

My Project

Generated by Doxygen 1.8.16

1 Main Page	1
1.1 Introduction	1
1.2 Contact	1
1.3 First Steps	1
2 Getting Started	3
2.1 Some Notes on the base of the Framework	3
2.1.1 CellData	3
2.1.2 Cells	3
2.1.3 ChunkLoader	3
2.1.4 Chunk	3
2.1.5 TerrainShaderData	3
2.1.6 CellMap	4
2.2 Using the Cell Editor	4
2.2.1 Block Data	4
2.2.2 Block and Background menu	5
3 Lighting	7
3.1 Light Sources	7
3.2 Shaders	7
4 MapGenerator	9
4.1 Creating Passes	9
4.1.1 Settings	10
5 Namespace Documentation	11
5.1 Structure2D Namespace Reference	11
5.1.1 Enumeration Type Documentation	12
5.1.1.1 CoordinateAnchor	12
5.2 Structure2D.Base Namespace Reference	12
5.3 Structure2D.Base.Utility Namespace Reference	12
5.4 Structure2D.Lighting Namespace Reference	12
5.5 Structure2D.MapGeneration Namespace Reference	13
5.6 Structure2D.MapGeneration.BasePasses Namespace Reference	13
5.7 Structure2D.Utility Namespace Reference	14
5.7.1 Enumeration Type Documentation	14
5.7.1.1 Direction	14
5.8 Structure2D.Utility.MapGeneration Namespace Reference	14
6 Class Documentation	15
6.1 Structure2D.Background Class Reference	15
6.1.1 Detailed Description	15
6.1.2 Member Data Documentation	16
6.1.2.1 BlocksSunLight	16

6.1.2.2 LightBlockAmount	16
6.1.2.3 Texture	16
6.1.3 Property Documentation	16
6.1.3.1 ID	16
6.2 Structure2D.MapGeneration.BasePassSubscriber Struct Reference	16
6.2.1 Detailed Description	17
6.3 Structure2D.Block Class Reference	17
6.3.1 Detailed Description	17
6.3.2 Member Data Documentation	17
6.3.2.1 IsSolid	18
6.3.2.2 LightBlockAmount	18
6.3.2.3 Texture	18
6.3.3 Property Documentation	18
6.3.3.1 ID	18
6.4 Structure2D.Lighting.BlockLighting Class Reference	18
6.4.1 Detailed Description	19
6.4.2 Member Function Documentation	19
6.4.2.1 AddTemporaryLight()	19
6.4.3 Member Data Documentation	19
6.4.3.1 MaxLightDistance	19
6.5 Structure2D.BlockSolidStateMetaData Class Reference	20
6.5.1 Detailed Description	20
6.5.2 Member Function Documentation	20
6.5.2.1 EnumerateBlockInitialization()	20
6.5.2.2 RegisterForBlockInitialization()	21
6.6 Structure2D.MapGeneration.BlockSpawnConditionChecker Class Reference	21
6.6.1 Detailed Description	21
6.6.2 Member Function Documentation	21
6.6.2.1 IsCellUsable()	21
6.7 Structure2D.MapGeneration.BlockSpawnData Class Reference	22
6.7.1 Detailed Description	22
6.8 Structure2D.MapGeneration.BlockSpawner Class Reference	22
6.8.1 Detailed Description	23
6.9 Structure2D.Cell Class Reference	23
6.9.1 Detailed Description	23
6.9.2 Member Data Documentation	23
6.9.2.1 Chunk	23
6.9.2.2 Coordinate	24
6.9.3 Property Documentation	24
6.9.3.1 Background	24
6.9.3.2 Block	24
6.10 Structure2D.CellData Class Reference	24

6.10.1 Detailed Description	25
6.10.2 Member Data Documentation	25
6.10.2.1 BaseBlock	25
6.10.2.2 CustomBackgrounds	25
6.10.2.3 CustomBlocks	26
6.10.2.4 DefaultBackground	26
6.10.2.5 DefaultBlock	26
6.11 Structure2D.CellExtensions Class Reference	26
6.11.1 Member Function Documentation	26
6.11.1.1 GetNeighbor()	26
6.12 Structure2D.CellMap Class Reference	27
6.12.1 Detailed Description	28
6.12.2 Member Function Documentation	28
6.12.2.1 GetCell() [1/2]	28
6.12.2.2 GetCell() [2/2]	28
6.12.2.3 GetCellAtWorldPoint()	28
6.12.2.4 GetCellUnsafe()	29
6.12.2.5 GetChunkAtOffset()	29
6.12.2.6 MousePositionToChunk()	29
6.12.2.7 ScreenPositionToCell()	30
6.12.2.8 WorldPointToChunk()	30
6.12.3 Member Data Documentation	30
6.12.3.1 MapInitialized	30
6.12.3.2 MapUnloaded	31
6.12.4 Property Documentation	31
6.12.4.1 IsMapHidden	31
6.12.4.2 MapHeight	31
6.12.4.3 MapWidth	31
6.13 Structure2D.CellMetrics Class Reference	31
6.13.1 Detailed Description	32
6.13.2 Member Data Documentation	32
6.13.2.1 CellSize	32
6.13.2.2 ChunkSize	32
6.14 Structure2D.CellObjectLoader Class Reference	32
6.15 Structure2D.Chunk Class Reference	32
6.15.1 Detailed Description	33
6.15.2 Member Function Documentation	33
6.15.2.1 GenerateColliders()	33
6.15.3 Property Documentation	33
6.15.3.1 CellOffset	33
6.15.3.2 IsVisible	33
6.15.3.3 Offset	34

6.16 Structure2D.ChunkLoader Class Reference	34
6.16.1 Detailed Description	34
6.16.2 Member Data Documentation	35
6.16.2.1 ChunkLoaded	35
6.16.2.2 ChunkUnloaded	35
6.16.2.3 Viewer	35
6.16.3 Property Documentation	35
6.16.3.1 DesiredViewPortSizeX	35
6.16.3.2 DesiredViewPortSizeY	35
6.17 Structure2D.Coordinate Struct Reference	36
6.17.1 Member Function Documentation	36
6.17.1.1 DistanceTo()	36
6.17.1.2 FromScreenPoint()	37
6.17.1.3 GetIndex()	37
6.17.1.4 ToIndex()	37
6.18 Structure2D.Base.Utility.DebugUtility Class Reference	38
6.18.1 Detailed Description	38
6.18.2 Member Function Documentation	38
6.18.2.1 SetDebugTarget()	38
6.18.3 Property Documentation	39
6.18.3.1 DoDebug	39
6.19 Structure2D.MapGeneration.BasePasses.ExamplePassSubscriber Class Reference	39
6.20 GrassBlockSpawnPass Class Reference	39
6.20.1 Member Function Documentation	39
6.20.1.1 Apply()	39
6.21 Structure2D.MapGeneration.IGenerationPassSubscriber Interface Reference	40
6.22 Structure2D.LightingMetaData Class Reference	40
6.22.1 Detailed Description	41
6.22.2 Member Function Documentation	41
6.22.2.1 EnumerateBackgroundInitialization()	41
6.22.2.2 EnumerateBlockInitialization()	41
6.22.2.3 RegisterForBackgroundInitialization()	41
6.22.2.4 RegisterForBlockInitialization()	42
6.23 Structure2D.Lighting.LightMap Class Reference	42
6.23.1 Detailed Description	42
6.24 Structure2D.Utility.ListPool< T > Class Template Reference	42
6.24.1 Detailed Description	43
6.24.2 Member Function Documentation	43
6.24.2.1 Add()	43
6.24.2.2 Get()	43
6.25 Structure2D.Utility.MapData Struct Reference	43
6.26 Structure2D.MapGeneration.MapGenerationPass Class Reference	44

6.26.1 Detailed Description	44
6.26.2 Member Function Documentation	44
6.26.2.1 Apply()	44
6.26.2.2 GetWeight()	44
6.26.2.3 PrepareGeneration()	45
6.27 Structure2D.MapGeneration.MapGenerator Class Reference	45
6.27.1 Detailed Description	46
6.27.2 Member Function Documentation	46
6.27.2.1 GenerateMap() [1/2]	47
6.27.2.2 GenerateMap() [2/2]	47
6.27.2.3 GetRandomSolidCell()	47
6.27.2.4 IsBlockInDesiredHeight()	47
6.27.2.5 SpawnGameObject()	47
6.27.3 Member Data Documentation	48
6.27.3.1 FinishedMapGeneration	48
6.27.3.2 PreparePasses	48
6.27.3.3 StartedMapGeneration	48
6.27.4 Property Documentation	48
6.27.4.1 AirChunkCellHeight	48
6.27.4.2 BaseChunkCellHeight	49
6.27.4.3 GenerationProgress	49
6.27.4.4 GroundCellHeight	49
6.27.4.5 GroundCellStart	49
6.27.4.6 MapGenRandom	49
6.27.4.7 MapHeight	49
6.27.4.8 MapWidth	50
6.27.4.9 TerrainCellHeight	50
6.27.4.10 UsableDefaultBlocks	50
6.28 Structure2D.MapGeneration.MapGeneratorSettings Class Reference	50
6.28.1 Detailed Description	51
6.28.2 Member Data Documentation	51
6.28.2.1 MapWidth	51
6.28.2.2 NoiseMapXScale	51
6.28.2.3 PassSubscribers	51
6.28.2.4 Seed	52
6.29 Structure2D.MapGeneration.MapGeneratorUi Class Reference	52
6.30 Structure2D.MetaDataBaseClass Class Reference	52
6.30.1 Detailed Description	52
6.30.2 Member Function Documentation	53
6.30.2.1 EnumerateBackgroundInitialization()	53
6.30.2.2 EnumerateBlockInitialization()	53
6.30.2.3 RegisterForBackgroundInitialization()	53

6.30.2.4 RegisterForBlockInitialization()	54
6.31 Structure2D.MapGeneration.NoiseMap Class Reference	54
6.31.1 Member Function Documentation	54
6.31.1.1 SetCustomNoiseMap()	54
6.32 Structure2D.MapGeneration.BasePasses.SurfaceObjectSpawnPass.ObjectSpawnData Struct Reference	54
6.33 Structure2D.MapGeneration.MapGenerator.ObjectSpawnQueueData Class Reference	55
6.33.1 Detailed Description	55
6.33.2 Member Data Documentation	55
6.33.2.1 ObjectToSpawn	55
6.33.2.2 Parent	55
6.33.2.3 Position	56
6.33.2.4 Rotation	56
6.33.2.5 Types	56
6.33.3 Property Documentation	56
6.33.3.1 SpawnedObject	56
6.34 Structure2D.MapGeneration.BasePasses.PlayerSpawnPass Class Reference	56
6.34.1 Member Function Documentation	57
6.34.1.1 Apply()	57
6.34.1.2 GetWeight()	57
6.34.1.3 PrepareGeneration()	57
6.35 Structure2D.Utility.MapGeneration.PriorityQueue Class Reference	58
6.35.1 Detailed Description	58
6.35.2 Member Function Documentation	58
6.35.2.1 Clear()	58
6.35.2.2 Dequeue()	59
6.35.2.3 Enqueue()	59
6.35.3 Member Data Documentation	59
6.35.3.1 Count	59
6.36 Structure2D.Utility.MapGeneration.PriorityQueue.QueueData Struct Reference	59
6.37 Structure2D.Utility.SaveManager Class Reference	59
6.37.1 Detailed Description	60
6.37.2 Member Function Documentation	60
6.37.2.1 LoadMapFromStream()	60
6.37.2.2 SaveMapToStream()	60
6.37.3 Member Data Documentation	62
6.37.3.1 OnLoadedCell	62
6.37.3.2 OnSavedCell	62
6.38 Structure2D.MapGeneration.ScriptableGenerationPassSubscriber Class Reference	62
6.38.1 Detailed Description	62
6.39 Structure2D.MapGeneration.BasePasses.SurfaceObjectSpawnPass Class Reference	63
6.39.1 Member Function Documentation	64

6.39.1.1 Apply()	64
6.39.1.2 GetRandomSpawnPoints()	65
6.39.1.3 PrepareGeneration()	65
6.39.1.4 SpawnObject()	65
6.40 Structure2D.MapGeneration.TerrainNoise Class Reference	65
6.40.1 Detailed Description	66
6.40.2 Member Function Documentation	66
6.40.2.1 GetTerrain()	66
6.41 Structure2D.TerrainShaderExtensions Class Reference	66
6.42 Structure2D.MapGeneration.BasePasses.TreeSpawnPass Class Reference	67
6.42.1 Member Function Documentation	67
6.42.1.1 PrepareGeneration()	68
6.42.1.2 SpawnObject()	68
6.42.2 Member Data Documentation	68
6.42.2.1 MaxTreeParts	68
6.42.2.2 MinTreeParts	68
6.43 Structure2D.Viewport Struct Reference	68
6.43.1 Member Function Documentation	69
6.43.1.1 ContainsCoordinate()	69
6.43.2 Member Data Documentation	69
6.43.2.1 BottomLeft	69
6.43.2.2 Height	70
6.43.2.3 Width	70
6.43.3 Property Documentation	70
6.43.3.1 CurrentViewport	70
Index	71

Chapter 1

Main Page

1.1 Introduction

Structure 2D is a Framework to create Cell Map based games like **Terraria** or **Starbound**.

The main reason I have built Structure 2D was the lack of high performance and easy to use frameworks on the market. As a matter of fact the base framework of [Structure2D](#) has zero runtime GC allocation.

The most notable **features** are:

- Super Easy API
- 2D Collider Generation
- Cell Lighting
- Editor to easily modify Cells
- Easy Pass based Map Generator
- Lots of examples
 1. Save Manager
 2. Map Editor
 3. Map Generation loading Screen
 4. ...

1.2 Contact

If you have any **questions**, **concerns**, **feature requests** or **bug reports** feel free to contact me:

[support@structure2D.com](mailto:support@structure2d.com)

1.3 First Steps

I recommend importing the package in to an empty project so you can dissect the demo scene.

Chapter 2

Getting Started

2.1 Some Notes on the base of the Framework

The following sections briefly describe some foundational **concepts and classes** of the Framework.

2.1.1 CellData

The [CellData](#) stores the information of the [Blocks](#) and [Backgrounds](#).
It handles the initialization of the texture arrays that you will use inside the **Shaders**.

2.1.2 Cells

A [Cell](#) is an entity on the Map that stores the information of its current Block and Background.

2.1.3 ChunkLoader

The [ChunkLoader](#) displays the part of the **CellMap** that the [Viewer](#) can currently see.
This is the only **Component** that you have to add to a GameObject in your scene for the base framework to function.

2.1.4 Chunk

[Chunks](#) is the method used to divide the **Cells** of the **CellMap** into managable pieces.
These pieces are responsible for mapping the **Block UVs** and generating the **Colliders**.

2.1.5 TerrainShaderData

The **TerrainShaderData** is a **MonoBehavior** which handles the drawing of the **Cells**.
You don't have to add this **Component** to a GameObject. This is done on runtime for you.

2.1.6 CellMap

As the name indicates the [CellMap](#) stores the Cells.
It also provides convenient methods of [fetching](#) Cells.

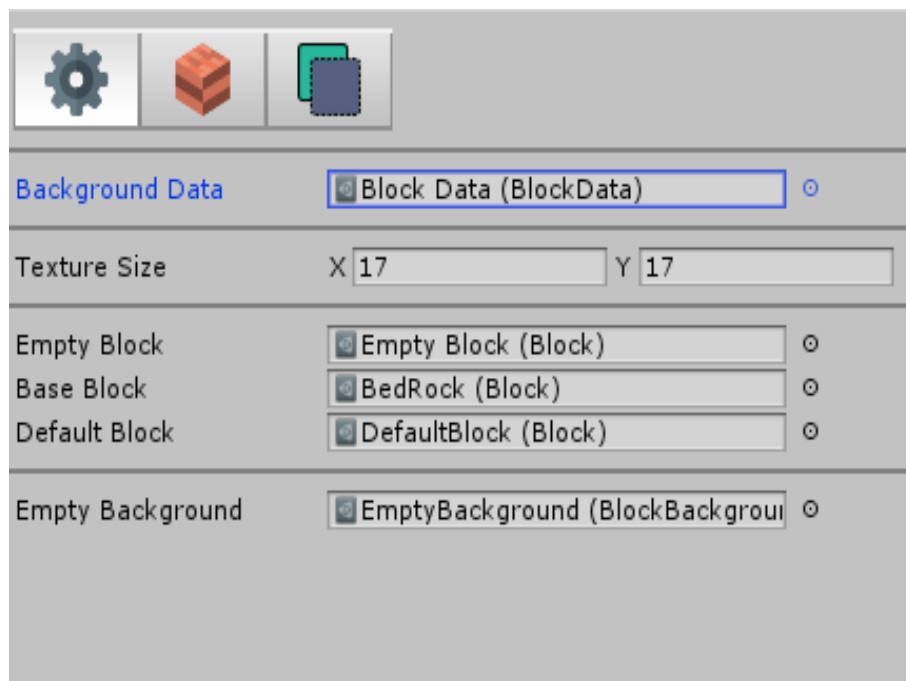
2.2 Using the Cell Editor

The Cell Editor gives you an easy interface of managing your **BlockData**, **Blocks** and **Backgrounds**.

After importing the package you can find it under: **Window->Structure2D->Editor**

2.2.1 Block Data

Setup the BlockData in the first menu.



Block Data: This field holds the Block Data that you use to edit, the one that you edit automatically becomes active one.

Texture Size: The dimensions that all textures of the **Blocks** and **Backgrounds** should have.

The following few Blocks and Backgrounds are necessary to the framework.

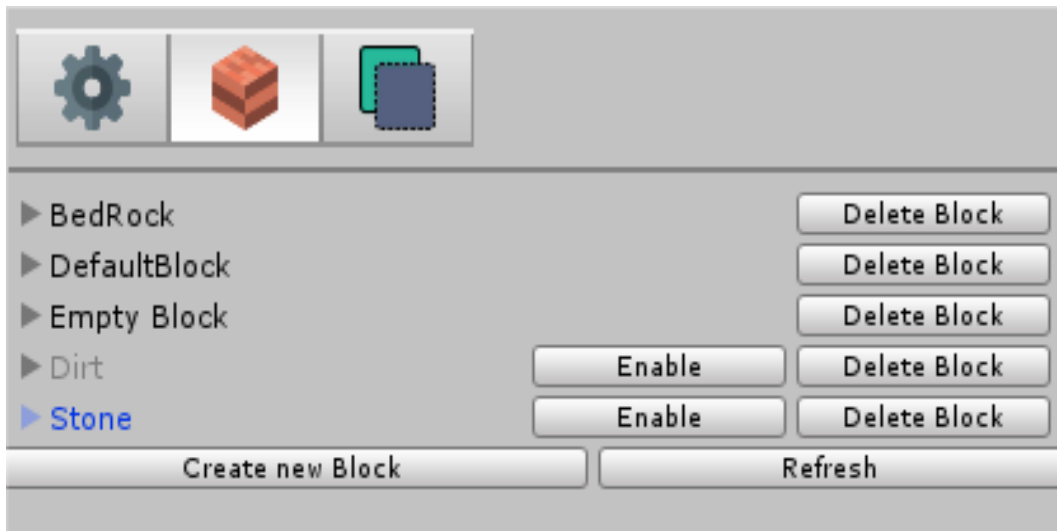
Empty Block/Background: This one is supposed to represent \diamond air \diamond .

Default Block/Background Every Cell has this one as its default Block/Background.

Base Block: The Base Block should be used as an unbreakable Block which gets placed on the bottom of the generated Map.

You probably know this one from Minecraft as Bedrock.

2.2.2 Block and Background menu

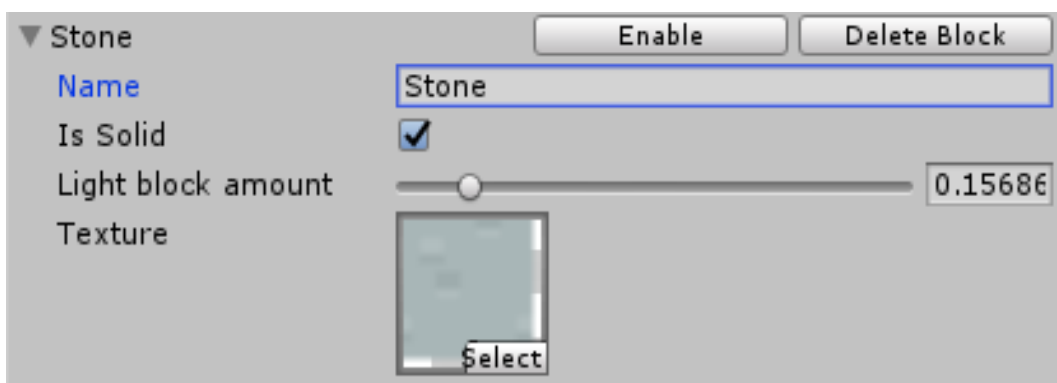


The **Block and Background Menus** are equivalent in functionality.

We can **enable**, **disable**, **create new** or **delete** entries through the editor.

Entries have in some situations other **label** colors:

- **Default:** Black
- **Error:** Red
- **Disabled:** Gray



Is Solid: Specifies whether colliders should be generated on a Cell that has this type of Block.

Light Block Amount: This is the amount of Light that gets used when light travels through this Block.

Texture This is the texture that this Block/Background should use.

Chapter 3

Lighting

Lighting in [Structure2D](#) is Cell based, this means that every cell on the map has a light value. This light value is represented as a byte where **0** is black and **255** is lit.

3.1 Light Sources

Light sources can be added by calling `AddTempLight`, this has to be called every frame since the `TempLights` get cleared in the Late Update of the [BlockLighting](#).

3.2 Shaders

The lighting data of the viewport is stored as a global **texture** called `_Celldata`.

Keep in mind that this is only the lighting data of the current viewport.

The **Texel Size** of the lighting data is a global **Vector4** named `_CellData_TexelSize`.

`CellData.cginc` demonstrates how to fetch the lighting data for the Cell of a Chunk.

Since the UVs of the Cells are not in Viewport space but in World Space we have to transform it with the global integer `_ViewerPosition`, this is the current one dimensional Cell Coordinate of the viewer.

Chapter 4

MapGenerator

The Map Generator is a **pass** based generator.

All generation specific logic is inside the passes, this means that all the generator has to know of is the passes which then generate the Map.

4.1 Creating Passes

To create a pass you have to inherit from the [MapGenerationPass](#).

Inside the newly created class you override the [Apply](#) method to add your own logic.

4.1.1 Settings

Script	MapGeneratorSettings
Noise Map X Scale	8
Noise Map Y Scale	1
Map Width	500
Map Height	102
Air Chunks	5
Base Chunks	2
Ground Chunks	92
Seed	-1
► Pass Subscribers	

NoiseMap Scale: This is used to scale the noise map of the base terrain pass.

Map Width: Desired width of the generated Map in Chunks.

Map Height: Desired height of the generated Map in Chunks.

Seed: This is the seed with which the [MapGenRandom](#) gets initialized, if it's -1 a random one will be generated.

Pass Subscribers: This is an array for adding [Scriptable Pass Subscribers](#) to the Map Generation.

The following is the amount of Chunks in height for the Base Generation Passes:

Base Chunks: These are Chunks that are at the ground of the Map, see Base Block for more informations.

Default Chunks: After the BaseChunks we add this amount of DefaultChunks to the Map.

Terrain Chunks: On top of the BaseChunks the TerrainChunks are added, these are generated by the active Noise map.

Note: The amount of Terrain Chunks is specified by the unused Chunks of the MapHeight.

Air Chunks: After we have generated the first base passes, we add the specified amount of empty Chunks on top.

Chapter 5

Namespace Documentation

5.1 Structure2D Namespace Reference

Classes

- class [Background](#)
[Background](#) used by the [BlockData](#) to create texture and meta data from.
- class [Block](#)
Class used by the [BlockData](#) to create texture and meta data from.
- class [BlockSolidStateMetaData](#)
Meta Data which Handles the solid state of Blocks.
- class [Cell](#)
Representation of a Block/Background inside the Map.
- class [CellData](#)
Class which handles the creation of the Texture Arrays and Meta Data.
- class [CellExtensions](#)
- class [CellMap](#)
This is the storage class for all the Cells. It also provides convenient methods of fetching Cells.
- class [CellMetrics](#)
Class which holds constant data about the Cells and Chunks.
- class [CellObjectLoader](#)
- class [Chunk](#)
Chunks handle the drawing and collider generation of Cells.
- class [ChunkLoader](#)
The [ChunkLoader](#) handles the Loading/Unloading of Chunks based on the current Viewer Position
- class **ChunkMaterial**
- class **ColliderGenerator**
- struct [Coordinate](#)
- class [LightingMetaData](#)
[Lighting](#) Meta Data stores information about the lighting data of Blocks and Backgrounds.
- class [MetaDataBaseClass](#)
[Base](#) class for implementing your own Meta Data.
- class **MetaDataManager**
This class handles the initialization of block meta data.
- class **TerrainShaderData**
This class handles the texture which represents the block data inside our Shader. It also draws the Chunks.
- class [TerrainShaderExtensions](#)
- struct [Viewport](#)

Enumerations

- enum [CoordinateAnchor](#) {
UpperLeft, UpperCenter, UpperRight, MiddleLeft,
MiddleCenter, MiddleRight, LowerLeft, LowerCenter,
LowerRight }

Used to transform a coordinate into WorldSpace.

5.1.1 Enumeration Type Documentation

5.1.1.1 CoordinateAnchor

```
enum Structure2D.CoordinateAnchor [strong]
```

Used to transform a coordinate into WorldSpace.

5.2 Structure2D.Base Namespace Reference

5.3 Structure2D.Base.Utility Namespace Reference

Classes

- class [DebugUtility](#)

Used to log additional debug information.

5.4 Structure2D.Lighting Namespace Reference

Classes

- class [BlockLighting](#)

This class handles the lighting of blocks

- class [LightMap](#)

Used to store lighting data for all Cells of the current Map.

5.5 Structure2D.MapGeneration Namespace Reference

Classes

- struct [BasePassSubscriber](#)
Pass subscriber which has the base passes.
- class [BlockSpawnConditionChecker](#)
Default spawn condition checker for Blocks.
- class [BlockSpawnData](#)
Data about how a given [Block](#) should be spawned. Used by adding a instance of this to a [Block](#) Spawner
- class [BlockSpawner](#)
A Pass subscriber which can be used to Spawn Blocks.
- interface [IGenerationPassSubscriber](#)
- class [MapGenerationPass](#)
Base class for the passes of the Map Generator
- class [MapGenerator](#)
Default Map Generator.
- class [MapGeneratorSettings](#)
This is the settings that you pass to the [MapGenerator](#) so he knows how to generate the Map.
- class [MapGeneratorUi](#)
- class [NoiseMap](#)
- class [ScriptableGenerationPassSubscriber](#)
Subscriber which you can add to the Map Generator to run your own MapGenPasses.
- class [TerrainNoise](#)
This is the Noise base class To add your custom Noise Map all you have to do is override the GetTerrain function and call SetCustomNoiseMap on the [NoiseMap](#) class

5.6 Structure2D.MapGeneration.BasePasses Namespace Reference

Classes

- class **AirChunkPass**
This pass generates Air chunks which are used as a buffer on top of the generated map to let the player build upwards
- class **BaseChunkPass**
Pass which generates Chunks of the base block type.
- class **DefaultBlockSpawnPass**
This pass generates veins for the given [Block](#).
- class [ExamplePassSubscriber](#)
- class **GroundChunkPass**
This pass generates the ground chunks.
- class [PlayerSpawnPass](#)
- class [SurfaceObjectSpawnPass](#)
- class **TerrainChunkPass**
This is the pass that uses the Noise Map to generate a terrain on top of the ground chunks
- class [TreeSpawnPass](#)

5.7 Structure2D.Utility Namespace Reference

Classes

- class **ChunkPool**
- class **ColliderPool**
- class [ListPool](#)
Generic Pool which you can use to reuse Lists.
- struct [MapData](#)
- class [SaveManager](#)
Utility class used to save/load the [CellMap](#) from a Stream.

Enumerations

- enum [Direction](#) {
Up, UpRight, Right, DownRight,
Down, DownLeft, Left, UpLeft }
8-Directional direction enum.

5.7.1 Enumeration Type Documentation

5.7.1.1 Direction

enum [Structure2D.Utility.Direction](#) [strong]

8-Directional direction enum.

5.8 Structure2D.Utility.MapGeneration Namespace Reference

Classes

- class [PriorityQueue](#)
Queue used to store Map Generation Cells based on their priority.

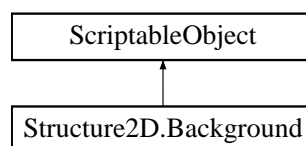
Chapter 6

Class Documentation

6.1 Structure2D.Background Class Reference

[Background](#) used by the BlockData to create texture and meta data from.

Inheritance diagram for Structure2D.Background:



Public Attributes

- byte [LightBlockAmount](#) = 20
Amount of light this [Background](#) blocks.
- Texture2D [Texture](#)
Texture of this [Background](#).
- bool [BlocksSunLight](#) = true
Does the [Background](#) block sunlight

Properties

- int [ID](#) = -1 [get, set]
This is the unique Identifier for this [Block](#). This gets set by the BlockData. If this is -1 the BlockID is not included in the BlockData.

6.1.1 Detailed Description

[Background](#) used by the BlockData to create texture and meta data from.

6.1.2 Member Data Documentation

6.1.2.1 BlocksSunLight

```
bool Structure2D.Background.BlocksSunLight = true
```

Does the [Background](#) block sunlight

6.1.2.2 LightBlockAmount

```
byte Structure2D.Background.LightBlockAmount = 20
```

Amount of light this [Background](#) blocks.

6.1.2.3 Texture

```
Texture2D Structure2D.Background.Texture
```

Texture of this [Background](#).

6.1.3 Property Documentation

6.1.3.1 ID

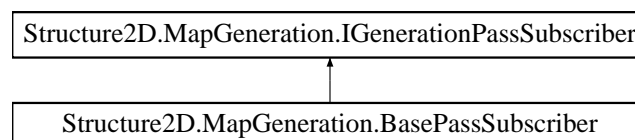
```
int Structure2D.Background.ID = -1 [get], [set]
```

This is the unique Identifier for this [Block](#). This gets set by the BlockData. If this is -1 the BlockID is not included in the BlockData.

6.2 Structure2D.MapGeneration.BasePassSubscriber Struct Reference

Pass subscriber which has the base passes.

Inheritance diagram for Structure2D.MapGeneration.BasePassSubscriber:



Public Member Functions

- [MapGenerationPass\[\]](#) **GetPasses** ()
- int **FetchPassOrder** ()
- int **PastProgressionWeight** ()

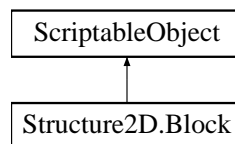
6.2.1 Detailed Description

Pass subscriber which has the base passes.

6.3 Structure2D.Block Class Reference

Class used by the BlockData to create texture and meta data from.

Inheritance diagram for Structure2D.Block:



Public Attributes

- Texture2D [Texture](#)
Texture of the [Block](#).
- byte [LightBlockAmount](#) = 40
Amount of Light this [Block](#) blocks;
- bool [IsSolid](#) = true
Should the [Block](#) be Solid.

Properties

- int [ID](#) = -1 [get, set]
This is the unique Identifier for this [Block](#). This gets set by the BlockData. If this is -1 the BlockID is not included in the BlockData.

6.3.1 Detailed Description

Class used by the BlockData to create texture and meta data from.

6.3.2 Member Data Documentation

6.3.2.1 IsSolid

```
bool Structure2D.Block.IsSolid = true
```

Should the [Block](#) be Solid.

6.3.2.2 LightBlockAmount

```
byte Structure2D.Block.LightBlockAmount = 40
```

Amount of Light this [Block](#) blocks;

6.3.2.3 Texture

```
Texture2D Structure2D.Block.Texture
```

Texture of the [Block](#).

6.3.3 Property Documentation

6.3.3.1 ID

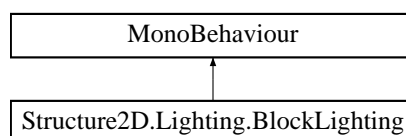
```
int Structure2D.Block.ID = -1 [get], [set]
```

This is the unique Identifier for this [Block](#). This gets set by the BlockData. If this is -1 the BlockID is not included in the BlockData.

6.4 Structure2D.Lighting.BlockLighting Class Reference

This class handles the lighting of blocks

Inheritance diagram for Structure2D.Lighting.BlockLighting:



Static Public Member Functions

- static void [AddTemporaryLight](#) ([Coordinate](#) lightPosition, float light)
Adds a Temporary Light at the given coordinate. Temporary Lights get cleared every LateUpdate, so it's necessary to add a new Light every Frame

Static Public Attributes

- const int [MaxLightDistance](#) = 60
This is the max distance in blocks to which a light source can emit light.

Properties

- static byte [SkyColor](#) = 255 [get, set]

6.4.1 Detailed Description

This class handles the lighting of blocks

6.4.2 Member Function Documentation

6.4.2.1 AddTemporaryLight()

```
static void Structure2D.Lighting.BlockLighting.AddTemporaryLight (
    Coordinate lightPosition,
    float light ) [inline], [static]
```

Adds a Temporary Light at the given coordinate. Temporary Lights get cleared every LateUpdate, so it's necessary to add a new Light every Frame

Parameters

<i>lightPosition</i>	Coordinate at which the light should be
<i>light</i>	The amount of light that the coordinate should emit in a range of 0 to 1

6.4.3 Member Data Documentation

6.4.3.1 MaxLightDistance

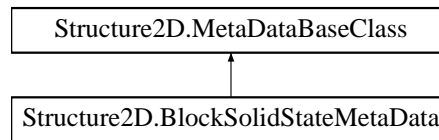
```
const int Structure2D.Lighting.BlockLighting.MaxLightDistance = 60 [static]
```

This is the max distance in blocks to which a light source can emit light.

6.5 Structure2D.BlockSolidStateMetaData Class Reference

Meta Data which Handles the solid state of Blocks.

Inheritance diagram for Structure2D.BlockSolidStateMetaData:



Public Member Functions

- override void [RegisterForBlockInitialization](#) (int size)
This will be called before we iterate over the blocks.
- override void [EnumerateBlockInitialization](#) ([Block](#) block)
This is called for every [Block](#) that we iterate over.

Static Public Attributes

- static bool[] [IsBlockSolid](#)

6.5.1 Detailed Description

Meta Data which Handles the solid state of Blocks.

6.5.2 Member Function Documentation

6.5.2.1 EnumerateBlockInitialization()

```

override void Structure2D.BlockSolidStateMetaData.EnumerateBlockInitialization (
    Block block ) [inline], [virtual]
  
```

This is called for every [Block](#) that we iterate over.

Parameters

<i>block</i>	The current Block .
--------------	-------------------------------------

Reimplemented from [Structure2D.MetaDataBaseClass](#).

6.5.2.2 RegisterForBlockInitialization()

```
override void Structure2D.BlockSolidStateMetaData.RegisterForBlockInitialization (
    int size ) [inline], [virtual]
```

This will be called before we iterate over the blocks.

Parameters

<code>size</code>	The amount of blocks we iterate over.
-------------------	---------------------------------------

Reimplemented from [Structure2D.MetaDataBaseClass](#).

6.6 Structure2D.MapGeneration.BlockSpawnConditionChecker Class Reference

Default spawn condition checker for Blocks.

Public Member Functions

- virtual bool [IsCellUsable](#) ([Cell](#) cell, [MapGenerator](#) mapGenerator)
Override this to create your own custom block spawn logic By default this returns false if the block is solid and doesn't to any additional checking

Public Attributes

- float [_minSpawnHeight](#)
- float [_maxSpawnHeight](#)

6.6.1 Detailed Description

Default spawn condition checker for Blocks.

6.6.2 Member Function Documentation

6.6.2.1 IsCellUsable()

```
virtual bool Structure2D.MapGeneration.BlockSpawnConditionChecker.IsCellUsable (
    Cell cell,
    MapGenerator mapGenerator ) [inline], [virtual]
```

Override this to create your own custom block spawn logic By default this returns false if the block is solid and doesn't to any additional checking

Parameters

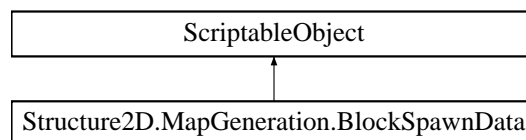
<i>cell</i>	
-------------	--

Returns

6.7 Structure2D.MapGeneration.BlockSpawnData Class Reference

Data about how a given [Block](#) should be spawned. Used by adding a instance of this to a [Block](#) Spawner

Inheritance diagram for Structure2D.MapGeneration.BlockSpawnData:



Public Attributes

- float **mapBudget**
- int **ID** => `_blockToSpawn.ID`
- [BlockSpawnConditionChecker](#) **BlockSpawnConditionChecker** = new [BlockSpawnConditionChecker](#)()

Properties

- int **MaxVeinSize** [get, set]
- int **MinVeinSize** [get, set]

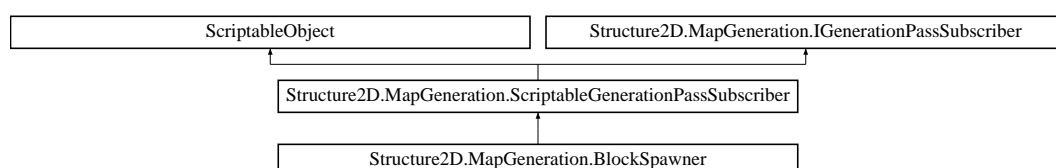
6.7.1 Detailed Description

Data about how a given [Block](#) should be spawned. Used by adding a instance of this to a [Block](#) Spawner

6.8 Structure2D.MapGeneration.BlockSpawner Class Reference

A Pass subscriber which can be used to Spawn Blocks.

Inheritance diagram for Structure2D.MapGeneration.BlockSpawner:



Public Member Functions

- override [MapGenerationPass\[\]](#) **GetPasses** ()
- override int **FetchPassOrder** ()

6.8.1 Detailed Description

A Pass subscriber which can be used to Spawn Blocks.

6.9 Structure2D.Cell Class Reference

Representation of a Block/Background inside the Map.

Public Attributes

- [Coordinate](#) [Coordinate](#)
[Coordinate](#) of this [Cell](#).
- [Chunk](#) [Chunk](#)
[Chunk](#) to which this cell is mapped currently. Do not set this directly, this gets set by the [Chunk](#).

Properties

- int [Block](#) [get, set]
[Block](#) ID of this [Cell](#).
- int [Background](#) [get, set]
[Background](#) ID of this [Cell](#).

6.9.1 Detailed Description

Representation of a Block/Background inside the Map.

6.9.2 Member Data Documentation

6.9.2.1 Chunk

[Chunk](#) `Structure2D.Cell.Chunk`

[Chunk](#) to which this cell is mapped currently. Do not set this directly, this gets set by the [Chunk](#).

6.9.2.2 Coordinate

[Coordinate](#) `Structure2D.Cell.Coordinate`

[Coordinate](#) of this [Cell](#).

6.9.3 Property Documentation

6.9.3.1 Background

```
int Structure2D.Cell.Background [get], [set]
```

[Background](#) ID of this [Cell](#).

6.9.3.2 Block

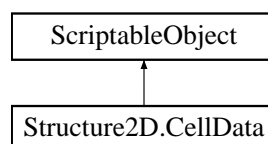
```
int Structure2D.Cell.Block [get], [set]
```

[Block](#) ID of this [Cell](#).

6.10 Structure2D.CellData Class Reference

Class which handles the creation of the Texture Arrays and Meta Data.

Inheritance diagram for Structure2D.CellData:



Public Member Functions

- bool **HasError** (out string errorCode)

Public Attributes

- bool **IsActive**
- [Block](#) **EmptyBlock**
- [Background](#) **EmptyBackground**
- Vector2 **TextureSize**
- [Block](#) **DefaultBlock**
This is the block with which the map gets filled initially
- [Background](#) **DefaultBackground**
This is the placeholder background with which the map gets filled initially
- [Block](#) **BaseBlock**
This blocks is a solid unbreakable block
- [Block](#)[] **CustomBlocks**
Here you can add your own blocks
- [Background](#)[] **CustomBackgrounds**
Here you can add all the [Background](#) CustomBackgrounds

Properties

- Material **BlockMaterial** [get, set]
- static Texture2D **EmptyTexture** [get]
- static Vector2Int **CellDimension** [get]

6.10.1 Detailed Description

Class which handles the creation of the Texture Arrays and Meta Data.

6.10.2 Member Data Documentation

6.10.2.1 BaseBlock

[Block](#) Structure2D.CellData.BaseBlock

This blocks is a solid unbreakable block

6.10.2.2 CustomBackgrounds

[Background](#) [] Structure2D.CellData.CustomBackgrounds

Here you can add all the [Background](#) CustomBackgrounds

6.10.2.3 CustomBlocks

```
Block [ ] Structure2D.CellData.CustomBlocks
```

Here you can add your own blocks

6.10.2.4 DefaultBackground

```
Background Structure2D.CellData.DefaultBackground
```

This is the placeholder background with which the map gets filled initially

6.10.2.5 DefaultBlock

```
Block Structure2D.CellData.DefaultBlock
```

This is the block with which the map gets filled initially

6.11 Structure2D.CellExtensions Class Reference

Static Public Member Functions

- static [Cell](#) [GetNeighbor](#) (this [Cell](#) cell, [Direction](#) direction)

Fetches the Neighbor at the given direction.

6.11.1 Member Function Documentation

6.11.1.1 GetNeighbor()

```
static Cell Structure2D.CellExtensions.GetNeighbor (
    this Cell cell,
    Direction direction ) [inline], [static]
```

Fetches the Neighbor at the given direction.

Parameters

<i>direction</i>	Direction from which we should fetch the Neighbor from
------------------	--

Returns

Returns the neighbor at the given direction, if there is none it returns null

6.12 Structure2D.CellMap Class Reference

This is the storage class for all the Cells. It also provides convenient methods of fetching Cells.

Static Public Member Functions

- static [Cell GetCell](#) (int x, int y)
This function returns the cell at the given index. Before returning this functions checks if the given coordinate was in bounds of the [CellMap](#). If it wasn't it returns null
- static [Cell GetCell](#) ([Coordinate](#) coordinate)
Returns the cell at the given coordinate
- static [Cell GetCellUnsafe](#) (int x, int y)
Returns the [Cell](#) at the given index. The difference to [GetCell](#) is that this call doesn't check if the index is in bounds of the [CellMap](#).
- static [Chunk GetChunkAtOffset](#) (Vector2Int offset)
Fetches the chunk at the given offset. If there is no [CHunk](#) at the offset it returns null.
- static [Chunk WorldPointToChunk](#) (Vector2 worldPoint)
Returns the chunk at the given point in world space.
- static Vector2Int [WorldPointToChunkOffset](#) (Vector3 worldPoint)
- static [Chunk MousePositionToChunk](#) (Vector2 mousePosition)
Returns the chunk at the given mouse position
- static [Cell GetCellAtWorldPoint](#) (Vector2 worldPoint)
Returns the [Cell](#) at the given world point.
- static List< [Cell](#) > [GetCellsInBounds](#) (Vector2 screenPosition, int bounds)
- static [Cell ScreenPositionToCell](#) (Vector2 screenPosition)
Fetches the [Cell](#) at the given position in Screen Space.

Static Public Attributes

- static Action [MapUnloaded](#)
This gets called when a new Map Gets Generated.
- static Action [MapInitialized](#)
This gets called when the Map leaves the hidden state.

Properties

- static int [MapHeight](#) [get]
Current Map Height In Cells.
- static int [MapWidth](#) [get]
Current Map Width in Cells.
- static bool [IsMapHidden](#) [get]
This is used to pause Updating components which are dependent on the [CellMap](#). This is useful when you want to access Cells from a separate thread, so you don't have to worry about race conditions.

6.12.1 Detailed Description

This is the storage class for all the Cells. It also provides convenient methods of fetching Cells.

6.12.2 Member Function Documentation

6.12.2.1 GetCell() [1/2]

```
static Cell Structure2D.CellMap.GetCell (
    Coordinate coordinate ) [static]
```

Returns the cell at the given coordinate

Parameters

<i>coordinate</i>	Coordinate of the desired Cell.
-------------------	---------------------------------

Returns

6.12.2.2 GetCell() [2/2]

```
static Cell Structure2D.CellMap.GetCell (
    int x,
    int y ) [inline], [static]
```

This function returns the cell at the given index. Before returning this functions checks if the given coordinate was in bounds of the CellMap. If it wasn't it returns null

Parameters

<i>x</i>	X coordinate of the desired Cell
<i>y</i>	Y coordinate of the desired Cell

6.12.2.3 GetCellAtWorldPoint()

```
static Cell Structure2D.CellMap.GetCellAtWorldPoint (
    Vector2 worldPoint ) [inline], [static]
```

Returns the Cell at the given world point.

Parameters

<i>worldPoint</i>	Point in world space.
-------------------	-----------------------

6.12.2.4 GetCellUnsafe()

```
static Cell Structure2D.CellMap.GetCellUnsafe (
    int x,
    int y ) [inline], [static]
```

Returns the [Cell](#) at the given index. The difference to GetCell is that this call doesn't check if the index is in bounds of the [CellMap](#).

Returns

6.12.2.5 GetChunkAtOffset()

```
static Chunk Structure2D.CellMap.GetChunkAtOffset (
    Vector2Int offset ) [inline], [static]
```

Fetches the chunk at the given offset. If there is no CHunk at the offset it returns null.

Parameters

<i>offset</i>	Offset of the desired Chunk .
---------------	---

Returns

6.12.2.6 MousePositionToChunk()

```
static Chunk Structure2D.CellMap.MousePositionToChunk (
    Vector2 mousePosition ) [inline], [static]
```

Returns the chunk at the given mouse position

Parameters

<i>mousePosition</i>	
----------------------	--

Returns

6.12.2.7 ScreenPositionToCell()

```
static Cell Structure2D.CellMap.ScreenPositionToCell (
    Vector2 screenPosition ) [inline], [static]
```

Fetches the [Cell](#) at the given position in Screen Space.

Parameters

<i>screenPosition</i>	Position in Screen Space of the desired Cell .
-----------------------	--

6.12.2.8 WorldPointToChunk()

```
static Chunk Structure2D.CellMap.WorldPointToChunk (
    Vector2 worldPoint ) [inline], [static]
```

Returns the chunk at the given point in world space.

Parameters

<i>worldPoint</i>	
-------------------	--

Returns

6.12.3 Member Data Documentation

6.12.3.1 MapInitialized

```
Action Structure2D.CellMap.MapInitialized [static]
```

This gets called when the Map leaves the hidden state.

6.12.3.2 MapUnloaded

```
Action Structure2D.CellMap.MapUnloaded [static]
```

This gets called when a new Map Gets Generated.

6.12.4 Property Documentation

6.12.4.1 IsMapHidden

```
bool Structure2D.CellMap.IsMapHidden [static], [get]
```

This is used to pause Updating components which are dependent on the [CellMap](#). This is useful when you want to access Cells from a separate thread, so you don't have to worry about race conditions.

6.12.4.2 MapHeight

```
int Structure2D.CellMap.MapHeight [static], [get]
```

Current Map Height In Cells.

6.12.4.3 MapWidth

```
int Structure2D.CellMap.MapWidth [static], [get]
```

Current Map Width in Cells.

6.13 Structure2D.CellMetrics Class Reference

Class which holds constant data about the Cells and Chunks.

Static Public Attributes

- const float [CellSize](#) = 0.25f
The size that every [Cell](#) should be big.
- const int [ChunkSize](#) = 10
The amount of Cells that a chunk can store in each dimensions.

6.13.1 Detailed Description

Class which holds constant data about the Cells and Chunks.

6.13.2 Member Data Documentation

6.13.2.1 CellSize

```
const float Structure2D.CellMetrics.CellSize = 0.25f [static]
```

The size that every [Cell](#) should be big.

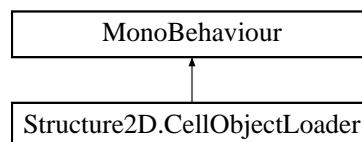
6.13.2.2 ChunkSize

```
const int Structure2D.CellMetrics.ChunkSize = 10 [static]
```

The amount of Cells that a chunk can store in each dimensions.

6.14 Structure2D.CellObjectLoader Class Reference

Inheritance diagram for Structure2D.CellObjectLoader:



Public Member Functions

- void **SetMappedPosition** (Vector2Int newPosition)

6.15 Structure2D.Chunk Class Reference

Chunks handle the drawing and collider generation of Cells.

Public Member Functions

- void [GenerateColliders](#) ()
Generates colliders for this [Chunk](#).

Properties

- bool [IsVisible](#) [get, set]
Is the [Chunk](#) currently Visible.
- Vector2Int [Offset](#) [get, set]
Offset of this [Chunk](#).
- Vector2Int [CellOffset](#) [get]
Offset of this [Chunk](#) in Cells.

6.15.1 Detailed Description

Chunks handle the drawing and collider generation of Cells.

6.15.2 Member Function Documentation

6.15.2.1 GenerateColliders()

```
void Structure2D.Chunk.GenerateColliders ( ) [inline]
```

Generates colliders for this [Chunk](#).

6.15.3 Property Documentation

6.15.3.1 CellOffset

```
Vector2Int Structure2D.Chunk.CellOffset [get]
```

Offset of this [Chunk](#) in Cells.

6.15.3.2 IsVisible

```
bool Structure2D.Chunk.IsVisible [get], [set]
```

Is the [Chunk](#) currently Visible.

6.15.3.3 Offset

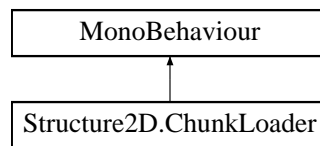
`Vector2Int Structure2D.Chunk.Offset [get], [set]`

Offset of this [Chunk](#).

6.16 Structure2D.ChunkLoader Class Reference

The [ChunkLoader](#) handles the Loading/Unloading of Chunks based on the current Viewer Position

Inheritance diagram for Structure2D.ChunkLoader:



Public Member Functions

- void **Update** ()

Public Attributes

- GameObject [Viewer](#)
Viewer on which position we base the viewport on

Static Public Attributes

- static Action< Vector2Int > [ChunkLoaded](#)
This gets called when a chunk gets loaded, with the chunk offset of the unloaded chunk.
- static Action< Vector2Int > [ChunkUnloaded](#)
This gets called when a chunk gets unloaded, with the chunk offset of the unloaded chunk.

Properties

- static [ChunkLoader Singleton](#) [get]
- static int [DesiredViewPortSizeX](#) [get]
Width of the [Viewport](#) in Chunks.
- static int [DesiredViewPortSizeY](#) [get]
Height of the [Viewport](#) in Chunks.

6.16.1 Detailed Description

The [ChunkLoader](#) handles the Loading/Unloading of Chunks based on the current Viewer Position

6.16.2 Member Data Documentation

6.16.2.1 ChunkLoaded

```
Action<Vector2Int> Structure2D.ChunkLoader.ChunkLoaded [static]
```

This gets called when a chunk gets loaded, with the chunk offset of the unloaded chunk.

6.16.2.2 ChunkUnloaded

```
Action<Vector2Int> Structure2D.ChunkLoader.ChunkUnloaded [static]
```

This gets called when a chunk gets unloaded, with the chunk offset of the unloaded chunk.

6.16.2.3 Viewer

```
GameObject Structure2D.ChunkLoader.Viewer
```

Viewer on which position we base the viewport on

6.16.3 Property Documentation

6.16.3.1 DesiredViewPortSizeX

```
int Structure2D.ChunkLoader.DesiredViewPortSizeX [static], [get]
```

Width of the [Viewport](#) in Chunks.

6.16.3.2 DesiredViewPortSizeY

```
int Structure2D.ChunkLoader.DesiredViewPortSizeY [static], [get]
```

Height of the [Viewport](#) in Chunks.

6.17 Structure2D.Coordinate Struct Reference

Public Member Functions

- **Coordinate** (int x, int y)
- int **DistanceTo** ([Coordinate](#) other)
Returns the distance from this coordinate to the given one
- int **GetIndex** ()
Returns the index of this coordinate as it's 1D representation
- override string **ToString** ()

Static Public Member Functions

- static Vector3 **ToWorldPoint** ([Coordinate](#) coordinate)
- static Vector3 **ToWorldPoint** ([Coordinate](#) coordinate, [CoordinateAnchor](#) anchor)
- static [Coordinate](#) **FromScreenPoint** (Vector2 screenPoint)
Creates a coordinate from the given ScreenPoint
- static [Coordinate](#) **FromWorldPoint** (Vector2 worldPoint)
- static int **ToIndex** (int x, int y)
This converts the given 2D index to a 1D index
- static [Coordinate](#) **operator+** ([Coordinate](#) a, [Coordinate](#) b)
- static [Coordinate](#) **operator-** ([Coordinate](#) a, [Coordinate](#) b)
- static bool **operator==** ([Coordinate](#) a, [Coordinate](#) b)
- static bool **operator!=** ([Coordinate](#) a, [Coordinate](#) b)
- static [Coordinate](#) **operator*** ([Coordinate](#) a, [Coordinate](#) b)
- static [Coordinate](#) **operator*** ([Coordinate](#) a, int b)
- static implicit **operator Coordinate** (Vector2Int vectorToTransform)

Public Attributes

- int **x**
- int **y**

Static Public Attributes

- static [Coordinate](#) **Up** => new [Coordinate](#)(0, 1)
- static [Coordinate](#) **Down** => new [Coordinate](#)(0, -1)
- static [Coordinate](#) **Left** => new [Coordinate](#)(-1, 0)
- static [Coordinate](#) **Right** => new [Coordinate](#)(1, 0)
- static [Coordinate](#) **One** => new [Coordinate](#)(1, 1)

6.17.1 Member Function Documentation

6.17.1.1 DistanceTo()

```
int Structure2D.Coordinate.DistanceTo (
    Coordinate other ) [inline]
```

Returns the distance from this coordinate to the given one

Parameters

<i>other</i>	
--------------	--

Returns

6.17.1.2 FromScreenPoint()

```
static Coordinate Structure2D.Coordinate.FromScreenPoint (
    Vector2 screenPoint ) [inline], [static]
```

Creates a coordinate from the given ScreenPoint

Parameters

<i>screenPoint</i>	
--------------------	--

Returns

6.17.1.3 GetIndex()

```
int Structure2D.Coordinate.GetIndex ( ) [inline]
```

Returns the index of this coordinate as it's 1D representation

Returns

6.17.1.4 ToIndex()

```
static int Structure2D.Coordinate.ToIndex (
    int x,
    int y ) [inline], [static]
```

This converts the given 2D index to a 1D index

Parameters

<i>x</i>	
<i>y</i>	

Returns

6.18 Structure2D.Base.Utility.DebugUtility Class Reference

Used to log additional debug information.

Public Member Functions

- delegate void **DebugTarget** (string stringToLog)

Static Public Member Functions

- static void [SetDebugTarget](#) (DebugTarget target)
Sets the target to which the Debug [Utility](#) should print.

Properties

- static bool [DoDebug](#) = true [get]
If this is set to true some classes will print additional information to the debug target.

6.18.1 Detailed Description

Used to log additional debug information.

6.18.2 Member Function Documentation

6.18.2.1 SetDebugTarget()

```
static void Structure2D.Base.Utility.DebugUtility.SetDebugTarget (  
    DebugTarget target ) [inline], [static]
```

Sets the target to which the Debug [Utility](#) should print.

Parameters

<i>target</i>	
---------------	--

6.18.3 Property Documentation

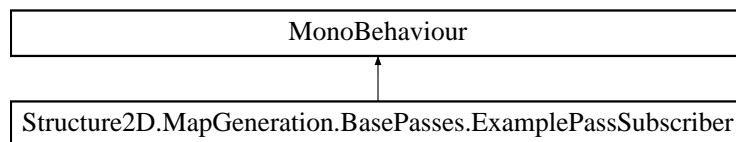
6.18.3.1 DoDebug

```
bool Structure2D.Base.Utility.DebugUtility.DoDebug = true [static], [get]
```

If this is set to true some classes will print additional information to the debug target.

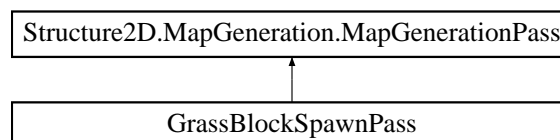
6.19 Structure2D.MapGeneration.BasePasses.ExamplePassSubscriber Class Reference

Inheritance diagram for Structure2D.MapGeneration.BasePasses.ExamplePassSubscriber:



6.20 GrassBlockSpawnPass Class Reference

Inheritance diagram for GrassBlockSpawnPass:



Public Member Functions

- override void [Apply](#) ([MapGenerator](#) mapGenerator)
Override this to add your Pass logic.

6.20.1 Member Function Documentation

6.20.1.1 Apply()

```
override void GrassBlockSpawnPass.Apply (
    MapGenerator mapGenerator ) [inline], [virtual]
```

Override this to add your Pass logic.

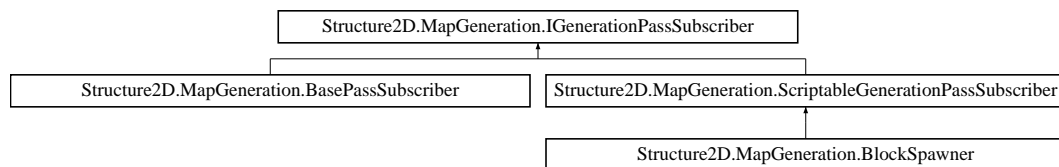
Parameters

<code>mapGenerator</code>	This is the generator that calls the pass.
---------------------------	--

Reimplemented from [Structure2D.MapGeneration.MapGenerationPass](#).

6.21 Structure2D.MapGeneration.IGenerationPassSubscriber Interface Reference

Inheritance diagram for Structure2D.MapGeneration.IGenerationPassSubscriber:



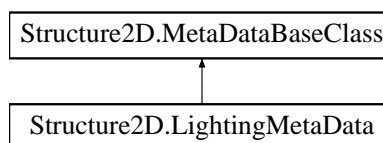
Public Member Functions

- [MapGenerationPass\[\]](#) **GetPasses** ()
- int **FetchPassOrder** ()

6.22 Structure2D.LightingMetaData Class Reference

[Lighting](#) Meta Data stores information about the lighting data of Blocks and Backgrounds.

Inheritance diagram for Structure2D.LightingMetaData:



Public Member Functions

- override void [RegisterForBlockInitialization](#) (int size)
This will be called before we iterate over the blocks.
- override void [EnumerateBlockInitialization](#) ([Block](#) block)
This is called for every [Block](#) that we iterate over.
- override void [RegisterForBackgroundInitialization](#) (int size)
This is called for every background we iterate over.
- override void [EnumerateBackgroundInitialization](#) ([Background](#) background)
This is called for every [Background](#) we iterate over.

Static Public Attributes

- static byte[] **BackgroundsLightBlockAmount**
- static byte[] **BlocksLightBlockAmount**
- static bool[] **DoesBackgroundTypeBlockSunlight**

6.22.1 Detailed Description

[Lighting](#) Meta Data stores information about the lighting data of Blocks and Backgrounds.

6.22.2 Member Function Documentation

6.22.2.1 EnumerateBackgroundInitialization()

```
override void Structure2D.LightingMetaData.EnumerateBackgroundInitialization (
    Background background ) [inline], [virtual]
```

This is called for every [Background](#) we iterate over.

Parameters

<i>background</i>	The current Background .
-------------------	--

Reimplemented from [Structure2D.MetaDataBaseClass](#).

6.22.2.2 EnumerateBlockInitialization()

```
override void Structure2D.LightingMetaData.EnumerateBlockInitialization (
    Block block ) [inline], [virtual]
```

This is called for every [Block](#) that we iterate over.

Parameters

<i>block</i>	The current Block .
--------------	-------------------------------------

Reimplemented from [Structure2D.MetaDataBaseClass](#).

6.22.2.3 RegisterForBackgroundInitialization()

```
override void Structure2D.LightingMetaData.RegisterForBackgroundInitialization (
    int size ) [inline], [virtual]
```

This is called for every background we iterate over.

Parameters

<i>size</i>	Amount of background that we iterate over.
-------------	--

Reimplemented from [Structure2D.MetadataBaseClass](#).

6.22.2.4 RegisterForBlockInitialization()

```
override void Structure2D.LightingMetaData.RegisterForBlockInitialization (
    int size ) [inline], [virtual]
```

This will be called before we iterate over the blocks.

Parameters

<i>size</i>	The amount of blocks we iterate over.
-------------	---------------------------------------

Reimplemented from [Structure2D.MetadataBaseClass](#).

6.23 Structure2D.Lighting.LightMap Class Reference

Used to store lighting data for all Cells of the current Map.

Static Public Member Functions

- static void **SetLightMap** (Color32[] threadLightValues)
- static NativeArray< Color32 > **GetLightMap** ()
- static void **Clear** ()

6.23.1 Detailed Description

Used to store lighting data for all Cells of the current Map.

6.24 Structure2D.Utility.ListPool< T > Class Template Reference

Generic Pool which you can use to reuse Lists.

Static Public Member Functions

- static List< T > [Get](#) ()
Returns a list of the pool. If there is none one will be created.
- static void [Add](#) (List< T > list)
Push a list back into the pool. The list will be cleared before being added.

6.24.1 Detailed Description

Generic Pool which you can use to reuse Lists.

Template Parameters

<i>T</i>	
----------	--

6.24.2 Member Function Documentation

6.24.2.1 Add()

```
static void Structure2D.Utility.ListPool< T >.Add (
    List< T > list ) [inline], [static]
```

Push a list back into the pool. The list will be cleared before being added.

6.24.2.2 Get()

```
static List<T> Structure2D.Utility.ListPool< T >.Get ( ) [inline], [static]
```

Returns a list of the pool. If there is none one will be created.

Returns

6.25 Structure2D.Utility.MapData Struct Reference

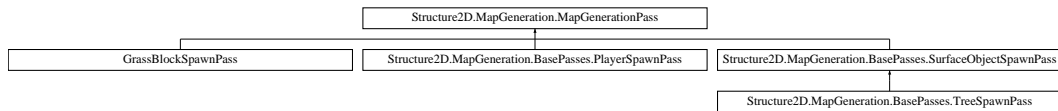
Public Attributes

- int **MapWidth**
- int **MapHeight**
- double **MapVersion**

6.26 Structure2D.MapGeneration.MapGenerationPass Class Reference

[Base](#) class for the passes of the Map Generator

Inheritance diagram for Structure2D.MapGeneration.MapGenerationPass:



Public Member Functions

- virtual void [Apply](#) ([MapGenerator](#) mapGenerator)
Override this to add your Pass logic.
- virtual void [PrepareGeneration](#) ()
This is called before the generation gets started on the separate thread, you can use this to prepare things that are only possible on the main thread.
- virtual int [GetWeight](#) ()
This weight is used to calculate the current Map Generation progress.

6.26.1 Detailed Description

[Base](#) class for the passes of the Map Generator

6.26.2 Member Function Documentation

6.26.2.1 Apply()

```
virtual void Structure2D.MapGeneration.MapGenerationPass.Apply (
    MapGenerator mapGenerator ) [inline], [virtual]
```

Override this to add your Pass logic.

Parameters

<i>mapGenerator</i>	This is the generator that calls the pass.
---------------------	--

Reimplemented in [Structure2D.MapGeneration.BasePasses.SurfaceObjectSpawnPass](#), [Structure2D.MapGeneration.BasePasses.PlayerSpawnPass](#) and [GrassBlockSpawnPass](#).

6.26.2.2 GetWeight()

```
virtual int Structure2D.MapGeneration.MapGenerationPass.GetWeight ( ) [inline], [virtual]
```

This weight is used to calculate the current Map Generation progress.

Reimplemented in [Structure2D.MapGeneration.BasePasses.PlayerSpawnPass](#).

6.26.2.3 PrepareGeneration()

```
virtual void Structure2D.MapGeneration.MapGenerationPass.PrepareGeneration ( ) [inline],  
[virtual]
```

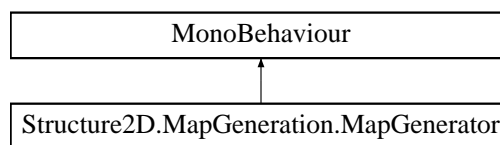
This is called before the generation gets started on the separate thread, you can use this to prepare things that are only possible on the main thread.

Reimplemented in [Structure2D.MapGeneration.BasePasses.SurfaceObjectSpawnPass](#), [Structure2D.MapGeneration.BasePasses.TreasureSpawnPass](#) and [Structure2D.MapGeneration.BasePasses.PlayerSpawnPass](#).

6.27 Structure2D.MapGeneration.MapGenerator Class Reference

Default Map Generator.

Inheritance diagram for Structure2D.MapGeneration.MapGenerator:



Classes

- class [ObjectSpawnQueueData](#)
Used to queue the spawning of objects while in a multi-threaded environment.

Public Member Functions

- void [GenerateMap](#) ()
Generates a map with the default settings.
- void [GenerateMap](#) ([MapGeneratorSettings](#) settings)
Generates a map with the given settings.
- [Cell](#) [GetRandomSolidCell](#) ()
Searches for a solid cell starting from a random starting cell
- bool [IsBlockInDesiredHeight](#) (int cellHeight, float minSpawnHeight, float maxSpawnHeight)
Returns whether the given block height is higher than the min spawn height and lower than the max spawn height
- void [AddPass](#) ([MapGenerationPass](#) pass, int priority)
- int? [GetSurfaceCellHeight](#) (int column)
- [ObjectSpawnQueueData](#) [SpawnGameObject](#) (GameObject objectToSpawn)
Use this to spawn GameObjects from passes, this add the given objects to a queue where it gets spawned as soon as the Map Generation finished.
- [ObjectSpawnQueueData](#) [SpawnGameObject](#) (GameObject objectToSpawn, [ObjectSpawnQueueData](#) parent)
- [ObjectSpawnQueueData](#) [SpawnGameObject](#) (GameObject objectToSpawn, Vector3 position, Quaternion rotation, [ObjectSpawnQueueData](#) parent)

Static Public Attributes

- static Action [StartedMapGeneration](#)
This is called when the Map Generator starts generating the Map.
- static Action [FinishedMapGeneration](#)
This is called when the Map Generation has finished.
- static Action< [MapGenerator](#) > [PreparePasses](#)
You can use this to add your passes without an [IPassGenerationSubscriber](#). Just call [AddPass](#) on the given [MapGenerator](#).

Properties

- [MapGeneratorSettings](#) **ActiveMapGenSettings** [get]
- int [BaseChunkCellHeight](#) [get]
Height in Cells which the [Base](#) pass uses.
- int [AirChunkCellHeight](#) [get]
Height in cells which the Air pass uses.
- int [GroundCellHeight](#) [get]
Height in Cells which the Ground pass uses.
- int [GroundCellStart](#) [get]
Height in Cells at which the Ground pass starts.
- int [TerrainCellHeight](#) [get]
Height in Cells which the Terrain pass uses.
- System.Random [MapGenRandom](#) [get]
If you need randoms inside a [MapGenerationPass](#) you can use this variable. Unity's' Random class doesn't work on any other thread than the main one.
- int [UsableDefaultBlocks](#) [get, set]
Amount of Blocks that are the default [Block](#). This is used by the Default [Block](#) Pass to check if there are enough blocks leftover to spawn.
- int **ActiveSeed** [get]
- int [MapWidth](#) [get]
Map Width in Cells.
- int [MapHeight](#) [get]
Map Height in Cells.
- static long [GenerationProgress](#) [get, set]
Generation Progress in a range of 0..100

6.27.1 Detailed Description

Default Map Generator.

6.27.2 Member Function Documentation

6.27.2.1 GenerateMap() [1/2]

```
void Structure2D.MapGeneration.MapGenerator.GenerateMap ( ) [inline]
```

Generates a map with the default settings.

6.27.2.2 GenerateMap() [2/2]

```
void Structure2D.MapGeneration.MapGenerator.GenerateMap (
    MapGeneratorSettings settings ) [inline]
```

Generates a map with the given settings.

Parameters

<i>settings</i>	Settings that the Map Generator should use.
-----------------	---

6.27.2.3 GetRandomSolidCell()

```
Cell Structure2D.MapGeneration.MapGenerator.GetRandomSolidCell ( ) [inline]
```

Searches for a solid cell starting from a random starting cell

6.27.2.4 IsBlockInDesiredHeight()

```
bool Structure2D.MapGeneration.MapGenerator.IsBlockInDesiredHeight (
    int cellHeight,
    float minSpawnHeight,
    float maxSpawnHeight ) [inline]
```

Returns whether the given block height is higher than the min spawn height and lower than the max spawn height

Parameters

<i>minSpawnHeight</i>	This is the lowest height in percent where the block will be counted as valid
<i>maxSpawnHeight</i>	This is the maximum height in percent where the block will be counted as valid

6.27.2.5 SpawnGameObject()

```
ObjectSpawnQueueData Structure2D.MapGeneration.MapGenerator.SpawnGameObject (
```

```
GameObject objectToSpawn ) [inline]
```

Use this to spawn GameObjects from passes, this add the given objects to a queue where it gets spawned as soon as the Map Generation finished.

Parameters

<i>objectToSpawn</i>	
----------------------	--

6.27.3 Member Data Documentation

6.27.3.1 FinishedMapGeneration

```
Action Structure2D.MapGeneration.MapGenerator.FinishedMapGeneration [static]
```

This is called when the Map Generation has finished.

6.27.3.2 PreparePasses

```
Action<MapGenerator> Structure2D.MapGeneration.MapGenerator.PreparePasses [static]
```

You can use this to add your passes without an IPassGenerationSubscriber. Just call AddPass on the given [MapGenerator](#).

6.27.3.3 StartedMapGeneration

```
Action Structure2D.MapGeneration.MapGenerator.StartedMapGeneration [static]
```

This is called when the Map Generator starts generating the Map.

6.27.4 Property Documentation

6.27.4.1 AirChunkCellHeight

```
int Structure2D.MapGeneration.MapGenerator.AirChunkCellHeight [get]
```

Height in cells which the Air pass uses.

6.27.4.2 BaseChunkCellHeight

```
int Structure2D.MapGeneration.MapGenerator.BaseChunkCellHeight [get]
```

Height in Cells which the [Base](#) pass uses.

6.27.4.3 GenerationProgress

```
long Structure2D.MapGeneration.MapGenerator.GenerationProgress [static], [get], [set]
```

Generation Progress in a range of 0..100

6.27.4.4 GroundCellHeight

```
int Structure2D.MapGeneration.MapGenerator.GroundCellHeight [get]
```

Height in Cells which the Ground pass uses.

6.27.4.5 GroundCellStart

```
int Structure2D.MapGeneration.MapGenerator.GroundCellStart [get]
```

Height in Cells at which the Ground pass starts.

6.27.4.6 MapGenRandom

```
System.Random Structure2D.MapGeneration.MapGenerator.MapGenRandom [get]
```

If you need randoms inside a [MapGenerationPass](#) you can use this variable. Unity's Random class doesn't work on any other thread than the main one.

6.27.4.7 MapHeight

```
int Structure2D.MapGeneration.MapGenerator.MapHeight [get]
```

Map Height in Cells.

6.27.4.8 MapWidth

```
int Structure2D.MapGeneration.MapGenerator.MapWidth [get]
```

Map Width in Cells.

6.27.4.9 TerrainCellHeight

```
int Structure2D.MapGeneration.MapGenerator.TerrainCellHeight [get]
```

Height in Cells which the Terrain pass uses.

6.27.4.10 UsableDefaultBlocks

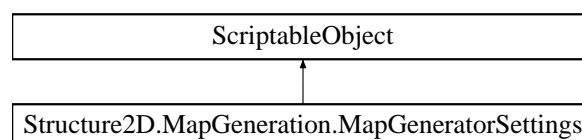
```
int Structure2D.MapGeneration.MapGenerator.UsableDefaultBlocks [get], [set]
```

Amount of Blocks that are the default [Block](#). This is used by the Default [Block](#) Pass to check if there are enough blocks leftover to spawn.

6.28 Structure2D.MapGeneration.MapGeneratorSettings Class Reference

This is the settings that you pass to the [MapGenerator](#) so he knows how to generate the Map.

Inheritance diagram for Structure2D.MapGeneration.MapGeneratorSettings:



Public Member Functions

- void **OnValidate** ()

Public Attributes

- float [NoiseMapXScale](#)
Used to scale the Noise Map for the terrain chunks
- float **NoiseMapYScale**
- int [MapWidth](#)
This is the
- int **MapHeight**
- int **AirChunks**
- int **BaseChunks**
- int **GroundChunks**
- int [Seed](#)
If this is -1 a random Seed will be used
- [ScriptableGenerationPassSubscriber](#)[] [PassSubscribers](#)
You can add custom passes that should be executed when generating the map in this field.

6.28.1 Detailed Description

This is the settings that you pass to the [MapGenerator](#) so he knows how to generate the Map.

6.28.2 Member Data Documentation

6.28.2.1 MapWidth

```
int Structure2D.MapGeneration.MapGeneratorSettings.MapWidth
```

This is the

6.28.2.2 NoiseMapXScale

```
float Structure2D.MapGeneration.MapGeneratorSettings.NoiseMapXScale
```

Used to scale the Noise Map for the terrain chunks

6.28.2.3 PassSubscribers

```
ScriptableGenerationPassSubscriber [ ] Structure2D.MapGeneration.MapGeneratorSettings.Pass↔  
Subscribers
```

You can add custom passes that should be executed when generating the map in this field.

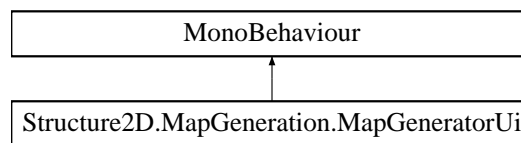
6.28.2.4 Seed

```
int Structure2D.MapGeneration.MapGeneratorSettings.Seed
```

If this is -1 a random Seed will be used

6.29 Structure2D.MapGeneration.MapGeneratorUi Class Reference

Inheritance diagram for Structure2D.MapGeneration.MapGeneratorUi:



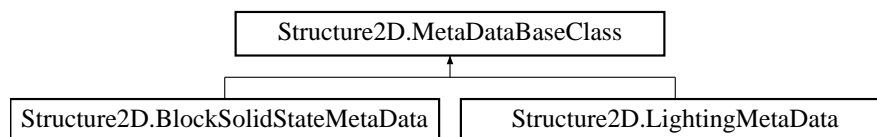
Public Member Functions

- void **LoadMap** ()

6.30 Structure2D.MetaDataBaseClass Class Reference

[Base](#) class for implementing your own Meta Data.

Inheritance diagram for Structure2D.MetaDataBaseClass:



Public Member Functions

- virtual void [RegisterForBlockInitialization](#) (int size)
This will be called before we iterate over the blocks.
- virtual void [EnumerateBlockInitialization](#) ([Block](#) block)
This is called for every [Block](#) that we iterate over.
- virtual void [RegisterForBackgroundInitialization](#) (int size)
This is called for every background we iterate over.
- virtual void [EnumerateBackgroundInitialization](#) ([Background](#) background)
This is called for every [Background](#) we iterate over.

6.30.1 Detailed Description

[Base](#) class for implementing your own Meta Data.

6.30.2 Member Function Documentation

6.30.2.1 EnumerateBackgroundInitialization()

```
virtual void Structure2D.MetadataBaseClass.EnumerateBackgroundInitialization (
    Background background ) [inline], [virtual]
```

This is called for every [Background](#) we iterate over.

Parameters

<i>background</i>	The current Background .
-------------------	--

Reimplemented in [Structure2D.LightingMetaData](#).

6.30.2.2 EnumerateBlockInitialization()

```
virtual void Structure2D.MetadataBaseClass.EnumerateBlockInitialization (
    Block block ) [inline], [virtual]
```

This is called for every [Block](#) that we iterate over.

Parameters

<i>block</i>	The current Block .
--------------	-------------------------------------

Reimplemented in [Structure2D.LightingMetaData](#), and [Structure2D.BlockSolidStateMetaData](#).

6.30.2.3 RegisterForBackgroundInitialization()

```
virtual void Structure2D.MetadataBaseClass.RegisterForBackgroundInitialization (
    int size ) [inline], [virtual]
```

This is called for every background we iterate over.

Parameters

<i>size</i>	Amount of background that we iterate over.
-------------	--

Reimplemented in [Structure2D.LightingMetaData](#).

6.30.2.4 RegisterForBlockInitialization()

```
virtual void Structure2D.MetadataBaseClass.RegisterForBlockInitialization (
    int size ) [inline], [virtual]
```

This will be called before we iterate over the blocks.

Parameters

<i>size</i>	The amount of blocks we iterate over.
-------------	---------------------------------------

Reimplemented in [Structure2D.LightingMetaData](#), and [Structure2D.BlockSolidStateMetaData](#).

6.31 Structure2D.MapGeneration.NoiseMap Class Reference

Static Public Member Functions

- static void [SetCustomNoiseMap](#) ([TerrainNoise](#) noise)
Use this to use your custom noise map instead of the default one.
- static IEnumerable< float > **GetTerrain** (int width, int height, int seed)
- static void **SetScale** (float settingsNoiseMapXScale, float settingsNoiseMapYScale)

6.31.1 Member Function Documentation

6.31.1.1 SetCustomNoiseMap()

```
static void Structure2D.MapGeneration.NoiseMap.SetCustomNoiseMap (
    TerrainNoise noise ) [inline], [static]
```

Use this to use your custom noise map instead of the default one.

Parameters

<i>noise</i>	
--------------	--

6.32 Structure2D.MapGeneration.BasePasses.SurfaceObjectSpawn↵ Pass.ObjectSpawnData Struct Reference

Public Attributes

- float **SpawnHeight**
- GameObject **ObjectToSpawn**

6.33 Structure2D.MapGeneration.MapGenerator.ObjectSpawnQueueData Class Reference

Used to queue the spawning of objects while in a multi-threaded environment.

Public Attributes

- GameObject [ObjectToSpawn](#)
Object to spawn.
- string **Name**
- [ObjectSpawnQueueData](#) **Parent**
Parent to spawn to.
- Vector3 [Position](#)
Desired Spawn Position.
- Quaternion [Rotation](#)
Desired Spawn Rotation.
- Type[] [Types](#)
These components get added when null is passed as the ObjectToSpawn.

Properties

- Transform [SpawnedObject](#) [get]
This will be resolved as soon as the object is spawned.

6.33.1 Detailed Description

Used to queue the spawning of objects while in a multi-threaded environment.

6.33.2 Member Data Documentation

6.33.2.1 ObjectToSpawn

GameObject Structure2D.MapGeneration.MapGenerator.ObjectSpawnQueueData.ObjectToSpawn

Object to Spawn.

6.33.2.2 Parent

[ObjectSpawnQueueData](#) Structure2D.MapGeneration.MapGenerator.ObjectSpawnQueueData.Parent

Parent to spawn to.

6.33.2.3 Position

`Vector3 Structure2D.MapGeneration.MapGenerator.ObjectSpawnQueueData.Position`

Desired Spawn Position.

6.33.2.4 Rotation

`Quaternion Structure2D.MapGeneration.MapGenerator.ObjectSpawnQueueData.Rotation`

Desired Spawn Rotation.

6.33.2.5 Types

`Type [] Structure2D.MapGeneration.MapGenerator.ObjectSpawnQueueData.Types`

These components get added when null is passed as the ObjectToSpawn.

6.33.3 Property Documentation

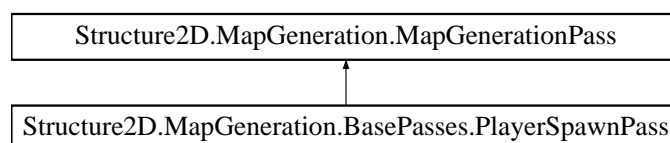
6.33.3.1 SpawnedObject

`Transform Structure2D.MapGeneration.MapGenerator.ObjectSpawnQueueData.SpawnedObject [get]`

This will be resolved as soon as the object is spawned.

6.34 Structure2D.MapGeneration.BasePasses.PlayerSpawnPass Class Reference

Inheritance diagram for Structure2D.MapGeneration.BasePasses.PlayerSpawnPass:



Public Member Functions

- override void [PrepareGeneration](#) ()
This is called before the generation gets started on the separate thread, you can use this to prepare things that are only possible on the main thread.
- override void [Apply](#) ([MapGenerator](#) mapGenerator)
Override this to add your Pass logic.
- override int [GetWeight](#) ()
This weight is used to calculate the current Map Generation progress.

6.34.1 Member Function Documentation

6.34.1.1 [Apply\(\)](#)

```
override void Structure2D.MapGeneration.BasePasses.PlayerSpawnPass.Apply (
    MapGenerator mapGenerator ) [inline], [virtual]
```

Override this to add your Pass logic.

Parameters

<i>mapGenerator</i>	This is the generator that calls the pass.
---------------------	--

Reimplemented from [Structure2D.MapGeneration.MapGenerationPass](#).

6.34.1.2 [GetWeight\(\)](#)

```
override int Structure2D.MapGeneration.BasePasses.PlayerSpawnPass.GetWeight ( ) [inline],
[virtual]
```

This weight is used to calculate the current Map Generation progress.

Reimplemented from [Structure2D.MapGeneration.MapGenerationPass](#).

6.34.1.3 [PrepareGeneration\(\)](#)

```
override void Structure2D.MapGeneration.BasePasses.PlayerSpawnPass.PrepareGeneration ( ) [inline],
[virtual]
```

This is called before the generation gets started on the separate thread, you can use this to prepare things that are only possible on the main thread.

Reimplemented from [Structure2D.MapGeneration.MapGenerationPass](#).

6.35 Structure2D.Utility.MapGeneration.PriorityQueue Class Reference

Queue used to store Map Generation Cells based on their priority.

Classes

- struct [QueueData](#)

Public Member Functions

- bool **IsCellQueueAble** ([Cell](#) cell)
- void **Enqueue** ([QueueData](#) data)
Enqueues the cell by it's priority.
- [Cell](#) **Dequeue** ([MapGenerator](#) generator)
Fetches the [Cell](#) with the highest priority.
- void **Clear** ()
Clears the queue.

Public Attributes

- int **Count** => count
Amount of Cells currently inside the queue.

6.35.1 Detailed Description

Queue used to store Map Generation Cells based on their priority.

6.35.2 Member Function Documentation

6.35.2.1 Clear()

```
void Structure2D.Utility.MapGeneration.PriorityQueue.Clear ( ) [inline]
```

Clears the queue.

6.35.2.2 Dequeue()

```
Cell Structure2D.Utility.MapGeneration.PriorityQueue.Dequeue (
    MapGenerator generator ) [inline]
```

Fetches the [Cell](#) with the highest priority.

Returns

6.35.2.3 Enqueue()

```
void Structure2D.Utility.MapGeneration.PriorityQueue.Enqueue (
    QueueData data ) [inline]
```

Enqueues the cell by it's priority.

6.35.3 Member Data Documentation

6.35.3.1 Count

```
int Structure2D.Utility.MapGeneration.PriorityQueue.Count => count
```

Amount of Cells currently inside the queue.

6.36 Structure2D.Utility.MapGeneration.PriorityQueue.QueueData Struct Reference

Public Attributes

- [Cell](#) Cell
- int Distance
- int SearchHeuristic

6.37 Structure2D.Utility.SaveManager Class Reference

[Utility](#) class used to save/load the [CellMap](#) from a Stream.

Static Public Member Functions

- static void [SaveMapToStream](#) (Stream stream)
Saves the currently active Map to the given stream
- static bool [LoadMapFromStream](#) (Stream stream, bool allowCrossVersionLoading=false)
Initializes the [CellMap](#) with the Map from the given stream
- static bool [CanLoadMap](#) (Stream stream, out [MapData](#) mapData, bool allowCrossVersionLoading=false)

Static Public Attributes

- const double **ProjectVersion** = 1.0
- static Action< BinaryWriter, [Coordinate](#) > [OnSavedCell](#)
This will be called for every [Cell](#) that gets serialized You can use the given writer to write your own data to the stream.
- static Action< BinaryReader, [Coordinate](#) > [OnLoadedCell](#)
If you use [OnSavedCell](#) to save your own custom data use this to read it back when the cells get loaded.

6.37.1 Detailed Description

[Utility](#) class used to save/load the [CellMap](#) from a Stream.

6.37.2 Member Function Documentation

6.37.2.1 LoadMapFromStream()

```
static bool Structure2D.Utility.SaveManager.LoadMapFromStream (
    Stream stream,
    bool allowCrossVersionLoading = false ) [inline], [static]
```

Initializes the [CellMap](#) with the Map from the given stream

Parameters

<i>stream</i>	Stream from which we should load the Map
<i>allowCrossVersionLoading</i>	Specifies if a Map with a different ProjectVersion can be loaded

Returns

6.37.2.2 SaveMapToStream()

```
static void Structure2D.Utility.SaveManager.SaveMapToStream (
    Stream stream ) [inline], [static]
```

Saves the currently active Map to the given stream

Parameters

<i>stream</i>	Stream to which we write the Map
---------------	----------------------------------

6.37.3 Member Data Documentation

6.37.3.1 OnLoadedCell

`Action<BinaryReader, Coordinate> Structure2D.Utility.SaveManager.OnLoadedCell [static]`

If you use OnSavedCell to save your own custom data use this to read it back when the cells get loaded.

6.37.3.2 OnSavedCell

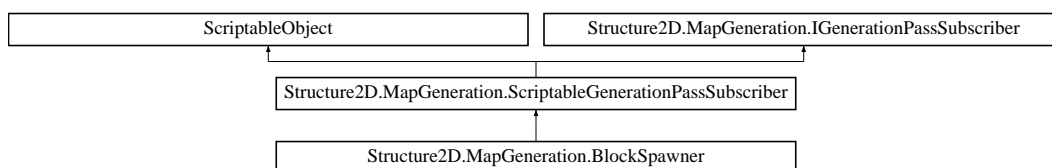
`Action<BinaryWriter, Coordinate> Structure2D.Utility.SaveManager.OnSavedCell [static]`

This will be called for every [Cell](#) that gets serialized You can use the given writer to write your own data to the stream.

6.38 Structure2D.MapGeneration.ScriptableGenerationPassSubscriber Class Reference

Subscriber which you can add to the Map Generator to run your own MapGenPasses.

Inheritance diagram for Structure2D.MapGeneration.ScriptableGenerationPassSubscriber:



Public Member Functions

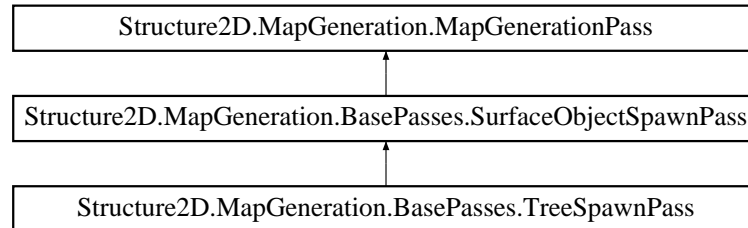
- virtual [MapGenerationPass](#)[] **GetPasses** ()
- virtual int **FetchPassOrder** ()
- int **PastProgressionWeight** ()

6.38.1 Detailed Description

Subscriber which you can add to the Map Generator to run your own MapGenPasses.

6.39 Structure2D.MapGeneration.BasePasses.SurfaceObjectSpawnPass Class Reference

Inheritance diagram for Structure2D.MapGeneration.BasePasses.SurfaceObjectSpawnPass:



Classes

- struct [ObjectSpawnData](#)

Public Member Functions

- override void [PrepareGeneration](#) ()
This is called before the generation gets started on the separate thread, you can use this to prepare things that are only possible on the main thread.
- override void [Apply](#) ([MapGenerator](#) mapGenerator)
Override this to add your Pass logic.

Protected Member Functions

- virtual int[] [GetRandomSpawnPoints](#) ([MapGenerator](#) mapGenerator)
This returns an array of random usable spawn points, which each have a random distance in range of min and max spawn distance to one another.
- virtual void [SpawnObject](#) ([Coordinate](#) spawnCoordinate, [MapGenerator](#) mapGenerator)
This is called for every object that the pass wants to spawn/. You can override this to add your own spawn logic.

Static Protected Member Functions

- static [MapGenerator.ObjectSpawnQueueData](#) [GetLayerParent](#) (int layer, [MapGenerator](#) mapGenerator)

Protected Attributes

- int **AmountOfObjectsToSpawn**
- int **MinSpawnDistance**
- int **MaxSpawnDistance**
- int **SpawnLayer**
- [ObjectSpawnData](#)[] **_spawnAbleObjects**

6.39.1 Member Function Documentation

6.39.1.1 Apply()

```
override void Structure2D.MapGeneration.BasePasses.SurfaceObjectSpawnPass.Apply (  
    MapGenerator mapGenerator ) [inline], [virtual]
```

Override this to add your Pass logic.

Parameters

<i>mapGenerator</i>	This is the generator that calls the pass.
---------------------	--

Reimplemented from [Structure2D.MapGeneration.MapGenerationPass](#).

6.39.1.2 GetRandomSpawnPoints()

```
virtual int [ ] Structure2D.MapGeneration.BasePasses.SurfaceObjectSpawnPass.GetRandomSpawn↵
Points (
    MapGenerator mapGenerator ) [inline], [protected], [virtual]
```

This returns an array of random usable spawn points, which each have a random distance in range of min and max spawn distance to one another.

6.39.1.3 PrepareGeneration()

```
override void Structure2D.MapGeneration.BasePasses.SurfaceObjectSpawnPass.PrepareGeneration (
) [inline], [virtual]
```

This is called before the generation gets started on the separate thread, you can use this to prepare things that are only possible on the main thread.

Reimplemented from [Structure2D.MapGeneration.MapGenerationPass](#).

Reimplemented in [Structure2D.MapGeneration.BasePasses.TreeSpawnPass](#).

6.39.1.4 SpawnObject()

```
virtual void Structure2D.MapGeneration.BasePasses.SurfaceObjectSpawnPass.SpawnObject (
    Coordinate spawnCoordinate,
    MapGenerator mapGenerator ) [inline], [protected], [virtual]
```

This is called for every object that the pass wants to spawn/. You can override this to add your own spawn logic.

Reimplemented in [Structure2D.MapGeneration.BasePasses.TreeSpawnPass](#).

6.40 Structure2D.MapGeneration.TerrainNoise Class Reference

This is the Noise base class To add your custom Noise Map all you have to do is override the GetTerrain function and call SetCustomNoiseMap on the [NoiseMap](#) class

Public Member Functions

- virtual IEnumerable< float > [GetTerrain](#) (int width, int height, int seed)
- void **SetScale** (float settingsNoiseMapXScale, float settingsNoiseMapYScale)

Protected Member Functions

- float **GetXScale** (int width)
- float **GetYScale** (int height)

6.40.1 Detailed Description

This is the Noise base class To add your custom Noise Map all you have to do is override the `GetTerrain` function and call `SetCustomNoiseMap` on the [NoiseMap](#) class

6.40.2 Member Function Documentation

6.40.2.1 GetTerrain()

```
virtual IEnumerable<float> Structure2D.MapGeneration.TerrainNoise.GetTerrain (
    int width,
    int height,
    int seed ) [inline], [virtual]
```

Parameters

<i>width</i>	
<i>height</i>	
<i>seed</i>	

Returns

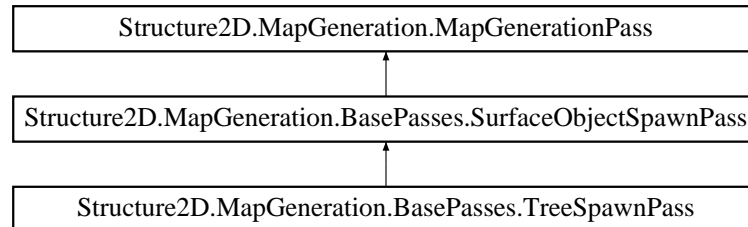
6.41 Structure2D.TerrainShaderExtensions Class Reference

Static Public Member Functions

- static void **SetBlock** (ref this Color32 color, byte block)
- static byte **GetBlock** (this Color32 color)
- static void **SetBackground** (ref this Color32 color, byte background)
- static byte **GetBackground** (this Color32 color)

6.42 Structure2D.MapGeneration.BasePasses.TreeSpawnPass Class Reference

Inheritance diagram for Structure2D.MapGeneration.BasePasses.TreeSpawnPass:



Public Member Functions

- override void [PrepareGeneration](#) ()

This is called before the generation gets started on the separate thread, you can use this to prepare things that are only possible on the main thread.

Public Attributes

- int [MaxTreeParts](#)

Max amount of parts a tree can have, without root and top.

- int [MinTreeParts](#)

Min amount of parts a tree can have, without root and top.

Static Public Attributes

- const int **TreeSpawnPassPriority** = 15

Protected Member Functions

- override void [SpawnObject](#) ([Coordinate](#) spawnCoordinate, [MapGenerator](#) mapGenerator)

This is called for every object that the pass wants to spawn/. You can override this to add your own spawn logic.

Additional Inherited Members

6.42.1 Member Function Documentation

6.42.1.1 PrepareGeneration()

```
override void Structure2D.MapGeneration.BasePasses.TreeSpawnPass.PrepareGeneration ( ) [inline],
[virtual]
```

This is called before the generation gets started on the separate thread, you can use this to prepare things that are only possible on the main thread.

Reimplemented from [Structure2D.MapGeneration.BasePasses.SurfaceObjectSpawnPass](#).

6.42.1.2 SpawnObject()

```
override void Structure2D.MapGeneration.BasePasses.TreeSpawnPass.SpawnObject (
    Coordinate spawnCoordinate,
    MapGenerator mapGenerator ) [inline], [protected], [virtual]
```

This is called for every object that the pass wants to spawn/. You can override this to add your own spawn logic.

Reimplemented from [Structure2D.MapGeneration.BasePasses.SurfaceObjectSpawnPass](#).

6.42.2 Member Data Documentation

6.42.2.1 MaxTreeParts

```
int Structure2D.MapGeneration.BasePasses.TreeSpawnPass.MaxTreeParts
```

Max amount of parts a tree can have, without root and top.

6.42.2.2 MinTreeParts

```
int Structure2D.MapGeneration.BasePasses.TreeSpawnPass.MinTreeParts
```

Min amount of parts a tree can have, without root and top.

6.43 Structure2D.Viewport Struct Reference

Public Member Functions

- bool [ContainsCoordinate](#) ([Coordinate](#) coordinate)
Checks if the given coordinate is in the [Viewport](#).

Public Attributes

- [Coordinate BottomLeft](#)
Bottom left coordinate of the [Viewport](#).
- int [Width](#)
Width of the [Viewport](#).
- int [Height](#)
Height of the [Viewport](#).

Properties

- static [Viewport CurrentViewport](#) [get, set]
The current [Viewport](#). This is set by the [ChunkLoader](#).

6.43.1 Member Function Documentation

6.43.1.1 ContainsCoordinate()

```
bool Structure2D.Viewport.ContainsCoordinate (
    Coordinate coordinate ) [inline]
```

Checks if the given coordinate is in the [Viewport](#).

Parameters

<i>coordinate</i>	Coordinate to check.
-------------------	--------------------------------------

Returns

Returns whether the coordinate is in the viewport.

6.43.2 Member Data Documentation

6.43.2.1 BottomLeft

```
Coordinate Structure2D.Viewport.BottomLeft
```

Bottom left coordinate of the [Viewport](#).

6.43.2.2 Height

```
int Structure2D.Viewport.Height
```

Height of the [Viewport](#).

6.43.2.3 Width

```
int Structure2D.Viewport.Width
```

Width of the [Viewport](#).

6.43.3 Property Documentation

6.43.3.1 CurrentViewport

```
Viewport Structure2D.Viewport.CurrentViewport [static], [get], [set]
```

The current [Viewport](#). This is set by the [ChunkLoader](#).

Index

- Add
 - Structure2D.Utility.ListPool< T >, [43](#)
- AddTemporaryLight
 - Structure2D.Lighting.BlockLighting, [19](#)
- AirChunkCellHeight
 - Structure2D.MapGeneration.MapGenerator, [48](#)
- Apply
 - GrassBlockSpawnPass, [39](#)
 - Structure2D.MapGeneration.BasePasses.PlayerSpawnPass,
[57](#)
 - Structure2D.MapGeneration.BasePasses.SurfaceObjectSpawnPass,
[64](#)
 - Structure2D.MapGeneration.MapGenerationPass,
[44](#)
- Background
 - Structure2D.Cell, [24](#)
- BaseBlock
 - Structure2D.CellData, [25](#)
- BaseChunkCellHeight
 - Structure2D.MapGeneration.MapGenerator, [48](#)
- Block
 - Structure2D.Cell, [24](#)
- BlocksSunLight
 - Structure2D.Background, [16](#)
- BottomLeft
 - Structure2D.Viewport, [69](#)
- CellOffset
 - Structure2D.Chunk, [33](#)
- CellSize
 - Structure2D.CellMetrics, [32](#)
- Chunk
 - Structure2D.Cell, [23](#)
- ChunkLoaded
 - Structure2D.ChunkLoader, [35](#)
- ChunkSize
 - Structure2D.CellMetrics, [32](#)
- ChunkUnloaded
 - Structure2D.ChunkLoader, [35](#)
- Clear
 - Structure2D.Utility.MapGeneration.PriorityQueue,
[58](#)
- ContainsCoordinate
 - Structure2D.Viewport, [69](#)
- Coordinate
 - Structure2D.Cell, [23](#)
- CoordinateAnchor
 - Structure2D, [12](#)
- Count
 - Structure2D.Utility.MapGeneration.PriorityQueue,
[59](#)
- CurrentViewport
 - Structure2D.Viewport, [70](#)
- CustomBackgrounds
 - Structure2D.CellData, [25](#)
- CustomBlocks
 - Structure2D.CellData, [25](#)
- DefaultBackground
 - Structure2D.CellData, [26](#)
- DefaultBlock
 - Structure2D.CellData, [26](#)
- Dequeue
 - Structure2D.Utility.MapGeneration.PriorityQueue,
[58](#)
- DesiredViewPortSizeX
 - Structure2D.ChunkLoader, [35](#)
- DesiredViewPortSizeY
 - Structure2D.ChunkLoader, [35](#)
- Direction
 - Structure2D.Utility, [14](#)
- DistanceTo
 - Structure2D.Coordinate, [36](#)
- DoDebug
 - Structure2D.Base.Utility.DebugUtility, [39](#)
- Enqueue
 - Structure2D.Utility.MapGeneration.PriorityQueue,
[59](#)
- EnumerateBackgroundInitialization
 - Structure2D.LightingMetaData, [41](#)
 - Structure2D.MetaDataBaseClass, [53](#)
- EnumerateBlockInitialization
 - Structure2D.BlockSolidStateMetaData, [20](#)
 - Structure2D.LightingMetaData, [41](#)
 - Structure2D.MetaDataBaseClass, [53](#)
- FinishedMapGeneration
 - Structure2D.MapGeneration.MapGenerator, [48](#)
- FromScreenPoint
 - Structure2D.Coordinate, [37](#)
- GenerateColliders
 - Structure2D.Chunk, [33](#)
- GenerateMap
 - Structure2D.MapGeneration.MapGenerator, [46](#), [47](#)
- GenerationProgress
 - Structure2D.MapGeneration.MapGenerator, [49](#)
- Get

- Structure2D.Utility.ListPool< T >, 43
- GetCell
 - Structure2D.CellMap, 28
- GetCellAtWorldPoint
 - Structure2D.CellMap, 28
- GetCellUnsafe
 - Structure2D.CellMap, 29
- GetChunkAtOffset
 - Structure2D.CellMap, 29
- GetIndex
 - Structure2D.Coordinate, 37
- GetNeighbor
 - Structure2D.CellExtensions, 26
- GetRandomSolidCell
 - Structure2D.MapGeneration.MapGenerator, 47
- GetRandomSpawnPoints
 - Structure2D.MapGeneration.BasePasses.SurfaceObjectSpawnPass, 65
- GetTerrain
 - Structure2D.MapGeneration.TerrainNoise, 66
- GetWeight
 - Structure2D.MapGeneration.BasePasses.PlayerSpawnPass, 57
 - Structure2D.MapGeneration.MapGenerationPass, 44
- GrassBlockSpawnPass, 39
 - Apply, 39
- GroundCellHeight
 - Structure2D.MapGeneration.MapGenerator, 49
- GroundCellStart
 - Structure2D.MapGeneration.MapGenerator, 49
- Height
 - Structure2D.Viewport, 69
- ID
 - Structure2D.Background, 16
 - Structure2D.Block, 18
- IsBlockInDesiredHeight
 - Structure2D.MapGeneration.MapGenerator, 47
- IsCellUsable
 - Structure2D.MapGeneration.BlockSpawnConditionChecker, 21
- IsMapHidden
 - Structure2D.CellMap, 31
- IsSolid
 - Structure2D.Block, 17
- IsVisible
 - Structure2D.Chunk, 33
- LightBlockAmount
 - Structure2D.Background, 16
 - Structure2D.Block, 18
- LoadMapFromStream
 - Structure2D.Utility.SaveManager, 60
- MapGenRandom
 - Structure2D.MapGeneration.MapGenerator, 49
- MapHeight
 - Structure2D.CellMap, 31
 - Structure2D.MapGeneration.MapGenerator, 49
- MapInitialized
 - Structure2D.CellMap, 30
- MapUnloaded
 - Structure2D.CellMap, 30
- MapWidth
 - Structure2D.CellMap, 31
 - Structure2D.MapGeneration.MapGenerator, 49
 - Structure2D.MapGeneration.MapGeneratorSettings, 51
- MaxLightDistance
 - Structure2D.Lighting.BlockLighting, 19
- MaxTreeParts
 - Structure2D.MapGeneration.BasePasses.TreeSpawnPass, 68
 - Structure2D.MapGeneration.BasePasses.TreeSpawnPass, 68
- MousePositionToChunk
 - Structure2D.CellMap, 29
- NoiseMapXScale
 - Structure2D.MapGeneration.MapGeneratorSettings, 51
- ObjectToSpawn
 - Structure2D.MapGeneration.MapGenerator.ObjectSpawnQueueData, 55
- Offset
 - Structure2D.Chunk, 33
- OnLoadedCell
 - Structure2D.Utility.SaveManager, 62
- OnSavedCell
 - Structure2D.Utility.SaveManager, 62
- Parent
 - Structure2D.MapGeneration.MapGenerator.ObjectSpawnQueueData, 55
- PassSubscribers
 - Structure2D.MapGeneration.MapGeneratorSettings, 51
- Position
 - Structure2D.MapGeneration.MapGenerator.ObjectSpawnQueueData, 55
- PrepareGeneration
 - Structure2D.MapGeneration.BasePasses.PlayerSpawnPass, 57
 - Structure2D.MapGeneration.BasePasses.SurfaceObjectSpawnPass, 65
 - Structure2D.MapGeneration.BasePasses.TreeSpawnPass, 67
 - Structure2D.MapGeneration.MapGenerationPass, 45
- PreparePasses
 - Structure2D.MapGeneration.MapGenerator, 48
- RegisterForBackgroundInitialization
 - Structure2D.Lighting.MetaData, 41

- Structure2D.MetadataBaseClass, 53
- RegisterForBlockInitialization
 - Structure2D.BlockSolidStateMetaData, 20
 - Structure2D.LightingMetaData, 42
 - Structure2D.MetadataBaseClass, 53
- Rotation
 - Structure2D.MapGeneration.MapGenerator.ObjectSpawnQueueData, 56
- SaveMapToStream
 - Structure2D.Utility.SaveManager, 60
- ScreenPositionToCell
 - Structure2D.CellMap, 30
- Seed
 - Structure2D.MapGeneration.MapGeneratorSettings, 51
- SetCustomNoiseMap
 - Structure2D.MapGeneration.NoiseMap, 54
- SetDebugTarget
 - Structure2D.Base.Utility.DebugUtility, 38
- SpawnedObject
 - Structure2D.MapGeneration.MapGenerator.ObjectSpawnQueueData, 56
- SpawnGameObject
 - Structure2D.MapGeneration.MapGenerator, 47
- SpawnObject
 - Structure2D.MapGeneration.BasePasses.SurfaceObjectSpawnPass, 65
 - Structure2D.MapGeneration.BasePasses.TreeSpawnPass, 68
- StartedMapGeneration
 - Structure2D.MapGeneration.MapGenerator, 48
- Structure2D, 11
 - CoordinateAnchor, 12
- Structure2D.Background, 15
 - BlocksSunLight, 16
 - ID, 16
 - LightBlockAmount, 16
 - Texture, 16
- Structure2D.Base, 12
- Structure2D.Base.Utility, 12
- Structure2D.Base.Utility.DebugUtility, 38
 - DoDebug, 39
 - SetDebugTarget, 38
- Structure2D.Block, 17
 - ID, 18
 - IsSolid, 17
 - LightBlockAmount, 18
 - Texture, 18
- Structure2D.BlockSolidStateMetaData, 20
 - EnumerateBlockInitialization, 20
 - RegisterForBlockInitialization, 20
- Structure2D.Cell, 23
 - Background, 24
 - Block, 24
 - Chunk, 23
 - Coordinate, 23
- Structure2D.CellData, 24
 - BaseBlock, 25
 - CustomBackgrounds, 25
 - CustomBlocks, 25
 - DefaultBackground, 26
 - DefaultBlock, 26
- Structure2D.CellExtensions, 26
 - GetNeighbor, 26
 - GetCell, 28
 - GetCellAtWorldPoint, 28
 - GetCellUnsafe, 29
 - GetChunkAtOffset, 29
 - IsMapHidden, 31
 - MapHeight, 31
 - MapInitialized, 30
 - MapUnloaded, 30
 - MapWidth, 31
 - MousePositionToChunk, 29
 - ScreenPositionToCell, 30
 - WorldPointToChunk, 30
- Structure2D.CellMetrics, 31
 - CellSize, 32
 - ChunkData, 32
 - ChunkSize, 32
- Structure2D.CellObjectLoader, 32
- Structure2D.Chunk, 32
 - CellOffset, 33
 - GenerateColliders, 33
 - IsVisible, 33
 - Offset, 33
- Structure2D.ChunkLoader, 34
 - ChunkLoaded, 35
 - ChunkUnloaded, 35
 - DesiredViewPortSizeX, 35
 - DesiredViewPortSizeY, 35
 - Viewer, 35
- Structure2D.Coordinate, 36
 - DistanceTo, 36
 - FromScreenPoint, 37
 - GetIndex, 37
 - ToIndex, 37
- Structure2D.Lighting, 12
- Structure2D.Lighting.BlockLighting, 18
 - AddTemporaryLight, 19
 - MaxLightDistance, 19
- Structure2D.Lighting.LightMap, 42
- Structure2D.LightingMetaData, 40
 - EnumerateBackgroundInitialization, 41
 - EnumerateBlockInitialization, 41
 - RegisterForBackgroundInitialization, 41
 - RegisterForBlockInitialization, 42
- Structure2D.MapGeneration, 13
- Structure2D.MapGeneration.BasePasses, 13
- Structure2D.MapGeneration.BasePasses.ExamplePassSubscriber, 39
- Structure2D.MapGeneration.BasePasses.PlayerSpawnPass, 56
 - Apply, 57
 - GetWeight, 57
 - PrepareGeneration, 57

- Structure2D.MapGeneration.BasePasses.SurfaceObjectSpawnPass,
 - 63
 - Apply, 64
 - GetRandomSpawnPoints, 65
 - PrepareGeneration, 65
 - SpawnObject, 65
- Structure2D.MapGeneration.BasePasses.SurfaceObjectSpawnPass.ObjectSpawnData,
 - 54
- Structure2D.MapGeneration.BasePasses.TreeSpawnPass,
 - 67
 - MaxTreeParts, 68
 - MinTreeParts, 68
 - PrepareGeneration, 67
 - SpawnObject, 68
- Structure2D.MapGeneration.BasePassSubscriber, 16
- Structure2D.MapGeneration.BlockSpawnConditionChecker,
 - 21
 - IsCellUsable, 21
- Structure2D.MapGeneration.BlockSpawnData, 22
- Structure2D.MapGeneration.BlockSpawner, 22
- Structure2D.MapGeneration.IGenerationPassSubscriber,
 - 40
- Structure2D.MapGeneration.MapGenerationPass, 44
 - Apply, 44
 - GetWeight, 44
 - PrepareGeneration, 45
- Structure2D.MapGeneration.MapGenerator, 45
 - AirChunkCellHeight, 48
 - BaseChunkCellHeight, 48
 - FinishedMapGeneration, 48
 - GenerateMap, 46, 47
 - GenerationProgress, 49
 - GetRandomSolidCell, 47
 - GroundCellHeight, 49
 - GroundCellStart, 49
 - IsBlockInDesiredHeight, 47
 - MapGenRandom, 49
 - MapHeight, 49
 - MapWidth, 49
 - PreparePasses, 48
 - SpawnGameObject, 47
 - StartedMapGeneration, 48
 - TerrainCellHeight, 50
 - UsableDefaultBlocks, 50
- Structure2D.MapGeneration.MapGenerator.ObjectSpawnQueueData,
 - 55
 - ObjectToSpawn, 55
 - Parent, 55
 - Position, 55
 - Rotation, 56
 - SpawnedObject, 56
 - Types, 56
- Structure2D.MapGeneration.MapGeneratorSettings, 50
 - MapWidth, 51
 - NoiseMapXScale, 51
 - PassSubscribers, 51
 - Seed, 51
- Structure2D.MapGeneration.MapGeneratorUi, 52
- Structure2D.MapGeneration.NoiseMap, 54
 - SetCustomNoiseMap, 54
- Structure2D.MapGeneration.ScriptableGenerationPassSubscriber,
 - 62
- Structure2D.MapGeneration.TerrainNoise, 65
 - GetTerrain, 66
- Structure2D.MapGeneration.TerrainNoiseBaseClass, 52
 - EnumerateBackgroundInitialization, 53
 - EnumerateBlockInitialization, 53
 - RegisterForBackgroundInitialization, 53
 - RegisterForBlockInitialization, 53
- Structure2D.TerrainShaderExtensions, 66
- Structure2D.Utility, 14
 - Direction, 14
- Structure2D.Utility.ListPool< T >, 42
 - Add, 43
 - Get, 43
- Structure2D.Utility.MapData, 43
- Structure2D.Utility.MapGeneration, 14
- Structure2D.Utility.MapGeneration.PriorityQueue, 58
 - Clear, 58
 - Count, 59
 - Dequeue, 58
 - Enqueue, 59
- Structure2D.Utility.MapGeneration.PriorityQueue.QueueData,
 - 59
- Structure2D.Utility.SaveManager, 59
 - LoadMapFromStream, 60
 - OnLoadedCell, 62
 - OnSavedCell, 62
 - SaveMapToStream, 60
- Structure2D.Viewport, 68
 - BottomLeft, 69
 - ContainsCoordinate, 69
 - CurrentViewport, 70
 - Height, 69
 - Width, 70
- TerrainCellHeight
 - Structure2D.MapGeneration.MapGenerator, 50
- Texture
 - Structure2D.Background, 16
 - Structure2D.Block, 18
- ToIndex
 - Structure2D.Coordinate, 37
- Types
 - Structure2D.MapGeneration.MapGenerator.ObjectSpawnQueueData,
 - 56
- UsableDefaultBlocks
 - Structure2D.MapGeneration.MapGenerator, 50
- Viewer
 - Structure2D.ChunkLoader, 35
- Width
 - Structure2D.Viewport, 70
- WorldPointToChunk
 - Structure2D.CellMap, 30