# Terrain Engine 2D

## A 2D Block Engine for Unity

Out now on the Unity Asset Store

BUY NOW!



**Terrain Engine 2D**
User Manual - V1.20



# Terrain

In depth information on how terrain works in the engine.

**Table of Contents**



## General

The terrain at its core is just a bunch of data representing the location of blocks and how they will be rendered. The terrain data is split up into layers of different block types, and this data is procedurally generated by a Terrain Generator script. Rendering of the terrain is done in Chunks so that only the visible portion of the world is rendered into scene.

## Terrain Data

Data is stored in 2-dimensional byte arrays, where each Block Layer has it's own set of data arrays. There are four main types of data stored in each Block Layer: **BlockType**, **RenderBlock**, **Bitmask**, and **Variation** data. The index of each data element in the 2d byte array represents a grid coordinate (x, y), where (0, 0) is the bottom left most point of the terrain.

When the Terrain Generator script runs it fills the BlockType and RenderBlock data arrays, the Bitmask and Variation data is generated after.

While in game, this data is constantly changing as the terrain is modified. When blocks are added or removed the location of blocktypes and renderblocks must be updated, bitmasking values must be regenerated for surrounding blocks, and variations must be generated or cleared.

### BlockType

The BlockType represents which type of block is location at that position. These are the block types that are defined in the Block Setup section of the World inspector. The default value 0 represents 'air' or no block at that position, other byte values represent the (index + 1) of the BlockType as setup in the inspector. Where the block at the top of the list has an index of 0.

If the BlockType takes up more than one grid unit or tile, then every single grid position that multi-block takes up will recieve the same BlockType index value.

### BlockInfo

Information about what each BlockType value represents can be found by accessing the BlockInfo of that type. The BlockInfo stores all the information about the block as setup in the World inspector. This includes but is not limited to the name of the block, the texture dimensions, the number of variations, and transparency.

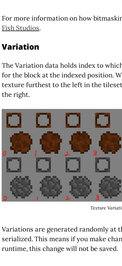
You can get the BlockInfo by calling the `GetBlockInfo` function (of the BlockLayer class) with either the BlockType value or location of a block which you want further information on.

### RenderBlock

The RenderBlock data represents the position of blocks that are to be rendered. This is important to distinguish as there are some instances in the BlockType array where a position may be filled by a block that you do not want to render. Such is the case of multi-tile blocks, where you only want to render the block texture from the origin (or bottom left position) of where that block was placed, not at every single position that BlockType is located in the BlockType array.

### Bitmask

The Bitmask is an important bit of information used to render the different edges and corners of the block textures. Each bitmask value is a number which represents the positioning of surrounding blocks. These bitmask values are 8-bit, where each bit represents a different adjacent block position. If the bit is on (1) then there is a block in that position, if it is off (0), then there is no block in that position. Below is an example calculation of a blocks bitmask:

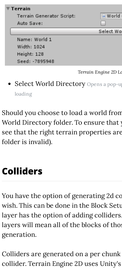



Block Bitmasking Example

For more information on how bitmasking works, checkout this article by Angry Fish Studios.

### Variation

The Variation data holds index to which texture variation is going to be used for the block at the indexed position. Where an index of 0 represents the block texture furthest to the left in the tileset, with increasing index horizontally to the right.

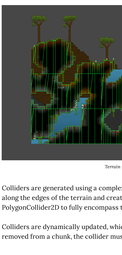


Texture Variation Index Values

Variations are generated randomly at the start of the game and are not serialized. This means if you make changes to the variation of a block at runtime, this change will not be saved.

## Terrain Properties

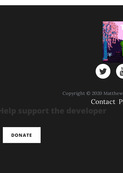
Near the top of the World inspector you will find the **Terrain** properties section. To setup these properties you will begin by adding a Terrain Generator Script which will be used to generate the terrain for the world. You can not directly drag and drop the script into the inspector field, you must add it as a component to the World GameObject. The input field will then automatically fill itself. Unless you are loading a world from file, then this must be done or else an error will be thrown in playmode. You can set whether you want to save the world you generate and/or load a world from file. When saving your world all the block data and terrain properties will be saved to file in the Streaming Assets folder when running in editor, or to Application.PersistantDataPath for builds by default.




Terrain Engine 2D Terrain Properties

- Terrain Generator Script The custom script used for procedurally generating the world
- Auto Save Saves the generated terrain to file in playmode
- Load World Loads terrain from file in playmode
- Name The name of your world (used as the save file name).
- Width The total width of the world (in block units)
- Height The total height of the world (in block units).
- Seed A integer value used to procedurally generate the world
- Random Seed Randomly generate a seed
- Generate World Generates the terrain in the editor and saves it to file

If you choose to generate a new world you will want to give your world a unique name (this is used for saving the world) and set its size. Do not set the size of the world too large or you will run into memory issues (I recommend a maximum of 1,000,000 (width x height) blocks total for running in the editor). Next, if the world is used to generate the terrain, it can be set to any positive or negative integer value, try playing around with the values to see how it affects the world you create. Note that the Generate World button allows you to generate the world and then load it right from the editor as opposed to generating the file at runtime.




Terrain Engine 2D Load Terrain Properties

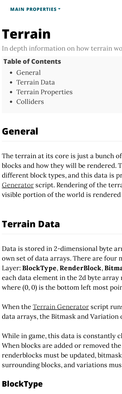
- Select World Directory Opens a pop-up menu for selecting the directory of the world for loading

Should you choose to load a world from file, you will need to select a valid World Directory folder. To ensure that you selected the proper folder check to see that the right terrain properties are displayed (an error will tell you if the folder is invalid).

## Colliders

You have the option of generating 2d colliders for any of the block layers you wish. This can be done in the Block Setup section of the World inspector. Each layer has the option of adding colliders. Adding colliders to multiple block layers will mean all of the blocks of those layers will be included in the collider generation.

Colliders are generated on a per chunk basis, meaning each chunk has it's own collider. Terrain Engine 2D uses Unity's PolygonCollider2D for chunk colliders.



Terrain Colliders

Colliders are generated using a complex recursive algorithm that traverses along the edges of the terrain and creates numerous paths for the PolygonCollider2D to fully encompass the terrain.

Colliders are dynamically updated, which means every time a block is added or removed from a chunk, the collider must regenerate.