoneyComb(HoneyCombType type) : base(new CultureInfo("en-US", false).
TextInfo.ToTitleCase($"{type.ToString()} Comb".ToLower()))
        {
            usesGameObject = true;
            this.type = type;
        }
        #endregion

        #region Item Overrides
        public override Sprite GetItemSprite()
        {
            return itemSprite ?? (itemSprite = SpriteDictionary.
GetSprite("HoneyComb").ColourSprite(CombColour));
        }

        public override GameObject GetGameObject()
        {
            GameObject obj = PrefabDictionary.GetPrefab("HoneyComb");
            //* cannot acess the instance material from here have to do it on the obejct
            obj.GetComponent<ApplyColour>().colour = CombColour;
            return obj;
        }

        public override string GetItemID()
        {
            return $"{GetHashCode()}\\{(int)type}";
        }
        #endregion

        #region Overrides
        public override int GetHashCode()
        {
            return ID;
        }
        #endregion
    }
}
```

7.71 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/Item.cs File Reference

Classes

- class BeeGame.Items.Item

Base class for all Items and Blocks in the game

- struct BeeGame.Items.Tile

  Position of the items texture

Namespaces

- namespace BeeGame.Items

7.72 Item.cs

```
using System;
using System.IO;
using UnityEngine;
using BeeGame.Core;
using BeeGame.Core.Enums;
using BeeGame.Terrain.Chunks;
using BeeGame.Core.Dictionarys;
using System.Runtime.Serialization.Formatters.Binary;

namespace BeeGame.Items
{
    [Serializable]
    public class Item : AbstractItem, ICloneable
    {
        #region Data
        internal string itemName { get; set;}
        public virtual bool placeable => false;
        public bool usesGameObject { get; set; }
        private const float tileSize = 0.1f;

        public int itemStackCount { set { count = value; } get{ return count; } }
        private int count = 1;

        public virtual int maxStackCount => 64;

        public static int ID => 0;
        #endregion

        #region Constructors
        public Item()
        {
            itemName = "TestItem";
        }

        public Item(string name)
        {
            itemName = name;
        }
        #endregion

        #region Player Item Interactions
        public virtual bool InteractWithObject()
        {
            return false;
        }
        #endregion

        #region Item Stuff
        public virtual GameObject GetGameObject() { return null; }

        public override string GetItemID()
        {
            return $"{GetHashCode()}";
        }

        public virtual Sprite GetItemSprite()
        {
            return SpriteDictionary.GetSprite("TestSprite");
        }

        public override string GetItemName()
        {
            return $"{itemName}";
        }
        #endregion

        #region Item Mesh
        public virtual Tile TexturePosition(Direction direction)
        {
            return new Tile() { x = 1, y = 9 };
        }

        public virtual MeshData ItemMesh(int x, int y, int z, MeshData meshData)
        {
            //* adds all faces of the item to the mesh as all faces could be seen at any time
            meshData = FaceDataUp(x, y, z, meshData, true, 0.25f);
            meshData = FaceDataDown(x, y, z, meshData, true, 0.25f);
            meshData = FaceDataNorth(x, y, z, meshData, true, 0.25f);
            meshData = FaceDataEast(x, y, z, meshData, true, 0.25f);
            meshData = FaceDataSouth(x, y, z, meshData, true, 0.25f);
            meshData = FaceDataWest(x, y, z, meshData, true, 0.25f);

            return meshData;
        }

        public virtual Vector2[] FaceUVs(Direction direction)
        {
            //* only 4 uvs per face
            Vector2[] UVs = new Vector2[4];
            Tile tilePos = TexturePosition(direction);

            //* sets the UVs for each vertex
            UVs[0] = new THVector2(tileSize * tilePos.x + tileSize - 0.01f, tileSize * tilePos.y + 0.01f);
            UVs[1] = new THVector2(tileSize * tilePos.x + tileSize - 0.01f, tileSize * tilePos.y + tileSize - 0.01f);
            UVs[2] = new THVector2(tileSize * tilePos.x + 0.01f, tileSize * tilePos.y + tileSize - 0.01f);
            UVs[3] = new THVector2(tileSize * tilePos.x + 0.01f, tileSize * tilePos.y + 0.01f);

            return UVs;
        }

        protected virtual MeshData FaceDataUp(int x, int y, int z, MeshData meshData, bool addToRenderMesh = true, float blockSize = 0.5f)
        {
            //* Adds vertices in a anti-clockwise order
            meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z + blockSize), addToRenderMesh, Direction.UP);
            meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z + blockSize), addToRenderMesh, Direction.UP);
            meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z - blockSize), addToRenderMesh, Direction.UP);
            meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z - blockSize), addToRenderMesh, Direction.UP);

            //* adds teh tirs for the quad
            meshData.AddQuadTriangles(addToRenderMesh);

            //* if the data should be added to the render mesh also add the uvs to the mesh
            if (addToRenderMesh)
                meshData.uv.AddRange(FaceUVs(Direction.UP));

            return meshData;
        }

        protected virtual MeshData FaceDataDown(int x, int y, int z, MeshData meshData, bool addToRenderMesh = true, float blockSize = 0.5f)
        {
            //* Adds vertices in a anti-clockwise order
            meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z - blockSize), addToRenderMesh);
            meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z - blockSize), addToRenderMesh);
            meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z + blockSize), addToRenderMesh);
            meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z + blockSize), addToRenderMesh);

            //* adds teh tirs for the quad
            meshData.AddQuadTriangles(addToRenderMesh);

            //* if the data should be added to the render mesh also add the uvs to the mesh
            if (addToRenderMesh)
                meshData.uv.AddRange(FaceUVs(Direction.DOWN));

            return meshData;
        }

        protected virtual MeshData FaceDataNorth(int x, int y, int z, MeshData meshData, bool addToRenderMesh = true, float blockSize = 0.5f)
        {
            //* Adds vertices in a anti-clockwise order
            meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z + blockSize), addToRenderMesh);
            meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z + blockSize), addToRenderMesh);
            meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z + blockSize), addToRenderMesh);
            meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z + blockSize), addToRenderMesh);

            //* adds teh tirs for the quad
            meshData.AddQuadTriangles(addToRenderMesh);

            //* if the data should be added to the render mesh also add the uvs to the mesh
            if (addToRenderMesh)
                meshData.uv.AddRange(FaceUVs(Direction.NORTH));

            return meshData;
        }

        protected virtual MeshData FaceDataEast(int x, int y, int z, MeshData meshData, bool addToRenderMesh = true, float blockSize = 0.5f)
        {
            //* Adds vertices in a anti-clockwise order
            meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z - blockSize), addToRenderMesh);
            meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z - blockSize), addToRenderMesh);
            meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z + blockSize), addToRenderMesh);
            meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z + blockSize), addToRenderMesh);

            //* adds teh tirs for the quad
            meshData.AddQuadTriangles(addToRenderMesh);

            //* if the data should be added to the render mesh also add the uvs to the mesh
            if (addToRenderMesh)
                meshData.uv.AddRange(FaceUVs(Direction.EAST));

            return meshData;
        }

        protected virtual MeshData FaceDataSouth(int x, int y, int z, MeshData meshData, bool addToRenderMesh = true, float blockSize = 0.5f)
        {
            //* Adds vertices in a anti-clockwise order
            meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z - blockSize), addToRenderMesh);
            meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z - blockSize), addToRenderMesh);
            meshData.AddVertices(new THVector3(x + blockSize, y + blockSize, z - blockSize), addToRenderMesh);
            meshData.AddVertices(new THVector3(x + blockSize, y - blockSize, z - blockSize), addToRenderMesh);

            //* adds teh tirs for the quad
            meshData.AddQuadTriangles(addToRenderMesh);

            //* if the data should be added to the render mesh also add the uvs to the mesh
            if (addToRenderMesh)
                meshData.uv.AddRange(FaceUVs(Direction.SOUTH));

            return meshData;
        }

        protected virtual MeshData FaceDataWest(int x, int y, int z, MeshData meshData, bool addToRenderMesh = true, float blockSize = 0.5f)
        {
            //* Adds vertices in a anti-clockwise order
            meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z + blockSize), addToRenderMesh);
            meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z + blockSize), addToRenderMesh);
            meshData.AddVertices(new THVector3(x - blockSize, y + blockSize, z - blockSize), addToRenderMesh);
            meshData.AddVertices(new THVector3(x - blockSize, y - blockSize, z - blockSize), addToRenderMesh);

            //* adds teh tirs for the quad
            meshData.AddQuadTriangles(addToRenderMesh);

            //* if the data should be added to the render mesh also add the uvs to the mesh
            if (addToRenderMesh)
                meshData.uv.AddRange(FaceUVs(Direction.WEST));

            return meshData;
        }
        #endregion

        #region Interfaces
        public object Clone()
        {
            //* Saves this to a file then reads it back so that a copy and not a reference is passed
            BinaryFormatter bf = new BinaryFormatter();
            MemoryStream ms = new MemoryStream();

            bf.Serialize(ms, this);
            ms.Seek(0, SeekOrigin.Begin);

            return bf.Deserialize(ms);
        }
        #endregion

        #region Overrides
        public override string ToString()
        {
            return $"{itemName} \nID: {GetItemID()}";
        }

        public override int GetHashCode()
        {
            return ID;
        }

        public override bool Equals(object obj)
        {
            if (!(obj is Item))
                return false;

            return this == (obj as Item);
        }

        public static bool operator ==(Item a, Item b)
        {
            if (ReferenceEquals(a, null) && ReferenceEquals(b, null))
                return true;
            if (ReferenceEquals(a, null) || ReferenceEquals(b, null))
                return false;

            if(a.GetItemID() == b.GetItemID())
                return true;

            return false;
        }

        public static bool operator !=(Item a, Item b)
        {
            return !(a == b);
        }
        #endregion
    }

    [Serializable]
    public struct Tile
    {
        public int x;
        public int y;
    }
}
```

7.73 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Items/ItemGameObject.cs File Reference

Classes

- class BeeGame.Items.ItemGameObject

  Interface between item and inity gameobjects

Namespaces

- namespace BeeGame.Items

7.74 ItemGameObject.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using BeeGame.Terrain.Chunks;
using BeeGame.Blocks;
using UnityEngine;

namespace BeeGame.Items
{
    [RequireComponent(typeof(Rigidbody))]
    [RequireComponent(typeof(MeshFilter))]
    [RequireComponent(typeof(MeshRenderer))]
    [RequireComponent(typeof(BoxCollider))]
    public class ItemGameObject : MonoBehaviour
    {
        public Item item;
        public GameObject go;

        private void Start()
        {
            if (!item.usesGameObject)
                MakeMesh();

            if (item.usesGameObject)
            {
                Instantiate(item.GetGameObject(), transform, false);
                transform.localScale = new Vector3(0.5f, 0.5f, 0.5f);
            }
        }

        private void Update()
        {
           if(transform.position.y < -100)
            {
                Destroy(gameObject);
            }
        }

        void MakeMesh()
        {
            MeshData meshData = new MeshData();
            if(item != null)
                meshData = item.ItemMesh(0, 0, 0, meshData);

            Mesh mesh = new Mesh()
            {
                vertices = meshData.verts.ToArray(),
                triangles = meshData.tris.ToArray(),
                uv = meshData.uv.ToArray()
            };

            mesh.RecalculateNormals();

            GetComponent<MeshFilter>().mesh = mesh;
        }
    }
}
```

7.75 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/LoadResources.cs File Reference

Classes

- class BeeGame.LoadResources

    Loads all of the resources in the game

Namespaces

- namespace BeeGame

7.76 LoadResources.cs

```
using UnityEngine;
using BeeGame.Core.Dictionarys;

namespace BeeGame
{
    public class LoadResources : MonoBehaviour
    {
        void Awake()
        {
            Serialization.Serialization.MakeDirectorys();

            Serialization.Serialization.LoadPlayerPosition(GameObject.Find("Player").GetComponent<Transform
>());

            SpriteDictionary.LoadSprites();
            PrefabDictionary.LoadPrefabs();
        }
    }
}
```

7.77 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/obj/Debug/Temporary←GeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs File Reference

7.78 TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs

7.79 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/obj/Debug/Temporary←GeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs File Reference

7.80 TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs

7.81 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/obj/Debug/Temporary←GeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs File Reference

7.82 TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs

7.83 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/PlayerLook.cs File Reference

Classes

- class BeeGame.Player.PlayerLook

  The look for the player

Namespaces

- namespace BeeGame.Player

7.84 PlayerLook.cs

```
using UnityEngine;
using BeeGame.Core;

namespace BeeGame.Player
{
    public class PlayerLook : MonoBehaviour
    {
        #region Data
        public Transform myTransform;
        public Transform cameraTransform;
        [Range(0, 360)]
        public float rotationLock;
        public float speed = 5;
        float yRot = 0;
        float xRot = 0;
        #endregion

        #region Unity Methods
        void Start()
        {
            Cursor.lockState = CursorLockMode.Locked;
            Cursor.visible = false;
        }

        void Update()
        {
            //*the look wil not update when a inventory GUI is open
            if (!THInput.isAnotherInventoryOpen)
            {
                Look();
            }
        }
        #endregion

        #region Methods
        void Look()
        {
            //Only X/Y rotation needed as Z rotation would be wierd
            yRot += Input.GetAxis("Mouse X") * speed * Time.timeScale;
            xRot -= Input.GetAxis("Mouse Y") * speed * Time.timeScale;

            //clamps the X rotation so the player camera cannot do flips
            xRot = Mathf.Clamp(xRot, -rotationLock, rotationLock);

            myTransform.rotation = Quaternion.Euler(0, yRot, 0);
            cameraTransform.localRotation = Quaternion.Euler(xRot, 0, 0);
        }
        #endregion
    }
}
```

7.85 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/PlayerMove.cs File Reference

Classes

- class BeeGame.Player.PlayerMove

  Moves the player

Namespaces

- namespace BeeGame.Player

7.86 PlayerMove.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using UnityEngine;
using BeeGame.Core;

namespace BeeGame.Player
{
    [RequireComponent(typeof(Rigidbody))]
    public class PlayerMove : MonoBehaviour
    {
        #region Data
        public float speed = 10f;
        public float gravity = 9.81f;
        public float maxVelocity = 10f;

        private bool canJump = false;
        public float jumpHeight = 2f;

        private Rigidbody myRigidBody;
        #endregion

        #region Unity Methods
        private void Awake()
        {
            myRigidBody = GetComponent<Rigidbody>();

            //i want to use myown gravity and rotation
            myRigidBody.useGravity = false;
            myRigidBody.freezeRotation = true;
        }

        void FixedUpdate()
        {
            //If the player is grounded it can move
            if (canJump)
            {
                MovePlayer();
            }

            //adds the downward force
            myRigidBody.AddForce(new Vector3(0, myRigidBody.mass * -gravity, 0));
        }

        private void OnCollisionStay(Collision collision)
        {
            canJump = true;
        }
        #endregion

        #region Movement Methods
        void MovePlayer()
        {
            //Calculate the speed we want to achive
            Vector3 targetVelocity = new Vector3(THInput.GetAxis("Horizontal"), 0,
THInput.GetAxis("Vertical"));
            targetVelocity = transform.TransformDirection(targetVelocity);
            targetVelocity *= speed;

            //Apply a force to reach the target speed
            Vector3 velocity = myRigidBody.velocity;
            Vector3 velocityChange = (targetVelocity - velocity);

            //Clamping the velocity so that the player does not infinatly accelerate
            velocityChange.x = Mathf.Clamp(velocityChange.x, -maxVelocity, maxVelocity);
            velocityChange.z = Mathf.Clamp(velocityChange.z, -maxVelocity, maxVelocity);
            velocityChange.y = 0;

            //Adds the force to the player so they move in the correct direction
            myRigidBody.AddForce(velocityChange, ForceMode.Impulse);

            //Jumping
            if (canJump && THInput.GetButton("Jump"))
            {
                canJump = false;
                myRigidBody.velocity = new Vector3(velocity.x, VerticalJumpSpeed(), velocity.z);
            }
        }

        float VerticalJumpSpeed()
        {
            //*Gets the correct of fore required for the player to reach the desired apex
            //*Can this be done without Square Root as that take alot of work?
            return Mathf.Sqrt(2 * jumpHeight * gravity);
        }
        #endregion
    }
}
```

7.87 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/SavePlayer↵ Position.cs File Reference

Classes

- class BeeGame.Player.SavePlayerPosition

  Saves the player postion

Namespaces

- namespace BeeGame.Player

7.88 SavePlayerPosition.cs

```
using UnityEngine;
using BeeGame.Serialization;

namespace BeeGame.Player
{
    public class SavePlayerPosition : MonoBehaviour
    {
        int counter = 0;

        void Update()
        {
            if(counter == 0)
            {
                counter = 1000;
                Serialization.Serialization.SavePlayerPosition(transform);
                //print("saved player");
            }

            counter--;
        }
    }
}
```

7.89 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Player/Selector.cs File Reference

Classes

- class BeeGame.Player.Selector

  Moves the Block selector

Namespaces

- namespace BeeGame.Player

7.90 Selector.cs

```
using UnityEngine;
using BeeGame.Blocks;
using BeeGame.Terrain.Chunks;
using BeeGame.Inventory.Player_Inventory;
using BeeGame.Items;
using BeeGame.Core;
using static BeeGame.Terrain.LandGeneration.Terrain;
using static BeeGame.Core.THInput;

namespace BeeGame.Player
{
    public class Selector : MonoBehaviour
    {
        #region Data
        public GameObject selector;

        public PlayerInventory playerInventory;

        public LayerMask layers;
        private RaycastHit hit;

        public int selectedHotbarSlot = 27;
        #endregion

        #region Unity Methods
        void Awake()
        {
            selector = Instantiate(selector);
        }

        void FixedUpdate()
        {
            if(!isAnotherInventoryOpen)
                UpdateSelector();
        }

        void Update()
        {
            if (!isAnotherInventoryOpen)
            {
                if (GetButtonDown("Break Block"))
                    BreakBlock();
                if (GetButtonDown("Place"))
                    PlaceBlock();
            }
        }
        #endregion

        #region Update
        void UpdateSelector()
        {
            if (Physics.Raycast(transform.position, transform.forward, out hit, 15, layers))
            {
                selector.SetActive(true);
                selector.transform.position = GetBlockPos(hit);
                //*selector.SetActive(BlockInPosition(GetBlockPos(hit),
 hit.collider.GetComponent<Chunk>()));
            }
            else
            {
                selector.SetActive(false);
            }
            SelectedSlot();
        }

        void SelectedSlot()
        {
            //* adds 1 to the selected slot and if that is out of range set it to the first hotbar slot
            if(Input.GetAxis("Mouse ScrollWheel") > 0)
            {
                selectedHotbarSlot += 1;
                if (selectedHotbarSlot == 36)
                    selectedHotbarSlot = 27;
            }
            //* removes one from the hotbar selector and if the selector would be inside the inventory set
 it to the last slot in the hotbar
            else if (Input.GetAxis("Mouse ScrollWheel") < 0)
            {
                selectedHotbarSlot -= 1;
                if (selectedHotbarSlot == 26)
                    selectedHotbarSlot = 35;
            }

            transform.parent.GetComponentInChildren<PlayerInventory>().SelectedSlot(
selectedHotbarSlot);
        }
        #endregion

        #region Break/Place
        void BreakBlock()
        {
            Chunk chunk = GetChunk(selector.transform.position);

            Block block = chunk.world.GetBlock((int)selector.transform.position.x, (int)selector.transform.position.y, (int)selector.transform.position.z);

            if (!block.breakable)
                return;

            chunk.world.SetBlock((int)selector.transform.position.x, (int)selector.transform.position.y, (int)selector.transform.position.z, new Air(), true);
            //* set to changed so when block is placed down again it will be saved
            block.changed = true;
            block.BreakBlock(selector.transform.position);
        }

        void PlaceBlock()
        {
            Chunk chunk = GetChunk(selector.transform.position);

            if (chunk == null)
                return;

            if (!chunk.GetBlock((int)selector.transform.position.x - chunk.chunkWorldPos.x, (int)selector.transform.position.y - chunk.chunkWorldPos.y, (int)selector.transform.position.z - chunk.chunkWorldPos.z).InteractWithBlock(playerInventory))
                //* gets the item in the hotbar and if the item is placeable place it
                if (transform.parent.GetComponentInChildren<PlayerInventory>().GetItemFromHotBar(selectedHotbarSlot, out Item blockToPlace))
                    chunk.world.SetBlock((int)(selector.transform.position.x + hit.normal.x), (int)(selector.transform.position.y + hit.normal.y), (int)(selector.transform.position.z + hit.normal.z), (Block)blockToPlace.CloneObject(), true);
        }
        #endregion
    }
}
```

7.91 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Properties/AssemblyInfo.cs File Reference

7.92 AssemblyInfo.cs

```
using System.Resources;
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

//* General Information about an assembly is controlled through the following
//* set of attributes. Change these attribute values to modify the information
//* associated with an assembly.
[assembly: AssemblyTitle("BeeGame")]
[assembly: AssemblyDescription("Game made for Conputer Science Project")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("BeeGame")]
[assembly: AssemblyCopyright("Copyright ©  2017")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

//* Setting ComVisible to false makes the types in this assembly not visible
//* to COM components.  If you need to access a type in this assembly from
//* COM, set the ComVisible attribute to true on that type.
[assembly: ComVisible(false)]

//* The following GUID is for the ID of the typelib if this project is exposed to COM
[assembly: Guid("9b332f5d-31cc-41f5-9517-5ed40d0e4855")]

//* Version information for an assembly consists of the following four values:
//*
//*      Major Version
//*      Minor Version
//*      Build Number
//*      Revision
```

```
//*
//* You can specify all the values or you can default the Build and Revision Numbers
//* by using the '*' as shown below:
//* [assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("0.0.0.1")]
[assembly: NeutralResourcesLanguage("en")]

```

7.93 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Quest/QuestBook.cs File Reference

Classes

- class BeeGame.Quest.QuestBook

Namespaces

- namespace BeeGame.Quest

7.94 QuestBook.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using BeeGame.Core.Dictionarys;
using BeeGame.Items;
using UnityEngine;

namespace BeeGame.Quest
{
    public class QuestBook : Item
    {
        public override int maxStackCount => 1;

        public QuestBook() : base("Quest Book")
        {

        }

        public override bool InteractWithObject()
        {
            return true;
        }

        public override Sprite GetItemSprite()
        {
            return SpriteDictionary.GetSprite("TestSprite");
        }

        public override int GetHashCode()
        {
            return 10;
        }
    }
}
```

7.95 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Resources/Resources.Designer.↩ cs File Reference

Classes

- class BeeGame.Resources.Resources

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

Namespaces

- namespace BeeGame.Resources

7.96 Resources.Designer.cs

```
//*------------------------------------------------------------------------------
//* <auto-generated>
//*     This code was generated by a tool.
//*     Runtime Version:4.0.30319.42000
//*
//*     Changes to this file may cause incorrect behavior and will be lost if
//*     the code is regenerated.
//* </auto-generated>
//*------------------------------------------------------------------------------

namespace BeeGame.Resources {
    using System;
    using System.Collections.Generic;
    using UnityEngine;

    //* This class was auto-generated by the StronglyTypedResourceBuilder
    //* class via a tool like ResGen or Visual Studio.
    //* To add or remove a member, edit your .ResX file then rerun ResGen
    //* with the /str option, or rebuild your VS project.
    [global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyTypedResourceBuilder", "4.0.0.0")]
    [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
    [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
    internal class Resources {

        private static global::System.Resources.ResourceManager resourceMan;

        private static global::System.Globalization.CultureInfo resourceCulture;

        [global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance", "CA1811:AvoidUncalledPrivateCode")]
        internal Resources() {
        }

        [global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
        internal static global::System.Resources.ResourceManager ResourceManager {
            get {
                if (object.ReferenceEquals(resourceMan, null)) {
                    global::System.Resources.ResourceManager temp = new global::System.Resources.ResourceManager("BeeGame.Resources.Resources", typeof(Resources).Assembly);
                    resourceMan = temp;
                }
                return resourceMan;
            }
        }

        [global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
        internal static global::System.Globalization.CultureInfo Culture {
            get {
                return resourceCulture;
            }
            set {
                resourceCulture = value;
            }
        }

        internal static byte[] Prefabs {
            get {
                object obj = ResourceManager.GetObject("Prefabs", resourceCulture);
                return ((byte[])(obj));
            }
        }

        internal static byte[] Sprites {
            get {
                object obj = ResourceManager.GetObject("Sprites", resourceCulture);
                return ((byte[])(obj));
            }
        }

        internal static Dictionary<string, Sprite> GetSprites()
        {
            string[] splitCharacters = new string[] { "," };
            object obj = ResourceManager.GetObject("Sprites", resourceCulture);

            string text = System.Text.Encoding.Default.GetString((byte[])obj);
            string lineText = "";
            string[] splitText;
            Texture2D tex;
            Dictionary<string, Sprite> sprites = new Dictionary<string, Sprite>();

            for (int i = 0; i < text.Length; i++)
            {
                if (text[i] != '\n')
                {
                    lineText += text[i];
                }
                else
                {
                    splitText = lineText.Split(splitCharacters, StringSplitOptions.RemoveEmptyEntries);
                    lineText = "";
                    tex = UnityEngine.Resources.Load("Sprites/" + splitText[1].Remove(splitText[
      1].Length - 1, 1)) as Texture2D;
                    sprites.Add(splitText[0], Sprite.Create(tex, new UnityEngine.Rect(0, 0, tex.
      width, tex.height), Vector2.zero));
                }
            }

            splitText = lineText.Split(splitCharacters, StringSplitOptions.RemoveEmptyEntries);
            lineText = "";
            tex = UnityEngine.Resources.Load("Sprites/" + splitText[1]) as Texture2D;
            sprites.Add(splitText[0], Sprite.Create(tex, new UnityEngine.Rect(0, 0, tex.width,
      tex.height), Vector2.zero));

            return sprites;
        }

        internal static Dictionary<string, GameObject> GetPrefabs()
        {
            string[] splitCharacters = new string[] { "," };
            object obj = ResourceManager.GetObject("Prefabs", resourceCulture);

            string text = System.Text.Encoding.Default.GetString((byte[])obj);
            text = text.Remove(0, 3);
            string lineText = "";
            string[] splitText;
            Dictionary<string, GameObject> objects = new Dictionary<string, GameObject>();

            for (int i = 0; i < text.Length; i++)
            {
                if(text[i] != '\n')
                {
                    lineText += text[i];
                }
                else
                {
                    splitText = lineText.Split(splitCharacters, StringSplitOptions.RemoveEmptyEntries);
                    lineText = "";
                    objects.Add(splitText[0], UnityEngine.Resources.Load("Prefabs/" + splitText[
      1].Remove(splitText[1].Length - 1, 1)) as GameObject);
                }
            }

            splitText = lineText.Split(splitCharacters, StringSplitOptions.RemoveEmptyEntries);
            lineText = "";
            objects.Add(splitText[0], UnityEngine.Resources.Load("Prefabs/" + splitText[1]) as
      GameObject);

            return objects;
        }
    }
}
```

7.97 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Serialization/Serialization.cs File Reference

Classes

- class BeeGame.Serialization.Serialization

  Serializes and Deserialises things

Namespaces

- namespace BeeGame.Serialization

7.98 Serialization.cs

```
using System.IO;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Formatters.Binary;
using UnityEngine;
using BeeGame.Core;
using BeeGame.Terrain;
using BeeGame.Terrain.Chunks;
using BeeGame.Inventory;
using BeeGame.Blocks;

namespace BeeGame.Serialization
{
    public static class Serialization
    {
        #region Data
        public static string worldName = "World";
        public static string saveFolderName = "Saves";
        private static string savePath;
        #endregion

        public static void MakeDirectorys()
        {
            savePath = $"{Application.dataPath}/{saveFolderName}/{worldName}";

            if (!(Directory.Exists(savePath)))
                Directory.CreateDirectory(savePath);
        }

        public static void DeleteFile(string fileName)
        {
            string[] file = Directory.GetFiles(Application.dataPath + "/Saves", "*.dat", SearchOption.
AllDirectories);

            string[] splitCharacters = { "/", "\\", ".dat" };

            for (int i = 0; i < file.Length; i++)
            {
                string[] temp = file[i].Split(splitCharacters, System.StringSplitOptions.
RemoveEmptyEntries);

                if(temp[temp.Length - 1] == fileName)
                {
                    File.Delete(file[i]);

                    return;
                }
            }
        }

        #region Player
        public static void SavePlayerPosition(Transform positon)
        {
            THVector3[] playerTransform = new THVector3[3];

            playerTransform[0] = positon.position;
            playerTransform[1] = positon.rotation.eulerAngles;
            playerTransform[2] = positon.localScale;

            string playerPosSavePath = $"{savePath}/player.dat";

            SaveFile(playerTransform, playerPosSavePath);
        }

        public static void LoadPlayerPosition(Transform playerTransfom)
        {
            string playerPosSavePath = $"{savePath}/player.dat";

            if (!File.Exists(playerPosSavePath))
                return;

            THVector3[] pos = (THVector3[])LoadFile(playerPosSavePath);

            playerTransfom.position = pos[0];
            playerTransfom.rotation = (Quaternion)pos[1];
            playerTransfom.localScale = pos[2];
        }
        #endregion

        #region Inventorys
        public static void SerializeInventory(Inventory.Inventory inventory, string inventoryName)
        {
            string inventorySavePath = $"{savePath}/Inventorys";

            if (!Directory.Exists(inventorySavePath))
                Directory.CreateDirectory(inventorySavePath);

            SaveFile(inventory.GetAllItems(), $"{inventorySavePath}/{inventoryName}.dat");
        }

        public static void DeSerializeInventory(Inventory.Inventory inventory,
    string inventoryName)
        {
            //* make the path
            string inventorySavePath = $"{savePath}/Inventorys/{inventoryName}.dat";

            //* checks that the file exists
            if (!File.Exists(inventorySavePath))
            {
                for (int i = 0; i < inventory.items.itemsInInventory.Length; i++)
                {
                    inventory.items.itemsInInventory[i] = null;
                }

                SerializeInventory(inventory, inventoryName);

                return;
            }

            inventory.SetAllItems((ItemsInInventory)LoadFile($"{inventorySavePath}"));
        }
        #endregion

        #region Chunk
        public static void SaveChunk(Chunk chunk)
        {
            //* saves the blocks
            SaveChunk save = new SaveChunk(chunk.blocks);

            //* if no block was changed return early
            if (save.blocks.Count == 0)
                return;

            //* otherwise save the file
            string saveFile = $"{savePath}/{FileName(chunk.chunkWorldPos)}.dat";

            SaveFile(save, saveFile);
        }

        public static bool LoadChunk(Chunk chunk)
        {
            //* gets the save file
            string saveFile = $"{savePath}/{FileName(chunk.chunkWorldPos)}.dat";

            //* if the file does not exist return false
            if (!File.Exists(saveFile))
                return false;

            //* set all of the changed blocks in the chunk
            SaveChunk save = (SaveChunk)LoadFile(saveFile);

            foreach (var block in save.blocks)
            {
                chunk.blocks[block.Key.x, block.Key.y, block.Key.z] = block.Value;
            }

            return true;
        }

        public static string FileName(ChunkWorldPos pos)
        {
            return $"{pos.x}, {pos.y}, {pos.z}";
        }
        #endregion

        #region Save/Load Files
        private static void SaveFile(object obj, string file)
        {
            BinaryFormatter bf = new BinaryFormatter();
            FileStream fs = new FileStream(file, FileMode.OpenOrCreate);

            try
            {
                bf.Serialize(fs, obj);
            }
            catch(SerializationException e)
            {
                Debug.Log($"Serialization Exception: {e}");
                throw new SerializationException();
            }
            finally
            {
                fs.Close();
            }
        }

        private static object LoadFile(string file)
        {
            BinaryFormatter bf = new BinaryFormatter();
            FileStream fs = new FileStream(file, FileMode.Open);

            try
            {
                return bf.Deserialize(fs);
            }
            catch(SerializationException e)
            {
                Debug.Log($"Deserialization Exception {e}");
                throw new SerializationException();
            }
            finally
            {
                fs.Close();
            }
        }
        #endregion
    }
}
```

7.99 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/SpawnITem.cs File Reference

Classes

- class BeeGame.SpawnItem

Namespaces

- namespace BeeGame

7.100 SpawnITem.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using UnityEngine;
using BeeGame.Items;
using BeeGame.Blocks;
using BeeGame.Core.Enums;

namespace BeeGame
{
    class SpawnItem : MonoBehaviour
    {
        void Start()
        {
            GameObject go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as
GameObject, transform.position, Quaternion.identity) as GameObject;
            go.GetComponent<ItemGameObject>().item = new Bee(
BeeType.DRONE, new NormalBee() { pSpecies = BeeSpecies.FOREST });

            go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject,
 transform.position, Quaternion.identity) as GameObject;
            go.GetComponent<ItemGameObject>().item = new Bee(
BeeType.PRINCESS, new NormalBee() { pSpecies = BeeSpecies.FOREST });
```

```

            go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject, transform.position, Quaternion.identity) as GameObject;
            go.GetComponent<ItemGameObject>().item = new Bee(BeeType.DRONE, new NormalBee() { pSpecies = BeeSpecies.COMMON, sSpecies = BeeSpecies.COMMON });

            go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject, transform.position, Quaternion.identity) as GameObject;
            go.GetComponent<ItemGameObject>().item = new Bee(BeeType.PRINCESS, new NormalBee() { pSpecies = BeeSpecies.COMMON, sSpecies = BeeSpecies.COMMON });

            //go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject, transform.position, Quaternion.identity) as GameObject;
            //go.GetComponent<ItemGameObject>().item = new Bee(BeeType.QUEEN, new QueenBee());

            go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject, transform.position, Quaternion.identity) as GameObject;
            go.GetComponent<ItemGameObject>().item = new HoneyComb(HoneyCombType.ICEY);

            go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject, transform.position, Quaternion.identity) as GameObject;
            go.GetComponent<ItemGameObject>().item = new HoneyComb(HoneyCombType.HONEY);

            go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject, transform.position, Quaternion.identity) as GameObject;
            go.GetComponent<ItemGameObject>().item = new Chest();
            go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject, transform.position, Quaternion.identity) as GameObject;
            go.GetComponent<ItemGameObject>().item = new Chest();

            go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject, transform.position, Quaternion.identity) as GameObject;
            go.GetComponent<ItemGameObject>().item = new Apiary();

            go = Instantiate(UnityEngine.Resources.Load("Prefabs/ItemGameObject") as GameObject, transform.position, Quaternion.identity) as GameObject;
            go.GetComponent<ItemGameObject>().item = new CraftingTable();
        }

        private void OnDrawGizmos()
        {
            //Gizmos.color = Color.green;
            //Gizmos.DrawSphere(transform.position, 0.5f);
        }
    }
}
```

7.101 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/↩ Chunk.cs File Reference

Classes

- class BeeGame.Terrain.Chunks.Chunk

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

Namespaces

- namespace BeeGame.Terrain.Chunks

7.102 Chunk.cs

```
using UnityEngine;
using BeeGame.Blocks;
using BeeGame.Terrain.LandGeneration;
using System.Threading;

namespace BeeGame.Terrain.Chunks
```

```
{
    [RequireComponent(typeof(MeshFilter))]
    [RequireComponent(typeof(MeshRenderer))]
    [RequireComponent(typeof(MeshCollider))]
    public class Chunk : MonoBehaviour
    {
        #region Data
        public static int chunkSize = 16;

        public Block[,,] blocks = new Block[chunkSize, chunkSize, chunkSize];

        public bool update = true;
        public bool rendered;

        public bool updateCollsionMesh = false;
        public bool applyCollisionMesh = false;

        public World world;
        public ChunkWorldPos chunkWorldPos;

        private MeshData mesh = new MeshData();

        private MeshFilter filter;
        private MeshCollider meshCollider;
        #endregion

        #region Unity Methods
        void Start()
        {
            filter = GetComponent<MeshFilter>();
            meshCollider = GetComponent<MeshCollider>();
        }

        void Update()
        {
            lock(mesh)
            {
                if (update)
                {
                    update = false;
                    updateCollsionMesh = true;
                    mesh = new MeshData();
                    //* Enabling threading here works in editor but not in build?
                    //* ok whatever...
                    //* Thread thread = new Thread(UpdateChunk);

                    //* thread.Start();
                    UpdateChunk();
                }

                if (mesh.done && mesh != new MeshData())
                {
                    RenderMesh(mesh);
                }

                if (applyCollisionMesh)
                    ColliderMesh();
            }
        }
        #endregion

        #region Get/Set Blocks
        public Block GetBlock(int x, int y, int z, bool checkNebouringChunks = true)
        {
            //* checks that block is in the chunk
            if (InRange(x) && InRange(y) && InRange(z))
                return blocks[x, y, z];

            //* if the block is not in the chunk and we should check other chunks do that, otherwise return
 an air block (empty block)
            //if(checkNebouringChunks)
                return world.GetBlock(chunkWorldPos.x + x, chunkWorldPos.
y + y, chunkWorldPos.z + z);

            //return new Air();
        }

        public void SetBlock(int x, int y, int z, Block block, bool checkNebouringChunks =
true)
        {
            //* sets the block in the position if it is in the chunk, then return early
            if (InRange(x) && InRange(y) && InRange(z))
            {
                blocks[x, y, z] = block;
                return;
            }

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

        public static bool InRange(int i)
        {
            //* if the value is less then 0 or greater than 16 the value is outside the chunk
            if (i < 0 || i >= chunkSize)
                return false;
            return true;
        }
        #endregion

        #region Mesh
        public void SetBlocksUnmodified()
        {
            foreach (var block in blocks)
            {
                block.changed = false;
            }
        }

        void UpdateChunk()
        {
            //* says that this chunk is rendered and initialtes the mesh
            rendered = true;

            //* goes through every block in the blocks array getting their mesh data
            for (int x = 0; x < chunkSize; x ++)
            {
                for (int z = 0; z < chunkSize; z ++)
                {
                    for (int y = 0; y < chunkSize; y ++)
                    {
                        blocks[x, y, z]?.UpdateBlock(x, y, z, this);
                        mesh = blocks[x, y, z]?.BlockData(this, x, y, z, mesh) ?? mesh;
                    }
                }
            }
            mesh.done = true;
        }

        void RenderMesh(MeshData meshData)
        {
            //* Applying the mesh takes the longest but nothing can be dont with the mesh class in a secondary thread...thanks unity

            mesh.done = false;
            //* clears the current chunk mesh
            filter.mesh.Clear();
            //* name for convenience
            filter.mesh.name = "Render Mesh";
            //* puts the tris and verts from the meshdata into the chunk mesh
            filter.mesh.vertices = meshData.verts.ToArray();
            filter.mesh.triangles = meshData.tris.ToArray();

            //* sets the uvs
            filter.mesh.uv = meshData.uv.ToArray();

            //* redoes the normals incase they got messed up
            filter.mesh.RecalculateNormals();
            //* is this necissary as it causes alsot of lag?
        }

        void ColliderMesh()
        {
            //* if the chunk has been told to update the collsions but the chunk has ne verts dont do it as their is no point
            if (this.mesh.verts.Count == 0)
                return;

            //* if the render and collision meshes should be shared set the render mesh to the collision mesh otherwise make a collision mesh
            if (this.mesh.shareMeshes)
            {
                world.chunkHasMadeCollisionMesh = true;
                applyCollisionMesh = false;
                meshCollider.sharedMesh = filter.mesh;
                return;
            }

            world.chunkHasMadeCollisionMesh = true;
            //* Applying the mesh takes the longest but nothing can be done with the mesh class in a secondary thread...thanks Unity

            //* makes a new mesh setting the name for convenience
            Mesh mesh = new Mesh()
            {
                name = "Collider Mesh",
                vertices = this.mesh.colVerts.ToArray(),
                triangles = this.mesh.colTris.ToArray()
            };

            //* recalcs the normals and applies the mesh
            mesh.RecalculateNormals();

            meshCollider.sharedMesh = mesh;

            applyCollisionMesh = false;
        }
        #endregion
    }
}
```

7.103 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/Load↩ Chunks.cs File Reference

Classes

- class BeeGame.Terrain.Chunks.LoadChunks

  Loads the Chunks around the player

Namespaces

- namespace BeeGame.Terrain.Chunks

7.104 LoadChunks.cs

```
using System;
using System.Collections.Generic;
using UnityEngine;
using BeeGame.Terrain.LandGeneration;

namespace BeeGame.Terrain.Chunks
{
    public class LoadChunks : MonoBehaviour
    {
        #region Data
        public World world;

        private List<ChunkWorldPos> buildList = new List<ChunkWorldPos>();

        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),
                             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),
                             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),
                             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),
                             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),
                             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),
                             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),
                             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),
                              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),
                              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),
                              new ChunkWorldPos(-4, 0,  1), new ChunkWorldPos(-1, 0, -4), new ChunkWorldPos(-1, 0,  4), new ChunkWorldPos( 1, 0, -4), new ChunkWorldPos( 1, 0,  4),
                              new ChunkWorldPos( 4, 0, -1), new ChunkWorldPos( 4, 0,  1), new ChunkWorldPos(-3, 0, -3), new ChunkWorldPos(-3, 0,  3), new ChunkWorldPos( 3, 0, -3),
                              new ChunkWorldPos( 3, 0,  3), new ChunkWorldPos(-4, 0, -2), new ChunkWorldPos(-4, 0,  2), new ChunkWorldPos(-2, 0, -4), new ChunkWorldPos(-2, 0,  4),
                              new ChunkWorldPos( 2, 0, -4), new ChunkWorldPos( 2, 0,  4), new ChunkWorldPos( 4, 0, -2), new ChunkWorldPos( 4, 0,  2), new ChunkWorldPos(-5, 0,  0),
                              new ChunkWorldPos(-4, 0, -3), new ChunkWorldPos(-4, 0,  3), new ChunkWorldPos(-3, 0, -4), new ChunkWorldPos(-3, 0,  4), new ChunkWorldPos( 0, 0, -5),
                              new ChunkWorldPos( 0, 0,  5), new ChunkWorldPos( 3, 0, -4), new ChunkWorldPos( 3, 0,  4), new ChunkWorldPos( 4, 0, -3), new ChunkWorldPos( 4, 0,  3),
                              new ChunkWorldPos( 5, 0,  0), new ChunkWorldPos(-5, 0, -1), new ChunkWorldPos(-5, 0,  1), new ChunkWorldPos(-1, 0, -5), new ChunkWorldPos(-1, 0,  5),
                              new ChunkWorldPos( 1, 0, -5), new ChunkWorldPos( 1, 0,  5), new ChunkWorldPos( 5, 0, -1), new ChunkWorldPos( 5, 0,  1), new ChunkWorldPos(-5, 0, -2),
                              new ChunkWorldPos(-5, 0,  2), new ChunkWorldPos(-2, 0, -5), new ChunkWorldPos(-2, 0,  5), new ChunkWorldPos( 2, 0, -5), new ChunkWorldPos( 2, 0,  5),
                              new ChunkWorldPos( 5, 0, -2), new ChunkWorldPos( 5, 0,  2), new ChunkWorldPos(-4, 0, -4), new ChunkWorldPos(-4, 0,  4), new ChunkWorldPos( 4, 0, -4),
                              new ChunkWorldPos( 4, 0,  4), new ChunkWorldPos(-5, 0, -3), new ChunkWorldPos(-5, 0,  3), new ChunkWorldPos(-3, 0, -5), new ChunkWorldPos(-3, 0,  5),
                              new ChunkWorldPos( 3, 0, -5), new ChunkWorldPos( 3, 0,  5), new ChunkWorldPos( 5, 0, -3), new ChunkWorldPos( 5, 0,  3), new ChunkWorldPos(-6, 0,  0),
                              new ChunkWorldPos( 0, 0, -6), new ChunkWorldPos( 0, 0,  6), new ChunkWorldPos( 6, 0,  0), new ChunkWorldPos(-6, 0, -1), new ChunkWorldPos(-6, 0,  1),
                              new ChunkWorldPos(-1, 0, -6), new ChunkWorldPos(-1, 0,  6), new ChunkWorldPos( 1, 0, -6), new ChunkWorldPos( 1, 0,  6), new ChunkWorldPos( 6, 0, -1),
                              new ChunkWorldPos( 6, 0,  1), new ChunkWorldPos(-6, 0, -2), new ChunkWorldPos(-6, 0,  2), new ChunkWorldPos(-2, 0, -6), new ChunkWorldPos(-2, 0,  6),
                              new ChunkWorldPos( 2, 0, -6), new ChunkWorldPos( 2, 0,  6), new ChunkWorldPos( 6, 0, -2), new ChunkWorldPos( 6, 0,  2), new ChunkWorldPos(-5, 0, -4),
                              new ChunkWorldPos(-5, 0,  4), new ChunkWorldPos(-4, 0, -5), new ChunkWorldPos(-4, 0,  5), new ChunkWorldPos( 4, 0, -5), new ChunkWorldPos( 4, 0,  5),
                              new ChunkWorldPos( 5, 0, -4), new ChunkWorldPos( 5, 0,  4), new ChunkWorldPos(-6, 0, -3), new ChunkWorldPos(-6, 0,  3), new ChunkWorldPos(-3, 0, -6),
                              new ChunkWorldPos(-3, 0,  6), new ChunkWorldPos( 3, 0, -6), new ChunkWorldPos( 3, 0,  6), new ChunkWorldPos( 6, 0, -3), new ChunkWorldPos( 6, 0,  3),
                              new ChunkWorldPos(-7, 0,  0), new ChunkWorldPos( 0, 0, -7), new ChunkWorldPos( 0, 0,  7), new ChunkWorldPos( 7, 0,  0), new ChunkWorldPos(-7, 0, -1),
                              new ChunkWorldPos(-7, 0,  1), new ChunkWorldPos(-5, 0, -5), new ChunkWorldPos(-5, 0,  5), new ChunkWorldPos(-1, 0, -7), new ChunkWorldPos(-1, 0,  7),
                              new ChunkWorldPos( 1, 0, -7), new ChunkWorldPos( 1, 0,  7), new ChunkWorldPos( 5, 0, -5), new ChunkWorldPos( 5, 0,  5), new ChunkWorldPos( 7, 0, -1),
                              new ChunkWorldPos( 7, 0,  1), new ChunkWorldPos(-6, 0, -4), new ChunkWorldPos(-6, 0,  4), new ChunkWorldPos(-4, 0, -6), new ChunkWorldPos(-4, 0,  6),
                              new ChunkWorldPos( 4, 0, -6), new ChunkWorldPos( 4, 0,  6), new ChunkWorldPos( 6, 0, -4), new ChunkWorldPos( 6, 0,  4), new ChunkWorldPos(-7, 0, -2),
                              new ChunkWorldPos(-7, 0,  2), new ChunkWorldPos(-2, 0, -7), new ChunkWorldPos(-2, 0,  7), new ChunkWorldPos( 2, 0, -7), new ChunkWorldPos( 2, 0,  7),
                              new ChunkWorldPos( 7, 0, -2), new ChunkWorldPos( 7, 0,  2), new ChunkWorldPos(-7, 0, -3), new ChunkWorldPos(-7, 0,  3), new ChunkWorldPos(-3, 0, -7),
                              new ChunkWorldPos(-3, 0,  7), new ChunkWorldPos( 3, 0, -7), new ChunkWorldPos( 3, 0,  7), new ChunkWorldPos( 7, 0, -3), new ChunkWorldPos( 7, 0,  3),
                             new ChunkWorldPos(-6, 0, -5), new ChunkWorldPos(-6, 0,  5), new ChunkWorldPos(-5, 0, -6), new ChunkWorldPos(-5, 0,  6), new ChunkWorldPos( 5, 0, -6),
                             new ChunkWorldPos( 5, 0,  6), new ChunkWorldPos( 6, 0, -5), new ChunkWorldPos( 6, 0,  5) };

        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),
                                                                            new ChunkWorldPos(1, 0, 1), new ChunkWorldPos(1, 0, -1), new ChunkWorldPos(-1, 0, 1), new ChunkWorldPos(-1, 0, -1)};

        private static int timer = 0;
        #endregion

        private void Start()
        {
            LandGeneration.Terrain.world = world;
        }

        void Update()
        {
            if (DeleteChunks())
                return;
            if (!world.chunkHasMadeCollisionMesh)
            {
                FindChunksToLoad();
                LoadAndRenderChunks();
                ApplyCollsionMeshToNearbyChunks();
            }
            //* stops chunks being made and collision meshes being made at the same time
            world.chunkHasMadeCollisionMesh = false;
        }

        void ApplyCollsionMeshToNearbyChunks()
        {
            //* gets the player position in chunk coordinates
            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);

            for (int i = 0; i < nearbyChunks.Length; i++)
            {
                ChunkWorldPos chunkPos = new ChunkWorldPos(nearbyChunks[i].x * Chunk.chunkSize + playerPos.x, 0, nearbyChunks[i].z * Chunk.chunkSize + playerPos.z);

                for (int j = -1; j < 2; j++)
                {
                    Chunk nearbyChunk = world.GetChunk(chunkPos.x, j * Chunk.chunkSize, chunkPos.z);

                    if (nearbyChunk != null)
                        nearbyChunk.applyCollisionMesh = true;
                }
            }
        }

        void LoadAndRenderChunks()
        {
            //* if their is somethign in the build list new chunks can be made
            if (buildList.Count != 0)
            {
                //* 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
                for (int i = buildList.Count - 1, j = 0; i >= 0 && j < 8; i--, j++)
                {
                    BuildChunk(buildList[0]);
                    buildList.RemoveAt(0);
                }
            }
        }

        void FindChunksToLoad()
        {
            if (buildList.Count == 0)
            {
                //* gets the player position in chunk coordinates
                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);

                //* check all of the chunk positions and if that position does not have a chunk in it make
```

```
it
                for (int i = 0; i < chunkPositions.Length; i++)
                {
                    ChunkWorldPos newChunkPos = new ChunkWorldPos(chunkPositions[i].x * Chunk.chunkSize + playerPos.x, 0, chunkPositions[i].z * Chunk.chunkSize + playerPos.z);

                    Chunk newChunk = world.GetChunk(newChunkPos.x, newChunkPos.y, newChunkPos.z);

                    if (newChunk != null && (newChunk.rendered || buildList.Contains(newChunkPos)))
                        continue;

                    for (int y = -1; y < 2; y++)
                    {
                        for (int x = newChunkPos.x - Chunk.chunkSize; x < newChunkPos.x + Chunk.chunkSize; x += Chunk.chunkSize)
                        {
                            for (int z = newChunkPos.z - Chunk.chunkSize; z < newChunkPos.z + Chunk.chunkSize; z += Chunk.chunkSize)
                            {
                                buildList.Add(new ChunkWorldPos(x, y * Chunk.chunkSize, z));
                            }
                        }
                    }
                    return;
                }
            }
        }

        void BuildChunk(ChunkWorldPos pos)
        {
            if (world.GetChunk(pos.x, pos.y, pos.z) == null)
                world.CreateChunk(pos.x, pos.y, pos.z);
        }

        bool DeleteChunks()
        {
            //* destroys every 10 call to reduce load on CPU so that chunks are not destroyed and created at the same time
            if(timer == 10)
            {
                timer = 0;
                var chunksToDelete = new List<ChunkWorldPos>();

                // *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
                foreach (var chunk in world.chunks)
                {
                    float distance = Vector3.Distance(chunk.Value.transform.position, transform.position);

                    if (distance > 256)
                        chunksToDelete.Add(chunk.Key);
                }

                foreach (var chunk in chunksToDelete)
                {
                    world.DestroyChunk(chunk.x, chunk.y, chunk.z);
                }

                return true;
            }

            timer++;

            return false;
        }
    }
}
```

7.105 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/Mesh↩ Data.cs File Reference

Classes

- class BeeGame.Terrain.Chunks.MeshData

  The data for a Chunks's Mesh

Namespaces

- namespace BeeGame.Terrain.Chunks

7.106 MeshData.cs

```
using System.Collections.Generic;
using UnityEngine;
using BeeGame.Core.Enums;
using BeeGame.Core;

namespace BeeGame.Terrain.Chunks
{
    public class MeshData
    {
        public List<Vector3> verts = new List<Vector3>();
        public List<int> tris = new List<int>();
        public List<Vector2> uv = new List<Vector2>();

        public List<Vector3> colVerts = new List<Vector3>();
        public List<int> colTris = new List<int>();

        public bool shareMeshes = true;

        public bool done = false;

        public void AddQuadTriangles(bool addToRenderMesh = true)
        {
            //*adds the triangles in an anticlockwise order

            if (addToRenderMesh)
            {
                tris.Add(verts.Count - 4);
                tris.Add(verts.Count - 3);
                tris.Add(verts.Count - 2);
                tris.Add(verts.Count - 4);
                tris.Add(verts.Count - 2);
                tris.Add(verts.Count - 1);
            }

            colTris.Add(colVerts.Count - 4);
            colTris.Add(colVerts.Count - 3);
            colTris.Add(colVerts.Count - 2);
            colTris.Add(colVerts.Count - 4);
            colTris.Add(colVerts.Count - 2);
            colTris.Add(colVerts.Count - 1);
        }

        public void AddVertices(THVector3 pos, bool addToRenderMesh = true,
Direction direction = Direction.DOWN)
        {
            if (addToRenderMesh)
                verts.Add(pos);

            //* if the vertice is on the top face make its positon slightly smaller
            if(direction == Direction.UP)
                colVerts.Add(pos - new THVector3(0.01f, 0, 0.01f));
        }

        public void AddTriangle(int tri)
        {
            tris.Add(tri);

            colTris.Add(tri - (verts.Count - colVerts.Count));
        }
    }
}
```

7.107 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/Chunks/Save↩ Chunk.cs File Reference

Classes

- class BeeGame.Terrain.Chunks.SaveChunk

    Saves a Chunks modified Blocks for save optimisation

Namespaces

- namespace BeeGame.Terrain.Chunks

7.108 SaveChunk.cs

```
using System;
using System.Collections.Generic;
using BeeGame.Blocks;


namespace BeeGame.Terrain.Chunks
{
    [Serializable]
    public class SaveChunk
    {
        public Dictionary<ChunkWorldPos, Block> blocks = new Dictionary<ChunkWorldPos, Block>();

        public SaveChunk(Block[,,] blockArray)
        {
            for (int x = 0; x < Chunk.chunkSize; x++)
            {
                for (int y = 0; y < Chunk.chunkSize; y++)
                {
                    for (int z = 0; z < Chunk.chunkSize; z++)
                    {
                        //* if the block has changed save it
                        if (blockArray[x, y, z].changed)
                            blocks.Add(new ChunkWorldPos(x, y, z), blockArray[x, y, z]);
                    }
                }
            }
        }
    }
}
```

7.109 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/ChunkWorld↩Pos.cs File Reference

Classes

- struct BeeGame.Terrain.ChunkWorldPos

  Serializable int version of THVector3

Namespaces

- namespace BeeGame.Terrain

7.110 ChunkWorldPos.cs

```
using System;
using BeeGame.Core;

namespace BeeGame.Terrain
{
    [Serializable]
    public struct ChunkWorldPos
    {
        public int x, y, z;

        public ChunkWorldPos(int x, int y, int z)
        {
            this.x = x;
            this.y = y;
            this.z = z;
        }

        public override string ToString()
        {
            return $"({x}, {y}, {z})";
        }

        //* TODO probly add the == and != but for now this is fine
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "
CA2231:OverloadOperatorEqualsOnOverridingValueTypeEquals")]
        public override bool Equals(object obj)
        {
            //* possibly remove and just check if obj is null
            if (!(obj is ChunkWorldPos))
                return false;

            ChunkWorldPos temp = (ChunkWorldPos)obj;

            //* possibly change to hashcode checking
            if (temp.x == x && temp.y == y && temp.z == z)
                return true;

            return false;
        }

        public override int GetHashCode()
        {
            unchecked
            {
                int hashcode = 47;

                hashcode *= 227 + x.GetHashCode();
                hashcode *= 227 + y.GetHashCode();
                hashcode *= 227 + z.GetHashCode();

                return hashcode;
            }
        }

        public static implicit operator THVector3(ChunkWorldPos pos)
        {
            return new THVector3(pos.x, pos.y, pos.z);
        }

        public static explicit operator ChunkWorldPos(THVector3 pos)
        {
            return new ChunkWorldPos((int)pos.x, (int)pos.y, (int)pos.
z);
        }
    }
}
```

7.111 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/LandGeneration/↩ Noise/SimplexNoise.cs File Reference

Classes

- class BeeGame.Terrain.LandGeneration.Noise.SimplexNoise

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

Namespaces

- namespace BeeGame.Terrain.LandGeneration.Noise

7.112 SimplexNoise.cs

```
//* SimplexNoise for C#
//* Author: Heikki Törmälä

//*This is free and unencumbered software released into the public domain.

```

```
//*Anyone is free to copy, modify, publish, use, compile, sell, or
//*distribute this software, either in source code form or as a compiled
//*binary, for any purpose, commercial or non-commercial, and by any
//*means.

//*In jurisdictions that recognize copyright laws, the author or authors
//*of this software dedicate any and all copyright interest in the
//*software to the public domain. We make this dedication for the benefit
//*of the public at large and to the detriment of our heirs and
//*successors. We intend this dedication to be an overt act of
//*relinquishment in perpetuity of all present and future rights to this
//*software under copyright law.

//*THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
//*EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
//*MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
//*IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY CLAIM, DAMAGES OR
//*OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
//*ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
//*OTHER DEALINGS IN THE SOFTWARE.

//*For more information, please refer to <http://unlicense.org/>


namespace BeeGame.Terrain.LandGeneration.Noise
{
    public class SimplexNoise
    {
        public static float Generate(float x)
        {
            int i0 = FastFloor(x);
            int i1 = i0 + 1;
            float x0 = x - i0;
            float x1 = x0 - 1.0f;

            float n0, n1;

            float t0 = 1.0f - x0 * x0;
            t0 *= t0;
            n0 = t0 * t0 * grad(perm[i0 & 0xff], x0);

            float t1 = 1.0f - x1 * x1;
            t1 *= t1;
            n1 = t1 * t1 * grad(perm[i1 & 0xff], x1);
            //* The maximum value of this noise is 8*(3/4)^4 = 2.53125
            //* A factor of 0.395 scales to fit exactly within [-1,1]
            return 0.395f * (n0 + n1);
        }

        public static float Generate(float x, float y)
        {
            const float F2 = 0.366025403f; //* F2 = 0.5*(sqrt(3.0)-1.0)
            const float G2 = 0.211324865f; //* G2 = (3.0-Math.sqrt(3.0))/6.0

            float n0, n1, n2; //* Noise contributions from the three corners

            //* Skew the input space to determine which simplex cell we're in
            float s = (x + y) * F2; //* Hairy factor for 2D
            float xs = x + s;
            float ys = y + s;
            int i = FastFloor(xs);
            int j = FastFloor(ys);

            float t = (float)(i + j) * G2;
            float X0 = i - t; //* Unskew the cell origin back to (x,y) space
            float Y0 = j - t;
            float x0 = x - X0; //* The x,y distances from the cell origin
            float y0 = y - Y0;

            //* For the 2D case, the simplex shape is an equilateral triangle.
            //* Determine which simplex we are in.
            int i1, j1; //* Offsets for second (middle) corner of simplex in (i,j) coords
            if (x0 > y0) { i1 = 1; j1 = 0; } //* lower triangle, XY order: (0,0)->(1,0)->(1,1)
            else { i1 = 0; j1 = 1; }      //* upper triangle, YX order: (0,0)->(0,1)->(1,1)

            //* A step of (1,0) in (i,j) means a step of (1-c,-c) in (x,y), and
            //* a step of (0,1) in (i,j) means a step of (-c,1-c) in (x,y), where
            //* c = (3-sqrt(3))/6

            float x1 = x0 - i1 + G2; //* Offsets for middle corner in (x,y) unskewed coords
            float y1 = y0 - j1 + G2;
            float x2 = x0 - 1.0f + 2.0f * G2; //* Offsets for last corner in (x,y) unskewed coords
            float y2 = y0 - 1.0f + 2.0f * G2;

            //* Wrap the integer indices at 256, to avoid indexing perm[] out of bounds
            int ii = i % 256;
            int jj = j % 256;

            //* Calculate the contribution from the three corners
            float t0 = 0.5f - x0 * x0 - y0 * y0;
            if (t0 < 0.0f) n0 = 0.0f;
            else
            {
                t0 *= t0;
                n0 = t0 * t0 * grad(perm[ii + perm[jj]], x0, y0);
            }

            float t1 = 0.5f - x1 * x1 - y1 * y1;
            if (t1 < 0.0f) n1 = 0.0f;
            else
            {
                t1 *= t1;
                n1 = t1 * t1 * grad(perm[ii + i1 + perm[jj + j1]], x1, y1);
            }

            float t2 = 0.5f - x2 * x2 - y2 * y2;
            if (t2 < 0.0f) n2 = 0.0f;
            else
            {
                t2 *= t2;
                n2 = t2 * t2 * grad(perm[ii + 1 + perm[jj + 1]], x2, y2);
            }

            //* Add contributions from each corner to get the final noise value.
            //* The result is scaled to return values in the interval [-1,1].
            return 40.0f * (n0 + n1 + n2); //* TODO: The scale factor is preliminary!
        }


        public static float Generate(float x, float y, float z)
        {
            //* Simple skewing factors for the 3D case
            const float F3 = 0.333333333f;
            const float G3 = 0.166666667f;

            float n0, n1, n2, n3; //* Noise contributions from the four corners

            //* Skew the input space to determine which simplex cell we're in
            float s = (x + y + z) * F3; //* Very nice and simple skew factor for 3D
            float xs = x + s;
            float ys = y + s;
            float zs = z + s;
            int i = FastFloor(xs);
            int j = FastFloor(ys);
            int k = FastFloor(zs);

            float t = (float)(i + j + k) * G3;
            float X0 = i - t; //* Unskew the cell origin back to (x,y,z) space
            float Y0 = j - t;
            float Z0 = k - t;
            float x0 = x - X0; //* The x,y,z distances from the cell origin
            float y0 = y - Y0;
            float z0 = z - Z0;

            //* For the 3D case, the simplex shape is a slightly irregular tetrahedron.
            //* Determine which simplex we are in.
            int i1, j1, k1; //* Offsets for second corner of simplex in (i,j,k) coords
            int i2, j2, k2; //* Offsets for third corner of simplex in (i,j,k) coords

            /* This code would benefit from a backport from the GLSL version! */
            if (x0 >= y0)
            {
                if (y0 >= z0)
                { i1 = 1; j1 = 0; k1 = 0; i2 = 1; j2 = 1; k2 = 0; } //* X Y Z order
                else if (x0 >= z0) { i1 = 1; j1 = 0; k1 = 0; i2 = 1; j2 = 0; k2 = 1; } //* X Z Y order
                else { i1 = 0; j1 = 0; k1 = 1; i2 = 1; j2 = 0; k2 = 1; } //* Z X Y order
            }
            else
            { //* x0<y0
                if (y0 < z0) { i1 = 0; j1 = 0; k1 = 1; i2 = 0; j2 = 1; k2 = 1; } //* Z Y X order
                else if (x0 < z0) { i1 = 0; j1 = 1; k1 = 0; i2 = 0; j2 = 1; k2 = 1; } //* Y Z X order
                else { i1 = 0; j1 = 1; k1 = 0; i2 = 1; j2 = 1; k2 = 0; } //* Y X Z order
            }

            //* A step of (1,0,0) in (i,j,k) means a step of (1-c,-c,-c) in (x,y,z),
            //* a step of (0,1,0) in (i,j,k) means a step of (-c,1-c,-c) in (x,y,z), and
            //* a step of (0,0,1) in (i,j,k) means a step of (-c,-c,1-c) in (x,y,z), where
            //* c = 1/6.

            float x1 = x0 - i1 + G3; //* Offsets for second corner in (x,y,z) coords
            float y1 = y0 - j1 + G3;
            float z1 = z0 - k1 + G3;
            float x2 = x0 - i2 + 2.0f * G3; //* Offsets for third corner in (x,y,z) coords
            float y2 = y0 - j2 + 2.0f * G3;
            float z2 = z0 - k2 + 2.0f * G3;
            float x3 = x0 - 1.0f + 3.0f * G3; //* Offsets for last corner in (x,y,z) coords
            float y3 = y0 - 1.0f + 3.0f * G3;
            float z3 = z0 - 1.0f + 3.0f * G3;

            //* Wrap the integer indices at 256, to avoid indexing perm[] out of bounds
            int ii = Mod(i, 256);
            int jj = Mod(j, 256);
            int kk = Mod(k, 256);

            //* Calculate the contribution from the four corners
            float t0 = 0.6f - x0 * x0 - y0 * y0 - z0 * z0;
            if (t0 < 0.0f) n0 = 0.0f;
            else
            {
                t0 *= t0;
                n0 = t0 * t0 * grad(perm[ii + perm[jj + perm[kk]]], x0, y0, z0);
            }

            float t1 = 0.6f - x1 * x1 - y1 * y1 - z1 * z1;
            if (t1 < 0.0f) n1 = 0.0f;
            else
            {
                t1 *= t1;
                n1 = t1 * t1 * grad(perm[ii + i1 + perm[jj + j1 + perm[kk + k1]]], x1, y1, z1);
            }

            float t2 = 0.6f - x2 * x2 - y2 * y2 - z2 * z2;
            if (t2 < 0.0f) n2 = 0.0f;
            else
            {
                t2 *= t2;
                n2 = t2 * t2 * grad(perm[ii + i2 + perm[jj + j2 + perm[kk + k2]]], x2, y2, z2);
            }

            float t3 = 0.6f - x3 * x3 - y3 * y3 - z3 * z3;
            if (t3 < 0.0f) n3 = 0.0f;
            else
            {
                t3 *= t3;
                n3 = t3 * t3 * grad(perm[ii + 1 + perm[jj + 1 + perm[kk + 1]]], x3, y3, z3);
            }

            //* Add contributions from each corner to get the final noise value.
            //* The result is scaled to stay just inside [-1,1]
            return 32.0f * (n0 + n1 + n2 + n3); //* TODO: The scale factor is preliminary!
        }

        public static byte[] perm = 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
            };

        private static int FastFloor(float x)
        {
            return (x > 0) ? ((int)x) : (((int)x) - 1);
        }

        private static int Mod(int x, int m)
        {
            int a = x % m;
            return a < 0 ? a + m : a;
        }

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

        private static float grad(int hash, float x, float y)
        {
            int h = hash & 7;      //* Convert low 3 bits of hash code
            float u = h < 4 ? x : y;  //* into 8 simple gradient directions,
            float v = h < 4 ? y : x;  //* and compute the dot product with (x,y).
            return ((h & 1) != 0 ? -u : u) + ((h & 2) != 0 ? -2.0f * v : 2.0f * v);
        }

        private static float grad(int hash, float x, float y, float z)
        {
            int h = hash & 15;     //* Convert low 4 bits of hash code into 12 simple
            float u = h < 8 ? x : y; //* gradient directions, and compute dot product.
            float v = h < 4 ? y : h == 12 || h == 14 ? x : z; //* Fix repeats at h = 12 to 15
            return ((h & 1) != 0 ? -u : u) + ((h & 2) != 0 ? -v : v);
        }

        private static float grad(int hash, float x, float y, float z, float t)
        {
            int h = hash & 31;      //* Convert low 5 bits of hash code into 32 simple
            float u = h < 24 ? x : y; //* gradient directions, and compute dot product.
            float v = h < 16 ? y : z;
            float w = h < 8 ? z : t;
            return ((h & 1) != 0 ? -u : u) + ((h & 2) != 0 ? -v : v) + ((h & 4) != 0 ? -w : w);
        }
    }
}
```

7.113 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/LandGeneration/↩ Terrain.cs File Reference

Classes

- class BeeGame.Terrain.LandGeneration.Terrain

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

Namespaces

- namespace BeeGame.Terrain.LandGeneration

7.114 Terrain.cs

```
using System;
using UnityEngine;
using BeeGame.Terrain.Chunks;
using BeeGame.Blocks;
using BeeGame.Core;

namespace BeeGame.Terrain.LandGeneration
{
    public class Terrain
    {
        public static World world;

        #region Setting Position To block Grid
        public static ChunkWorldPos GetBlockPos(THVector3 pos)
        {
            return new ChunkWorldPos()
            {
                x = Mathf.RoundToInt(pos.x),
                y = Mathf.RoundToInt(pos.y),
                z = Mathf.RoundToInt(pos.z)
            };
        }

        public static THVector3 GetBlockPos(RaycastHit hit)
        {
            THVector3 vec3 = new THVector3()
            {
                x = RoundXZ(hit.point.x, hit.normal.x),
                y = RoundY(hit.point.y, hit.normal.y),
                z = RoundXZ(hit.point.z, hit.normal.z)
            };
            return (vec3);
        }

        public static ChunkWorldPos GetBlockPosFromRayCast(RaycastHit
hit)
        {
            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));
        }

        static float RoundXZ(float pos, float normal)
        {
            //* if we are looking at + x/z vlaues
            if (pos > 0)
            {
                if (normal > 0)
                {
                    pos = (int)pos;
                    return pos;
                }
                else if (normal < 0)
                {
                    pos = (int)pos;
                    return pos - -1;
                }
                else
                {
                    if ((pos - (int)pos) > 0.5)
                    {
                        return (int)pos + 1;
                    }
                    return (int)pos;
                }
            }
            //* if we are looking at - x/z values
            else
            {
                //* if poitive normal
                if (normal > 0)
                {
                    pos = (int)pos;
                    return pos - 1;
                }

                //* if negative nomrmal
                if (normal < 0)
                {
                    pos = (int)pos;
                    return pos;
                }
                //* if their is no normal

                //* if pos is greater than 0.5 we are in the next block so go to it
                if ((-pos - (int)-pos) > 0.5)
                {
                    return (int)pos - 1;
                }

                return (int)pos;
            }
        }

        static float RoundY(float pos, float normal)
        {
            pos = (float)Math.Round(pos, 1);
            if (pos >= 0)
            {
                if(normal > 0)
                {
                    if((int)pos % 2 == 0)
                        return Mathf.RoundToInt((float)Math.Round(pos, 1));

                    return Mathf.RoundToInt((float)Math.Round(pos, 1)) - normal;
                }

                if((int)pos % 2 == 0)
                    return Mathf.RoundToInt((float)Math.Round(pos, 1)) - normal;

                return Mathf.RoundToInt((float)Math.Round(pos, 1));
            }

            if(pos <= 0)
            {
                if (normal > 0)
                {
                    if ((int)pos % 2 == 0)
                        //* the Math.Round removes strange rounding errors shown with Mathf.Round eg sometimes 0.5 would round to 0 not 1
                        return Mathf.RoundToInt((float)Math.Round(pos, 1)) - normal;

                    return Mathf.RoundToInt((float)Math.Round(pos, 1));// - normal;
                }

                if ((int)pos % 2 == 0)
                    return Mathf.RoundToInt((float)Math.Round(pos, 1));

                return Mathf.RoundToInt((float)Math.Round(pos, 1)) - normal;
            }


            return Mathf.RoundToInt((float)Math.Round(pos, 1));
        }

        public static float Round(float pos, float norm, bool adjacent = false)
        {
            if(pos - (int)pos == 0.5f || pos - (int)pos == -0.5f)
            {
                if(adjacent)
                {
                    pos += (norm / 2);
                }
                else
                {
                    pos -= (norm / 2);
                }
            }

            return pos;
        }
        #endregion

        #region Get Block
        public static ChunkWorldPos GetBlockPos(RaycastHit hit, bool adjacent = false)
        {
            return GetBlockPos(new THVector3()
            {
                //* rounds the hit to the correct position
                x = Round(hit.point.x, hit.normal.x, adjacent),
                y = Round(hit.point.y, hit.normal.y, adjacent),
                z = Round(hit.point.z, hit.normal.z, adjacent)
            });
        }

        public static Block GetBlock(RaycastHit hit, bool adjacent = false)
        {
            //* checks that a chunk was hit and if it wasnt return early
            Chunk chunk = hit.collider.GetComponent<Chunk>();

            if (chunk == null)
                return null;

            //* allignes the hit to the block grid and returns the block
            ChunkWorldPos pos = GetBlockPos(hit, adjacent);

            return chunk.world.GetBlock(pos.x, pos.y, pos.z);
        }

        public static Block GetBlock(THVector3 pos)
        {
            Chunk chunk = GetChunk(pos);

            if (chunk == null)
                return new Air();

            chunk.world.GetBlock((int)pos.x, (int)pos.y, (int)pos.z);

            return new Block();
        }

        public static bool BlockInPosition(THVector3 pos, Chunk chunk)
        {
            if (chunk == null)
                return false;

            if (chunk.GetBlock((int)pos.x, (int)pos.y, (int)pos.z) != new
Air())
                return true;

            return false;
        }
        #endregion

        public static Chunk GetChunk(THVector3 vec3)
        {
            return world.GetChunk((int)vec3.x, (int)vec3.y, (int)vec3.
z);
        }

        #region Set Block
        public static bool SetBlock(RaycastHit hit, Block block, bool adjacent = false)
        {
            //* checks that a chnk was hit
            Chunk chunk = hit.collider.GetComponent<Chunk>();

            if (chunk == null)
                return false;

            //* alligns the hit to the block grid
            ChunkWorldPos pos = GetBlockPosFromRayCast(hit);

            //* checks that the block tryign to be replaced can be replaced eg bedrock cannot be replaced
            if (GetBlock(hit, adjacent).breakable)
            {
                //* sets the position of the block and saves the chunk
                chunk.world.SetBlock(pos.x, pos.y, pos.z, block);
                Serialization.Serialization.SaveChunk(chunk);
            }

            return true;
        }
        #endregion
    }
}
```

7.115 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/LandGeneration/↩ TerrainGeneration.cs File Reference

Classes

- class BeeGame.Terrain.LandGeneration.TerrainGeneration

    Generates the terrain for the game

Namespaces

- namespace BeeGame.Terrain.LandGeneration

7.116 TerrainGeneration.cs

```
using UnityEngine;
using BeeGame.Terrain.Chunks;
using BeeGame.Terrain.LandGeneration.Noise;
using BeeGame.Serialization;
using System.Collections.Generic;
using System.Threading;

namespace BeeGame.Terrain.LandGeneration
{
    public class TerrainGeneration
    {
        #region Data
        private float stoneBaseHeight = -24;
        private float stoneBaseNoise = 0.05f;
        private float stoneBaseNoiseHeight = 4;

        private float stoneMountainHeight = 48;
        private float stoneMountainFrequency = 0.008f;
        private float stoneMinHeight = -12;

        private float dirtBaseHeight = 1;
        private float dirtNoise = 0.04f;
        private float dirtNoiseHeight = 3;

        private float treeFrequency = 0.2f;
        private int treeDensity = 3;

        private float caveFrequency = 0.025f;
        private int caveSize = 8;
        #endregion

        public Chunk ChunkGen(Chunk chunk)
        {
            Chunk outChunk = chunk;
            lock (chunk)
            {
                Thread thread = new Thread(() => ChunkGenThread(chunk, out outChunk)) { Name = $"Generate Chunk Thread @ {chunk.chunkWorldPos}"};

                thread.Start();
                return outChunk;
            }
        }

        public void ChunkGenThread(Chunk chunk, out Chunk outChunk)
        {
            //* for each x and z position in teh chunk
            for (int x = chunk.chunkWorldPos.x-3; x < chunk.chunkWorldPos.x + Chunk.chunkSize + 3; x++)
            {
                for (int z = chunk.chunkWorldPos.z-3; z < chunk.chunkWorldPos.z + Chunk.chunkSize + 3; z++)
                {
                    chunk = GenChunkColum(chunk, x, z);
                }
            }

            chunk.SetBlocksUnmodified();
            outChunk = chunk;
        }

        public Chunk GenChunkColum(Chunk chunk, int x, int z)
        {
            //* the height of the mountain
            int stoneHeight = Mathf.FloorToInt(stoneBaseHeight);
            stoneHeight += GetNoise(-x, 0, z, stoneMountainFrequency, Mathf.FloorToInt(stoneMountainHeight));

            //* if the colum is currenly to low make it not so low
            if (stoneHeight < stoneMinHeight)
                stoneHeight = Mathf.FloorToInt(stoneMinHeight);

            //* add the height of normal stone on to the mountain
            stoneHeight += GetNoise(x, 0, -z, stoneBaseNoise, Mathf.RoundToInt(stoneBaseNoiseHeight));

            //*put dirt on top
            int dirtHeight = stoneHeight + Mathf.FloorToInt(dirtBaseHeight);
            dirtHeight += GetNoise(x, 100, z, dirtNoise, Mathf.FloorToInt(dirtNoiseHeight));

            //* set the colum to the correct blocks
            for (int y = chunk.chunkWorldPos.y - 8; y < chunk.chunkWorldPos.y + Chunk.chunkSize; y ++)
            {
                int caveChance = GetNoise(x + 40, y + 100, z - 50, caveFrequency, 200);

                //* puts a layer of bedrock at the botton the the world
                if (y <= (chunk.chunkWorldPos.y) && chunk.chunkWorldPos.y == -16)
                {
                    SetBlock(x, y, z, new Blocks.Bedrock(), chunk);
                }
                else if (y <= stoneHeight && caveSize < caveChance)
                {
                    SetBlock(x, y, z, new Blocks.Block(), chunk);
                }
                else if (y <= dirtHeight && caveSize < caveChance)
                {
                    SetBlock(x, y, z, new Blocks.Grass(), chunk);
                    if (y == dirtHeight && GetNoise(x, 0, z, treeFrequency, 100) < treeDensity)
                        CreateTree(x, y + 1, z, chunk);
                }
                else
                {
                    SetBlock(x, y, z, new Blocks.Air(), chunk);
                }
            }
```

```

                return chunk;
        }

        public static int GetNoise(int x, int y, int z, float scale, int max)
        {
            return Mathf.FloorToInt((SimplexNoise.Generate(x * scale, y * scale, z * scale) + 1f) * (max / 2f));
        }

        public static void SetBlock(int x, int y, int z, Blocks.Block block, Chunk chunk, bool replacesBlocks = false)
        {
            //* corrects the x, y, z pos of the so that the block is placed in the correct position
            x -= chunk.chunkWorldPos.x;
            y -= chunk.chunkWorldPos.y;
            z -= chunk.chunkWorldPos.z;

            //* checks that the block is in the chunk and that no block is already their then sets it
            if (Chunk.InRange(x) && Chunk.InRange(y) && Chunk.InRange(z))
                if (replacesBlocks || chunk.blocks[x, y, z] == null)
                    chunk.SetBlock(x, y, z, block, false);
        }

        void CreateTree(int x, int y, int z, Chunk chunk)
        {
            //* makes the leaves of teh tree
            for (int xi = -2; xi <= 2; xi++)
            {
                for (int yi = 4; yi <= 8; yi++)
                {
                    for (int zi = -2; zi <= 2; zi++)
                    {
                        SetBlock(xi + x, yi + y, zi + z, new Blocks.Leaves(), chunk, true);
                    }
                }
            }

            //* makes the trunk of the tree
            for (int i = 0; i < 6; i++)
            {
                SetBlock(x, y + i, z, new Blocks.Wood(), chunk, true);
            }
        }
    }
}
```

7.117 C:/Users/Toothless/Documents/GitHub/BeeGame/Code/BeeGame/BeeGame/Terrain/LandGeneration/↩ World.cs File Reference

Classes

- class BeeGame.Terrain.LandGeneration.World

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

Namespaces

- namespace BeeGame.Terrain.LandGeneration

7.118 World.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using UnityEngine;
using BeeGame.Terrain.Chunks;
using BeeGame.Blocks;

namespace BeeGame.Terrain.LandGeneration
{
    public class World : MonoBehaviour
    {
        #region Data
        public Dictionary<ChunkWorldPos, Chunk> chunks = new Dictionary<ChunkWorldPos, Chunk>();

        public GameObject chunkPrefab;

        public bool chunkHasMadeCollisionMesh = false;
        #endregion

        #region Creation and Destruction
        #region Chunk
        public void CreateChunk(int x, int y, int z)
        {
            //* pos of the chunk
            ChunkWorldPos pos = new ChunkWorldPos(x, y, z);

            //* makes the chunk at the given position
            GameObject newChunk = Instantiate(chunkPrefab, new Vector3(x, y, z), Quaternion.identity);

            Chunk chunk = newChunk.GetComponent<Chunk>();

            //* setting the chunks pos and a reference to this
            chunk.chunkWorldPos = pos;
            chunk.world = this;

            //* adds the nwe chunk to the dictionary
            chunks.Add(pos, chunk);

            //* generates the new chunks blocks
            chunk = new TerrainGeneration().ChunkGen(chunk);

            //loads any blocks that the chunk has had modified
            Serialization.Serialization.LoadChunk(chunk);

            //* updates all chunks around this one to reduce drawing of unecisary faces
            chunks.TryGetValue(new ChunkWorldPos(x, y - 16, z), out chunk);
            if (chunk != null)
                chunk.update = true;

            chunks.TryGetValue(new ChunkWorldPos(x, y, z - 16), out chunk);
            if (chunk != null)
                chunk.update = true;

            chunks.TryGetValue(new ChunkWorldPos(x - 16, y, z), out chunk);
            if (chunk != null)
                chunk.update = true;

            chunks.TryGetValue(new ChunkWorldPos(x, y + 16, z), out chunk);
            if (chunk != null)
                chunk.update = true;

            chunks.TryGetValue(new ChunkWorldPos(x, y, z + 16), out chunk);
            if (chunk != null)
                chunk.update = true;

            chunks.TryGetValue(new ChunkWorldPos(x + 16, y, z), out chunk);
            if (chunk != null)
                chunk.update = true;
            //* the chunk will then make its meshes
        }

        public void DestroyChunk(int x, int y, int z)
        {
            //* if teh chnks exists destroy it
            if (chunks.TryGetValue(new ChunkWorldPos(x, y, z), out Chunk chunk))
            {
                //* saves the chunk before destroying it incase any block were changed in it
                Serialization.Serialization.SaveChunk(chunk);
                Destroy(chunk.gameObject);
                chunks.Remove(new ChunkWorldPos(x, y, z));
            }
        }
        #endregion

        #region Block
        public void SetBlock(int x, int y, int z, Block block, bool saveChunk = false)
        {
            //*gets the chunk for the block to be placed in
            Chunk chunk = GetChunk(x, y, z);

            //*if the chunk is not null and the block trying to be replaced is replaceable, replace it
            if(chunk != null && chunk.blocks[x - chunk.chunkWorldPos.x, y - chunk.chunkWorldPos.y, z - chunk.chunkWorldPos.z].breakable)
            {

                chunk.SetBlock(x - chunk.chunkWorldPos.x, y - chunk.chunkWorldPos.y, z - chunk.chunkWorldPos.z, block);
                chunk.update = true;

                //*updates the nebouring chunks as when a block is broken it may be in the edje of the chunk so their meshes also need to be updated
                //*only updates chunks that need to be updated as not every chunk will need to be and sometines none of them will need to be

                //*checks if the block chaged is in the edge if the x value for the chunk
                UpdateIfEqual(x - chunk.chunkWorldPos.x, 0, new ChunkWorldPos(x - 1, y, z));
                UpdateIfEqual(x - chunk.chunkWorldPos.x, Chunk.chunkSize - 1, new ChunkWorldPos(x + 1, y, z));

                //*checks if the block chaged is in the edge if the y value for the chunk
                UpdateIfEqual(y - chunk.chunkWorldPos.y, 0, new ChunkWorldPos(x, y - 1, z));
                UpdateIfEqual(y - chunk.chunkWorldPos.y, Chunk.chunkSize - 1, new ChunkWorldPos(x, y + 1, z));

                //*checks if the block chaged is in the edge if the z value for the chunk
                UpdateIfEqual(z - chunk.chunkWorldPos.z, 0, new ChunkWorldPos(x, y, z - 1));
                UpdateIfEqual(z - chunk.chunkWorldPos.z, Chunk.chunkSize - 1, new ChunkWorldPos(x, y, z + 1));

                if (saveChunk)
                    Serialization.Serialization.SaveChunk(chunk);
            }
        }
        #endregion
        #endregion

        #region Get Things
        public Chunk GetChunk(int x, int y, int z)
        {
            float multiple = Chunk.chunkSize;
            //* rounds the given x, y, z to a multiple of 16 as chunks are 16x16x16 in size
            ChunkWorldPos pos = new ChunkWorldPos()
            {
                x = Mathf.FloorToInt(x / multiple) * Chunk.chunkSize,
                y = Mathf.FloorToInt(y / multiple) * Chunk.chunkSize,
                z = Mathf.FloorToInt(z / multiple) * Chunk.chunkSize
            };

            //* gets the chunk if it exists
            chunks.TryGetValue(pos, out Chunk chunk);
            //* if the chunk does not exist will return null
            return chunk;
        }

        public Block GetBlock(int x, int y, int z)
        {
            //* gets the chunk that the block is in
            Chunk chunk = GetChunk(x, y, z);

            if(chunk != null)
            {
                //* gets the block in the chunk
                return chunk.GetBlock(x - chunk.chunkWorldPos.x, y - chunk.chunkWorldPos.y, z - chunk.chunkWorldPos.z) ?? new Air();
            }

            //* returns an empty block is the chunk was not found
            return new Air();
        }
        #endregion

        void UpdateIfEqual(int value1, int value2, ChunkWorldPos pos)
        {
            if(value1 == value2)
            {
                Chunk chunk = GetChunk(pos.x, pos.y, pos.z);

                if (chunk != null)
                    chunk.update = true;
            }
        }
    }
}
```

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

Classes

- class BeeGame.Test

Namespaces

- namespace BeeGame

7.120 test.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using UnityEngine;
using UnityEngine.UI;
using BeeGame.Core.Dictionarys;
using BeeGame.Items;
using BeeGame.Blocks;
using BeeGame.Core;

namespace BeeGame
{
    public class Test : MonoBehaviour
    {
        private void Start()
        {
            CraftingRecipies.AddShapedRecipie(new object[] { "   ", " X ", "   ", "X", Dirt.ID }, new Grass());
            CraftingRecipies.AddShaplessRecipie(new object[] { new Grass(), 1 }, new Dirt());

            Events.shapedRecipieCrafted += Print;
        }
        public void Print(Item item) => print(item.GetItemID());
    }
}
```

Index