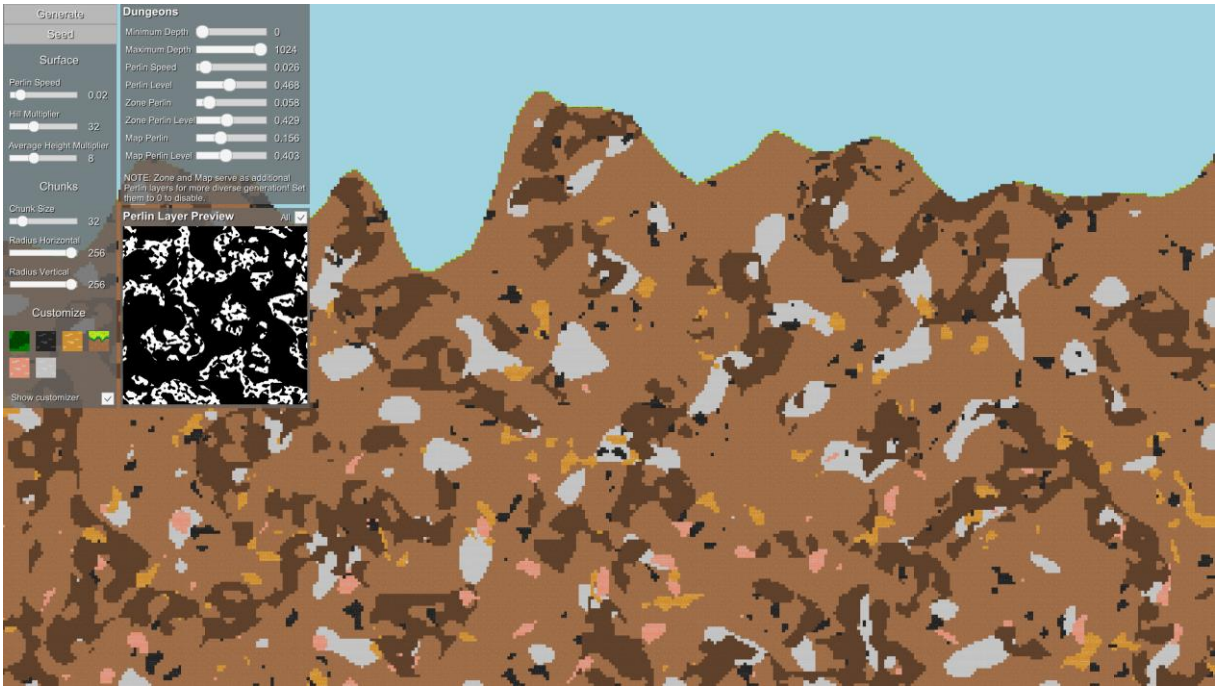
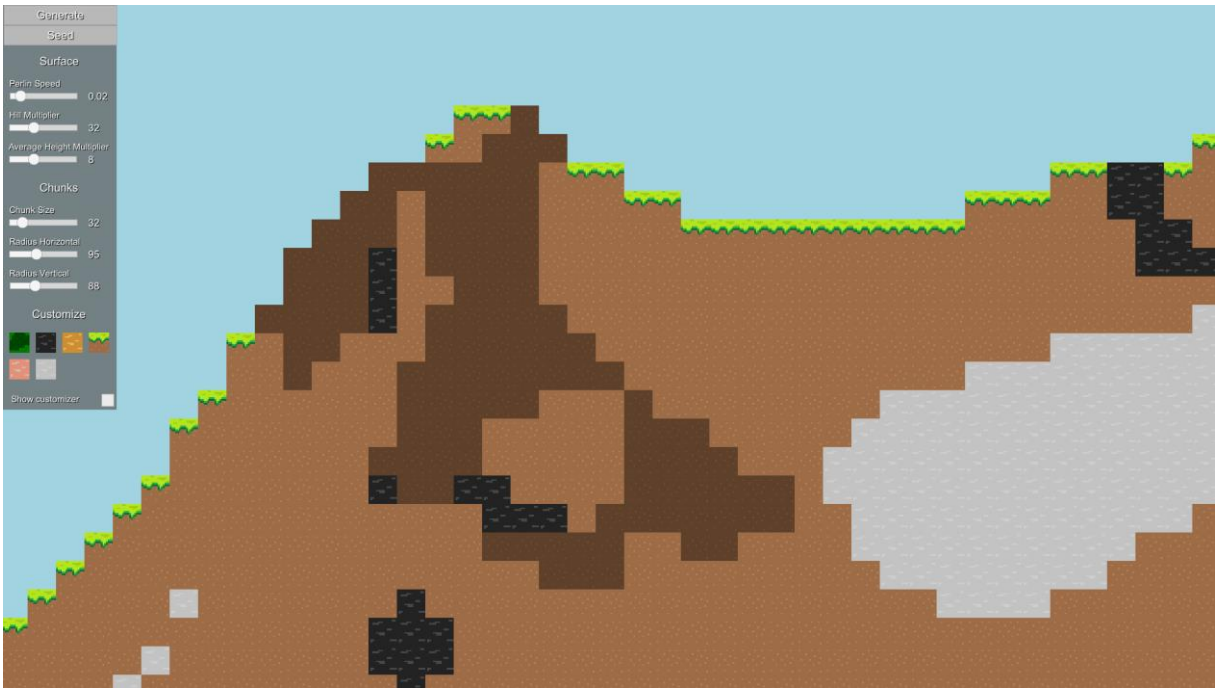
Terrainify 2D

Robronic Games





Instructions

This asset handles dynamic terrain generation with chunk loading and comes with a demo scene that includes a simple user interface to work with the asset. There are three UI sections:

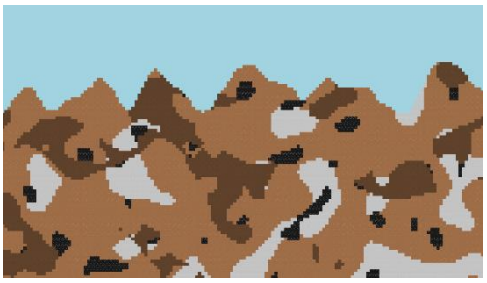
Left Vertical Bar

At the top, there are two buttons:

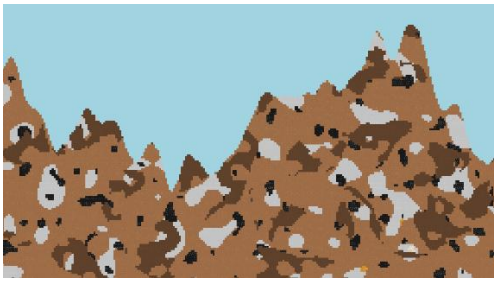
- **Generate** – Clears all generated chunks and restarts terrain building;
- **Seed** - Set a new random seed for the Generate button.

Below that is the Surface section that lets you manipulate the surface shape on a basic level. The surface is generated with Perlin noise and the sliders change the way the Perlin noise is sampled:

- **Perlin Speed** – Changes the speed at which the Perlin noise map is traversed. Higher values will result in a spiky surface, low values will have very wide hills.
- **Hill Multiplier** – Perlin noise sampling gives a value roughly between 0 and 1. Hill Multiplier multiplies this to form appropriately sized hills. The higher this value, the higher the surface hills.
- **Average Height Multiplier** – The surface is actually a double-layer Perlin noise. There is a base height and from this base height the hills are formed. This slider manipulates this base height.



Average Height Multiplier = 0



Average Height Multiplier = 10

Below that is the Chunks section:

- **Chunk Size** – Changes the size of the chunks, obviously;
- **Radius Horizontal** – Sets the horizontal border radius. Any chunks within this radius will be generated. Otherwise they will be unloaded.
- **Radius Vertical** – Sets the vertical border radius. Any chunks within this radius will be generated. Otherwise they will be unloaded.

And below that is the Customize grid. This grid shows all possible blocks and, once selected, shows their generation settings in the Customizer panel. Finally there's a simple toggle that'll show or hide the other panels we're about to discuss.

NOTE: There are a few controls and they are shown at the bottom right side of the screen!

Customizer

This panel allows you to manipulate the generation directly by changing the Perlin sampling conditions with sliders. At the top you can see the name of the block (or for example Dungeons) so you know what you edit. The sliders are pretty self-explanatory but they cover a depth range at the top and the option for three Perlin layers to enhance generation.

Perlin speed sliders change the speed at which the Perlin noise map is traversed, just like the surface Perlin Speed slider. If we take for example Coal, then the higher this value, the closer coal deposits are from each other. Do take in mind that Perlin Speed also has effect on the size of the deposits, since we scroll across the map faster.

Perlin level sliders on the other hand determine which level of Perlin noise sampling will be considered valid or invalid. If the level is 0 or close to 0, basically all samples are valid. If it is 1 or close to 1, it is rarely valid. If we take coal again and we lower this Perlin level, all deposits will become bigger. It can also generate new ones if there was a peak on the Perlin map that was very close to surfacing above our level value before, but is now valid. A good example of how you should see this is imagining a 3D mountainous landscape with a water level. All areas above the water (looking top-down) are valid, the ones underwater aren't. the Perlin Level controls this water level.

Add this as three layers and you can get away with generating a pretty solid world.

Perlin Layer Preview

This acts like an additional aid in visualizing the Perlin settings of the Customizer sliders and shows a simple black and white image. White is valid, black is invalid. An additional toggle is present that'll combine all three layers when on, or show just the layer you're currently customizing when off. Mousing over Customizer sliders will show the map.

How to implement in your game

The user interface is standalone and can be removed when necessary. It is only there to help change the settings in a structured manner. There are only three things you need in a scene to make this asset work:

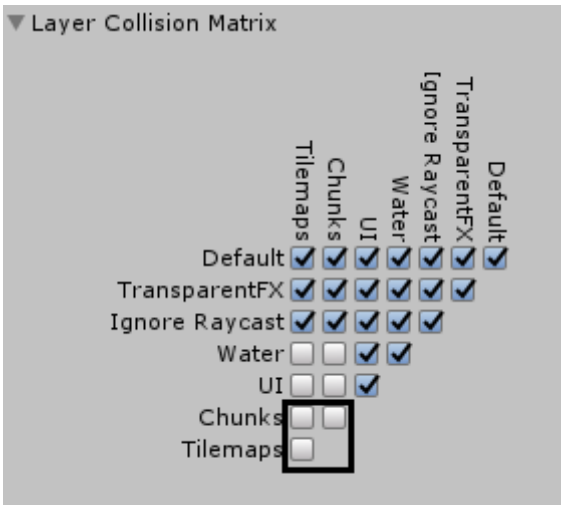
- A camera in the scene (if camera movement and zoom are needed, add **SimpleCameraMovement** script as component. WASD/keys to move the camera, scroll wheel to zoom in and out, by default in the demo scene);
- A **ChunkLoadManager**. Simply drag and drop the prefab into your scene;
- A **GenerationManager**. Simply drag and drop the prefab into your scene.

That's it. If you want to easily customize your settings, drag a **UserInterface** prefab into the scene and edit away. Remove or disable it anytime you want.

Collisions

If you want to have a player walking on the world for example, simply go to the Chunk prefab and add a **Rigidbody 2D**, **Tilemap Collider 2D** and a **Composite Collider 2D** to the **Tilemap you want to collide with**. Set the **Body Type** of the **Rigidbody 2D** to **Static** and enable the **Used By Composite** boolean on the **Tilemap Collider 2D**. The prefab should already have the **Chunks** layer tag and all **Tilemaps** on the chunk should already have a **Tilemaps** layer tag. Why composite? The Composite collider will adjust the collision shape of the **Tilemap** when blocks are added to or removed from the **Tilemap**.

Make sure chunks don't collide with each other by going into **Edit – Project Settings – Physics 2D** and disable the chunk layer and the **Tilemap** layer from interacting with each other using the **Layer Collision Matrix**. I turned this off by default for you.



Adding blocks to the system

To add a block, you only need the following:

- A sprite for the block. Currently 16x16 format is the general setting.
- **(Optional)** A sprite for the grid item that shows the block in the UI. Currently 16x16 format is the general setting.

Then:

- Add the sprite(s) to the project.
- Go to **Resources – Scriptable Objects** in the **Project View** and duplicate one of the existing ones with **Ctrl-D** or create a new one under **Create – Blocks – SliderDataCustomizer**.
- Go to **Resources – Tiles** in the **Project View** and duplicate one of the existing ones with **Ctrl-D** or create a new one under **Create – Tile.**
- **(OPTIONAL)** Open **Chunk.cs** and find the **TileType** enum. Add a new constant for your block there **at the end**. You can place it somewhere in between, but do note that there's a risk of some **Scriptable Objects** no longer having the correct type assigned for their block and you have to change them. This type is saved in the chunks to relate to the correct block type. **This isn't used in this asset by default but is for easy integration into your own game.**
- Go to the **Scriptable Object** you created earlier and give it a **name**, the **tile** you just created, select your enum constant as **type** and set the **sprite** that the **grid item** will use. Setting it to the sprite of your block would make sense if you don't have one.
- Go to the **Tile** you created and give it the sprite for your block.

Launch game and your block will appear in the UI grid.

Removing blocks from the system

Simply removing the related Scriptable Object will no longer integrate it into the generation system. You may want to remove the Tile, the enum constant and sprites from the project to remove everything though.

F.A.Q.

Hey, I get an error when dragging and dropping the UserInterface prefab into my scene "Assertion failed on expression 'modifications.empty()'"! What do I do?

This bug is common and many people experience issues with it. It seems to be linked to GUI components on prefabs, such as a Canvas. The bug does not break the **UserInterface** or the asset in any way, so I'd say just look the other way for now. It's related to dropping it into the scene at runtime so simply having it in scene and activating it when you need it avoids it.

It doesn't save my settings in build mode! Help!

This asset works with Scriptable Objects that store the slider data. The ones that store block data are imported from the Resources folder when the game runs and changes made to them will be saved only while playing in the **Unity Editor**. It is of course possible to make this save in build mode too, but is currently unsupported. You could use **PlayerPrefs** to store the data if you really want to export it or resort to **JSON** for example, but this asset was designed to work together with the Unity Editor.

A new block I added isn't showing up in the generation!

If you happened to use Duplicate from another Tile, make sure you changed the sprite to your new block! It is probably showing up, but the Tile shows the sprite of a different block.

The generated blocks don't match what's stored in the chunk!

Double check the **Item Type field** on your **Scriptable Objects**. They may get jumbled once you mess with the order of enum constants.

What dungeon settings did you use in the screenshots / demo?

I used the same settings you have when you first play the asset. I'll paste them here:

Further details

This asset is part of an ongoing series of assets towards making a full 2D world. These assets are split off from my own game as I'm creating it. I decided to make this one free! Other assets in this series:

- **Block Lighting Engine 2D** (released) – A complete lighting system for ground penetrating lighting with Unity's **Tilemap** system. Integration with this system is of course possible;
- **Terrainify 2D** (this asset);
- **Block Water 2D** (to be announced) – A dynamic water system with Unity's **Tilemap** system.

More details can be found in the code itself. Still have questions or problems? Feel free to contact me at http://robronic.com/contact/ and I'll help in any way that I can. Don't hesitate to contact me too if you found a bug, you have tips on improving the code or if you have suggestions for features you'd like to see in this asset.

Thanks for reading!