EVENT HORIZON
School of Digital Art
welcome to the knowledge era

PROCEDURAL WORLDS GENERATOR

A Unity Tool for Efficient Procedural World Creation – v1.0 (Release)

Latella Nicola / Game Programming 3 / 2023-2024

TABLE OF CONTENTS

INTRODUCTION

The Procedural Worlds Generator is a versatile Unity scripting package designed to streamline the creation of procedurally generated worlds. This tool allows developers and game designers to efficiently generate dynamic environments that are fully customizable, from terrain types to the number of chunks and the objects that populate the world.

As a package of scripts, the Procedural Worlds Generator integrates seamlessly into any Unity project. The tool provides flexibility in configuring world generation, with options to control terrain, object spawning, and performance optimizations like adjusting the level of detail (LOD) based on the player's distance. The tool also includes a pre-integrated shader for terrain materials, ensuring visually appealing environments.

In addition to its customization options, the Procedural Worlds Generator leverages advanced performance techniques, including multithreading. This allows the tool to perform complex computations and generate worlds more efficiently by utilizing multiple CPU cores, resulting in smoother and more responsive world generation.

To further optimize performance, it uses Unity’s "Addressable" package for asynchronous object spawning, minimizing resource usage and improving runtime efficiency.

Whether for large-scale games or smaller projects, the Procedural Worlds Generator offers an efficient, customizable solution for creating procedurally generated worlds in Unity.

INSTALLATION

Installation of Addressables Package

To use the **Procedural Worlds Generator**, you will need to install Unity's "Addressables" package, which is required for efficient asynchronous object spawning and resource management. Follow these steps to install the package:

1. Open your Unity project.
2. Go to **Window > Package Manager**.
3. In the top left corner of the Package Manager window, set the filter to Unity Registry.
4. Search for **Addressables** in the search bar.
5. Select the "Addressables" package from the list and click **Install**.

Once the installation is complete, your project is ready to use the **Procedural Worlds Generator** with optimized performance features.

Installing the Procedural Worlds Generator Package

Once you have downloaded and saved the Procedural Worlds Generator package on your computer, follow these steps to import it into your Unity project:

1. Open your Unity project.
2. In the top menu, go to **Assets > Import Package > Custom Package**.
3. Locate and select the **World Generation Tool.unity.package** file from your computer.
4. Click Import to bring the tool into your project.

After the import process is complete, the Procedural Worlds Generator will be available in your project, ready for configuration and use.

SETUP

Scene Setup

Before you begin using the Procedural Worlds Generator, you need to set up your Unity scene by creating a few empty GameObjects. These objects will serve as containers for the tool's scripts and will allow you to preview the generated world and its textures. Follow the steps below to create the necessary objects. The names provided are examples; you may name the objects as you prefer:

1. Create the main **Empty GameObjects**
    - In your Unity scene, create **two** empty GameObjects:
        - **Map Generator**
        - **Multithreading**
    - These GameObjects will serve as containers for the core scripts of the Procedural Worlds Generator.

2. Setup the **Preview Object**
    - Create an empty GameObject and name it Preview.
    - Inside the Preview object, create the following:
        - An empty GameObject named Mesh Preview . Make sure to add the following components to this object:
            - **Mesh Filter**
            - **Mesh Renderer**
    - A **Plane** object named Texture Preview. This object will **display the generated noise** texture and allow you to visualize the procedural generation process.

With this scene setup complete, you are now ready to assign the necessary scripts to these objects and configure the Procedural Worlds Generator for your project.

Materials Setup

In addition to setting up the scene, you will need to create and configure two materials that will be used for both the texture preview and the actual generated world. Follow these steps to create and assign the materials correctly:

1. **Create** Materials
    - In the Unity Project view, create two new materials:
        - ❖ **MapMat**: This material will be used not only for the mesh preview but also for rendering the generated terrain in the final world.
        - ❖ **PreviewMat**: This material will be used for the texture preview.

2. **Configure** Materials
    - **PreviewMat**: This material can remain as the **default** material. No additional configuration is needed.
    - **MapMat**:
        - ❖ Assign the **custom shader** named **"Terrain"** to this material. This shader is essential for rendering both the procedural terrain preview and the final generated world.
        - ❖ Adjust the **"Scale"** parameter of the **MapMat** material to fit your preferences. A default value of 10 is recommended, but this may **vary** depending **on your project needs** and the scale of your generated world.

3. **Assign** Materials to Objects
    - Mesh Preview:
        - ❖ In the **Mesh Renderer** component of the Mesh Preview GameObject, assign the MapMat material. This will allow you to preview the terrain in the editor.
    - Texture Preview:
        - ❖ In the **Mesh Renderer** component of the Texture Preview Plane, assign the PreviewMat material.

With the materials **configured** and **assigned**, your scene is now ready for the **next step**: attaching the necessary **scripts** and configuring the Procedural Worlds Generator to generate your world.

Scripts Setup

Script Setup: General Information

Please note that **all variables have an associated tooltip**. To get **additional information** about each variable and its purpose, simply **hover your mouse over** the field **in the Inspector**.

Map Preview

The "Map Preview" component is responsible for displaying a preview of the maps. Add it to the “Preview” object in the scene.

To properly configure this script in the inspector, follow these steps:

1. **Assign References:**
    - Preview Texture Render:
        - ❖ Drag the **Renderer** component associated with the Texture Preview object into the field.
    - Preview Mesh Filter:
        - ❖ Drag the **Mesh Filter** component associated with the Mesh Preview object into the field.
    - Terrain Material:
        - ❖ Assign the **MapMat** material to the field.
2. **Configure Settings:**
    - DrawMode:
        - ❖ Select the desired **draw mode** from the Draw Mode enumeration.
    - Editor Preview LOD:
        - ❖ Set the **level of detail** for the editor preview mesh.
3. **Assign Scriptable Objects:**
    - HeightMap Settings:
        - ❖ Drag the **Scriptable Object** containing the height map settings into the Heightmap Settings field (see below).
    - Mesh Settings:
        - ❖ Drag the **Scriptable Object** containing the chunk mesh settings into the Mesh Settings field (see below).
    - Texture Data:
        - ❖ Drag the **Scriptable Object** containing the world texture settings into the Texture Data field (see below).
4. **Enable Auto-Update:**
    - Auto Update:
        - ❖ Check the **Auto Update** box if you want the map preview to automatically regenerate in the editor.

With these configurations, the "Map Preview" script will be set up to visualize the procedural world data effectively in the Unity editor.

Scriptable Objects

The tool relies on several ScriptableObjects to manage and store various settings for terrain generation, including height maps, mesh data, textures, and spawnable objects. These ScriptableObjects allow you to fine-tune different aspects of the world generation process **without directly modifying the code**, providing a flexible way to experiment with different configurations.

Each ScriptableObject holds a specific type of data:

- **HeightMapSettings**: This ScriptableObject defines the noise parameters and falloff settings for generating height maps. It determines the terrain's vertical scaling and smooth transitions between different height levels.
- **MeshSettings**: This ScriptableObject manages the mesh details, such as chunk size, level of detail (LOD), and whether to use flat shading. It controls the structure of the terrain, from large landscapes to detailed sections.
- **TextureData**: This ScriptableObject handles the texture layers that define the appearance of the terrain. It allows customization of textures, tinting, blending, and scaling to achieve the desired look for each terrain layer.
- **Spawnable**: This ScriptableObject defines spawnable objects, such as trees or water bodies, and includes settings like prefab paths, spawn density, and height restrictions. It controls how and where objects are placed in the generated world.

By adjusting the parameters within these ScriptableObjects, you can **customize** the appearance and behavior of the procedurally generated world.

You can create these Scriptable Objects by right-clicking in the Project window, selecting "Create", then navigating to "Terrain Generation", and choosing the desired Scriptable Object.

HeightMap Settings

The key variables to configure within this Scriptable Object are located under Noise Settings, Height Multiplier, and Height Curve. These parameters are **crucial** as they define the vertical scaling and overall shape of the generated terrain.

Noise Settings: These control the generation of the **noise map**, which forms the basis for the terrain's height variation. This includes parameters such as noise scale, persistence, lacunarity, and octaves. Adjusting these settings will influence the roughness and features of the terrain, from large, gradual hills to small, intricate details.

Height Multiplier: This determines the **overall vertical scale** of the terrain. Higher values will result in taller and more dramatic terrain features, while lower values will produce flatter landscapes.

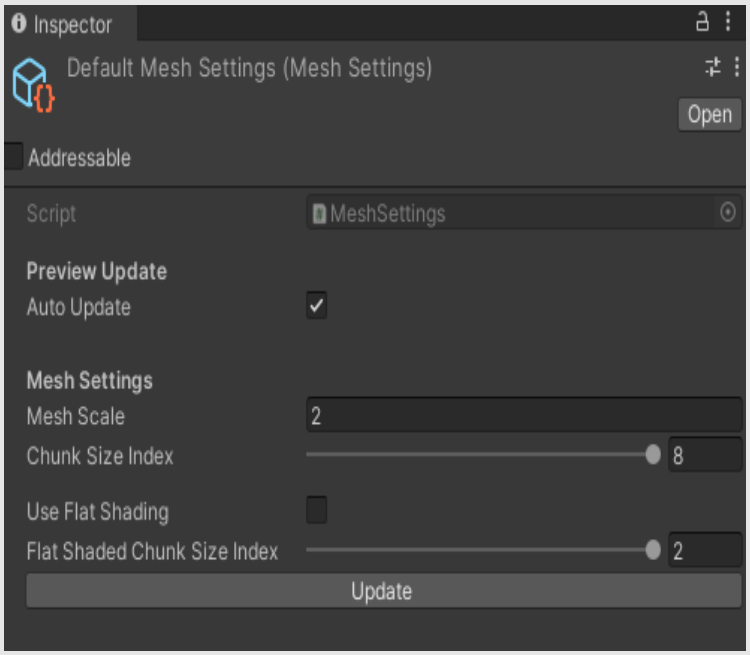
Height Curve: This curve refines how height values are interpolated along the terrain. By adjusting the curve, you can create more **complex elevation profiles**, allowing for varied terrain features like steep cliffs or gentle slopes.

Falloff Settings: These settings are **optional** and are used to apply a falloff map to smooth out the transitions at the edges of the terrain. This feature helps in creating more natural and gradual borders between different terrain types, enhancing the overall realism of the terrain.

Normalize Mode: To ensure that terrain generation works correctly across large worlds, the Normalize Mode within the Noise Settings should be set to **Global**. This mode normalizes the noise values across the entire world, rather than on a per-chunk basis. This approach is **essential** for maintaining consistent terrain elevation across different chunks, avoiding mismatches, and ensuring a smooth and continuous landscape throughout the entire map.

By carefully configuring these settings, you can create detailed, dynamic, and visually appealing terrains for your procedural world generator.

Mesh Settings



This Scriptable Object is used to configure various aspects of the mesh generation for your terrain, allowing you to customize how the terrain chunks are created and rendered.

Mesh Scale: This parameter defines the **scale** of the chunk **mesh**. Adjusting the mesh scale affects the size of each terrain chunk, with larger scales resulting in more expansive terrains and smaller scales producing more detailed chunks.

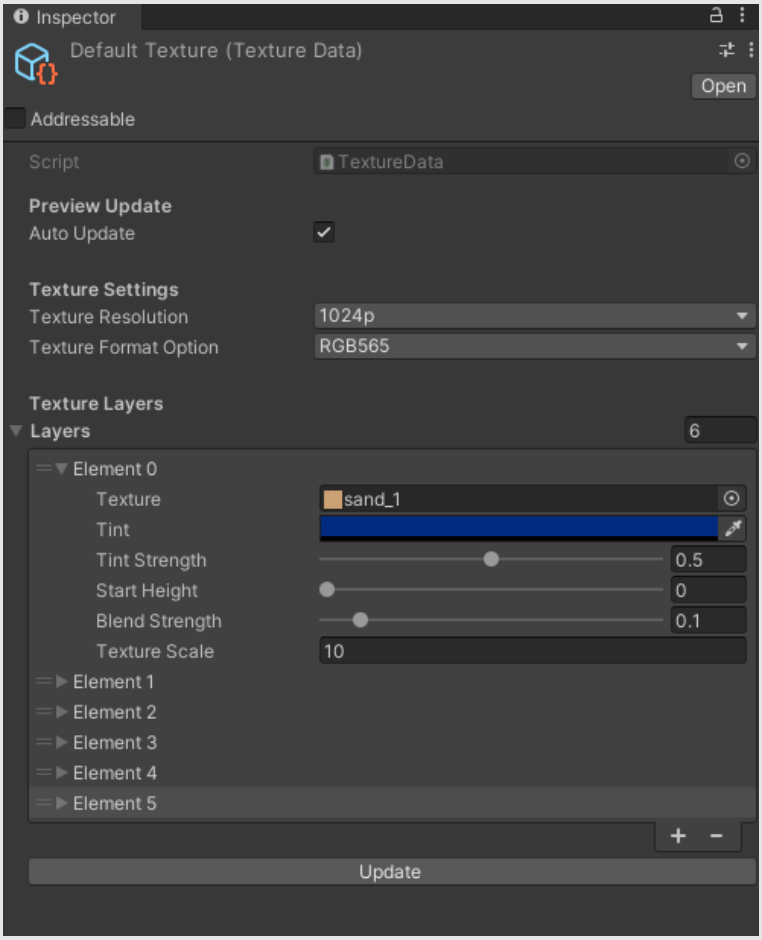
Chunk Size: This determines the size of each chunk in terms of **vertices**. Larger chunk sizes provide a higher level of detail and resolution for the terrain, but they may also increase computational demands. Conversely, smaller chunk sizes result in less detail but can improve performance, especially for large-scale terrains.

Flat Shading: These settings are **optional** and enables or disables flat shading for the mesh. When enabled, flat shading gives the terrain a **blocky**, **angular appearance**, which can be useful for achieving a stylized look. When disabled, the mesh will have smooth shading, providing a more natural and realistic appearance.

Flat Shaded Chunk Size: If flat shading is enabled, this parameter controls the **size of chunks** used for flat shaded meshes. Adjusting this setting changes the size of flat shaded chunks, similar to the chunk size parameter but specifically for flat shading.

By configuring these settings, you can tailor the **appearance** of your terrain to suit your project's needs, balancing **detail**, **scale**, and **shading** to create the perfect terrain environment.

Texture Data



The TextureData Scriptable Object is essential for managing how textures are applied to your terrain, providing control over various texture-related settings to ensure a well-integrated and visually appealing landscape.

Texture Resolution: This setting defines the resolution that all textures used for the terrain must share. It's crucial that all textures specified in the TextureData have the **same resolution** to maintain consistency and avoid issues. Matching resolution ensures that the textures blend seamlessly on the terrain surface.

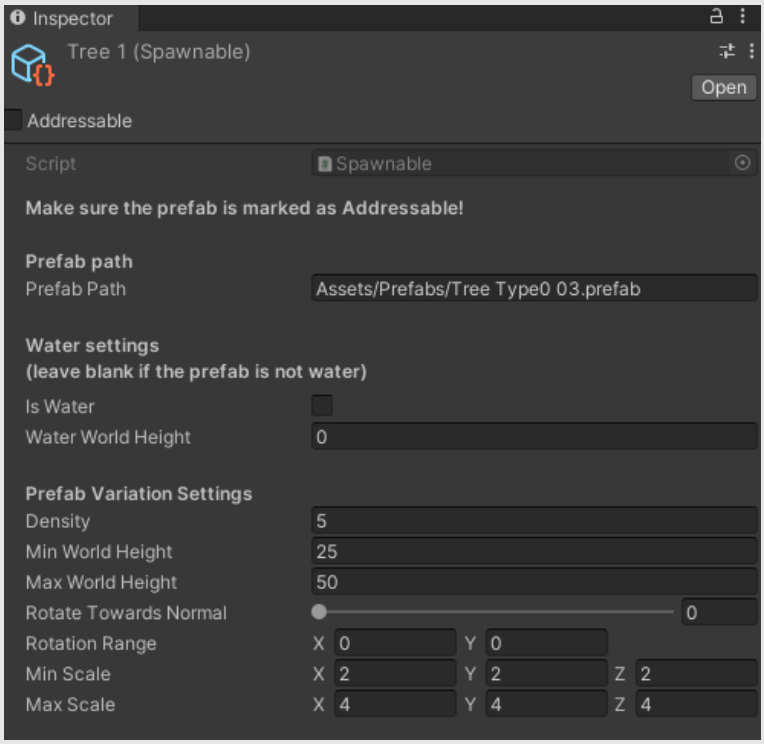
Texture Format: This option allows you to choose the **encoding format** for the textures. The texture format affects how the textures are compressed, which can impact both their visual quality and performance. Selecting the appropriate format helps optimize texture loading and rendering efficiency.

Layers: The TextureData Scriptable Object supports the use of **multiple texture layers** on your terrain. Each layer allows you to customize various aspects of how textures are applied and blended. This includes adjusting the appearance of each layer to achieve desired effects such as specific color tints, blending strengths between textures, and scaling of the textures. By configuring these layers, you can create a richly detailed and varied terrain surface.

Important: Ensure that all textures used in the TextureData are marked as **Read/Write Enabled** in their import settings. This setting is necessary to allow the textures to be properly manipulated and applied within the game, ensuring that all visual adjustments and blending functions work as intended.

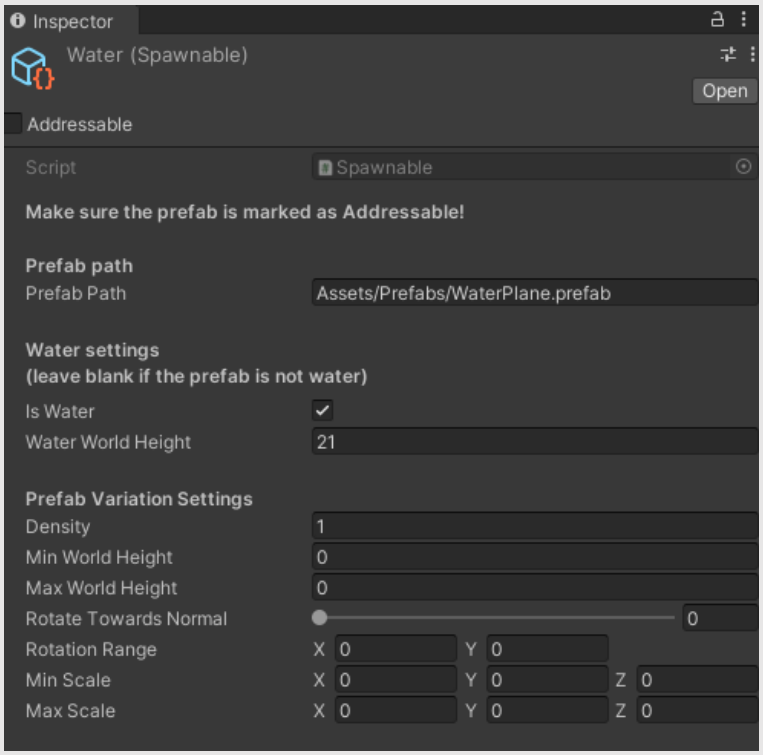
Spawnables

The Spawnable Scriptable Object is used to configure how **prefabs** are **spawned** in the terrain. It allows you to specify various parameters for prefab **placement** and **customization**, which is crucial for adding dynamic elements to your terrain.



Important Directives:

Prefab Addressability: Each prefab used with the Spawnable Scriptable Object must be **marked as Addressable**. Once marked, an addressable path will be automatically generated. This path needs to be copied and pasted into the appropriate field within the Spawnable Scriptable Object to ensure correct prefab loading and instantiation.



Water Prefab Configuration: If the prefab is designated as water, it must be marked as such within the Spawnable Scriptable Object (as illustrated in the example provided). When configured as water, the prefab will be spawned at the **center of each chunk with LOD 0**, positioned at the **specified world height**. It is advisable that the water prefab be square and match the size of the chunk or be able to overlap without causing visual artifacts.

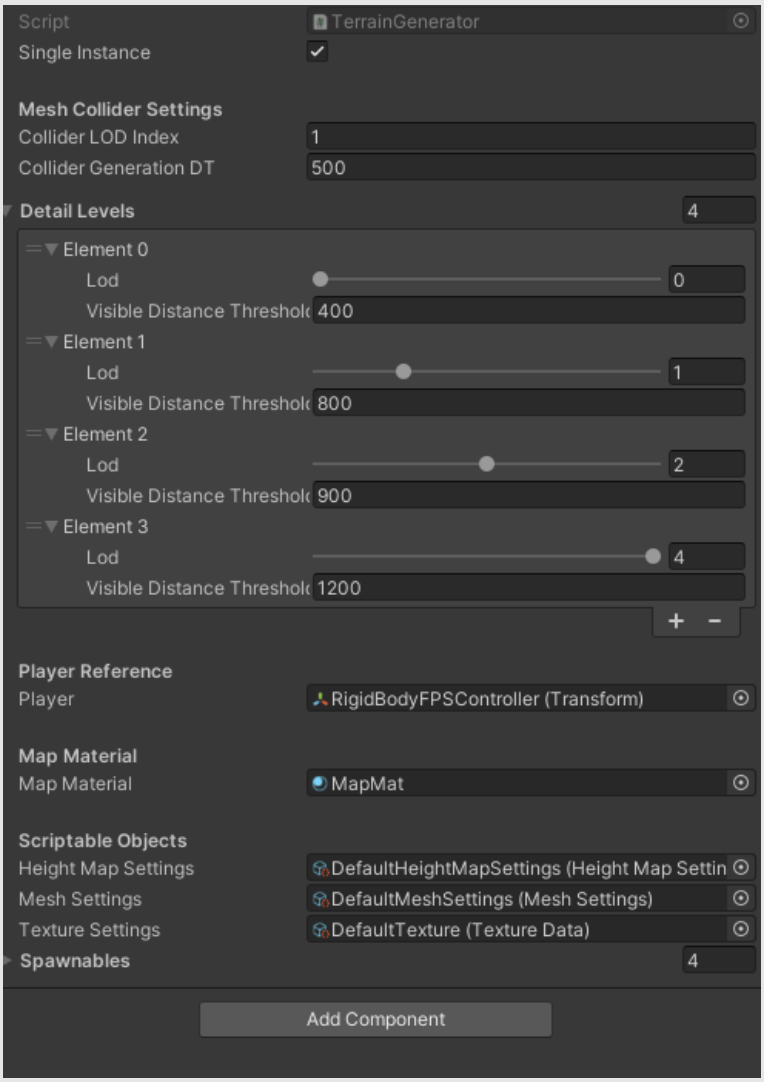
Prefab Variation Settings for Water: For water prefabs, it is generally unnecessary to adjust the settings within Prefab Variation Settings except for **setting the density to 1**.

This ensures that one instance of the prefab is spawned per chunk, aligning with the intended behavior for water bodies in your terrain.

Terrain Generator

The **Terrain Generator** is responsible for managing the **procedural generation** of terrain in the game.

On the other hand, **Multi-Threading optimizes performance** by allowing the generation processes to run concurrently across multiple threads.



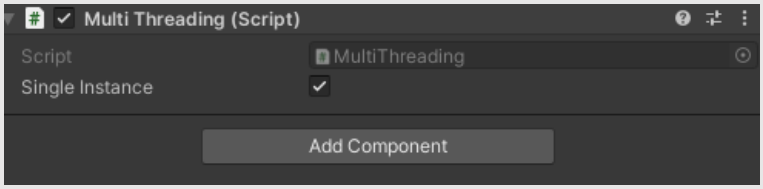
The **Single Instance** variable. This variable ensures that only **one instance** of the script exists across all scenes.

Terrain Generator Setup:

Mesh Collider Settings sets which **level of detail** (LOD) is used for the mesh colliders of the terrain chunks.

Collider Generation Distance Threshold defines the **distance from the player** at which chunks will have a **mesh collider generated**.

The **Detail Levels** array allows you to specify multiple levels of detail (LODs) for the terrain mesh. **Make sure to have at least one LOD in order to generate the world.**



The larger the **Visible Distance Threshold**, the more chunks will be visible at once, increasing the overall view distance.

Make sure to set the **reference** to the **Player/Viewer**, otherwise, the tool will not generate the terrain.

The **map material** should be the same as the one used for the mesh preview.

As a final step, make sure to assign the previously created **Scriptable Objects** to the terrain generator to ensure that all configurations are applied correctly.

After configuring all the scripts, **simply run the game**, and the world will be **generated automatically**.