# Noise Maker

Lunra Games

## What is Procedural Noise?

*Without even realizing it, you've likely seen or interacted with countless examples of procedural content through games and cinema. Whenever there is a need for random, yet reproducible content, procedural noise is there to help.*

## What is Noise Maker?

*Noise Maker is an authoring tool for procedural content, particularly environments or anything that leverages 1D, 2D, 3D or 4D noise. I'm trying to solve the following problems with Noise Maker.*

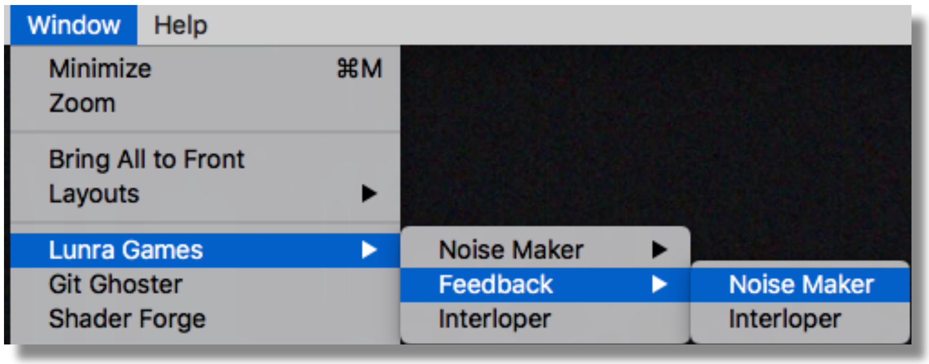
- *Noise generation is traditionally tweaked with a vast number of "magic" values*
- *Procedural content is hard for non-technical folks to work on*
- *Very few satisfactory previewing solutions exist*
- *The "Oatmeal Problem", where everything looks the same, despite being random*

*Noise Maker starts by demystifying the values associated with the generation of procedural content, allowing you to play around with them and preview their effects almost instantaneously. By leveraging a node based system, an easy way of interacting with your noise is provided, especially to anyone familiar with node based shader tools. Iteration is key, which is why multiple methods of previewing your noise exists, including flat, spherical, and tessellated spherical renderers. This allows you to be confident that the noise you're making is exactly what you want. Finally, we have the "Oatmeal Problem", as termed by Kate Compton. This is the most challenging to solve, since what makes an environment fun and exciting to explore is incredibly subjective. Noise Maker's solution is to empower more individuals on your team to author noise, by making the task less technical.*

## Thank you,

*Your support of Noise Maker helps immensely, and I hope you enjoy using it! Please provide any feedback, bug reports, or comments through our feedback portal, as detailed below.*

*- Brian Ostrander*



## Dependencies

Noise Maker leverages several 3rd party libraries and frameworks to generate its noise, all written in C# and available across all platforms. If you you have newer versions of the following, please provide feedback so that I may update the plugin. Otherwise, please use the latest version of the libraries included in with this plugin.

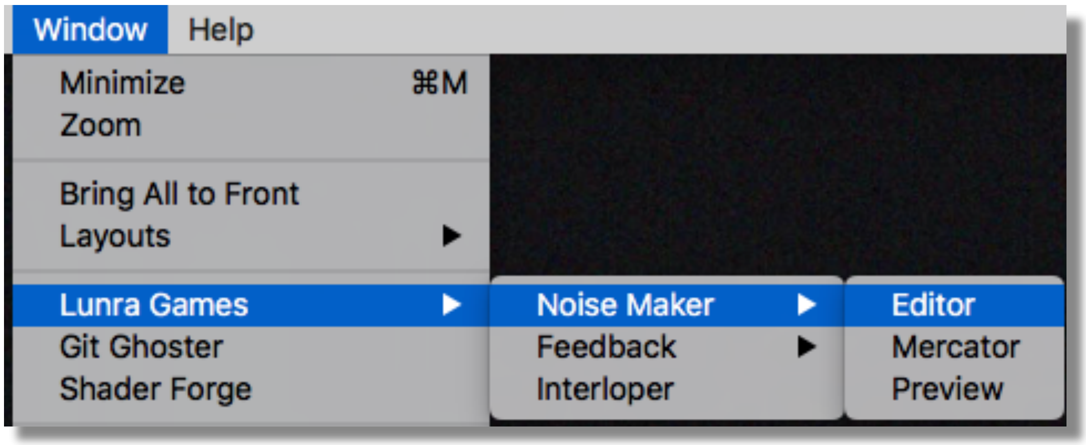
- Newtonsoft JSON.NET
- LibNoise for .NET

Also, several in-house libraries are used, and may be shared across different plugins by Lunra Games. Always keep the latest version of these libraries.

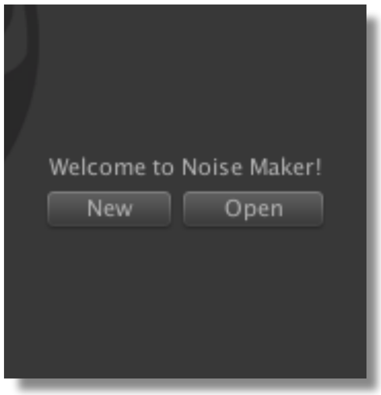
- Core
- Number Demon
- PlugIt
- Singletonnes

## Getting Started

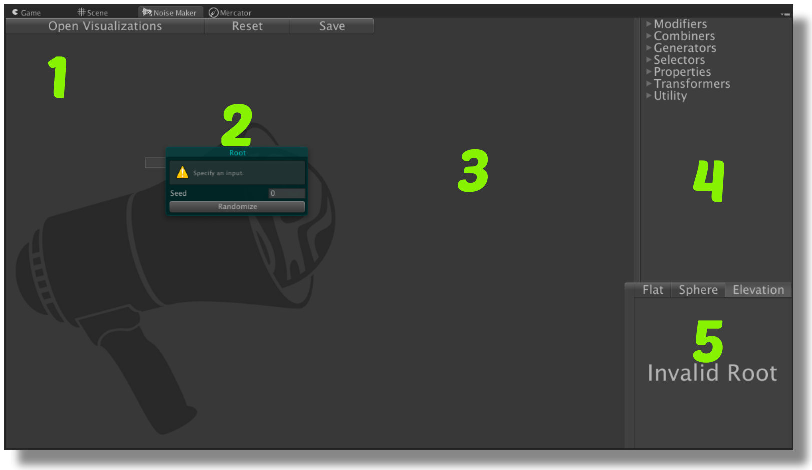
Once you have imported Noise Maker to your Unity project, open up the "Editor" window.



From here, you can create a new Noise asset, which is how Noise Maker stores the node based data of your procedural noise. Select "New" and save the noise asset somewhere in your project directory.



## The Editor



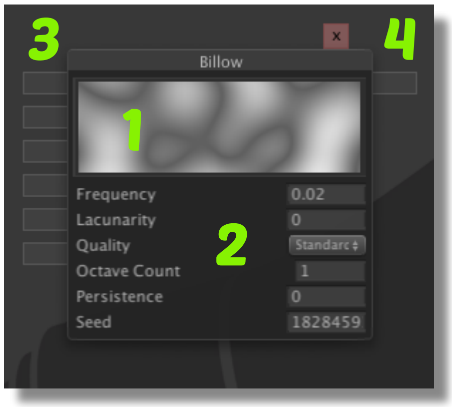
Specify an input.

Seed 0

Randomize

Invalid Root

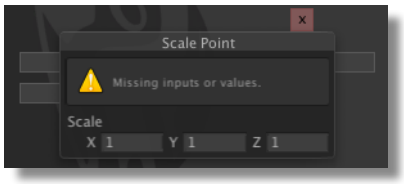
When you start editing a new noise asset, you'll see the following areas. The visualizations toolbar[1] is used to switch between different color palettes for previewing noise. Every noise asset includes a root node[2], where the noise asset's seed and sources are defined. This node, and any others you create, can be arranged in the graph area[3]. The node list[4] is used to spawn nodes for editing. Finally, the preview area[5] is where you can see the output of your root node. You can preview your noise as a flat heightmap, a spherical heightmap, or a tessellated sphere. Note that your noise will be previewed at a low resolution to speed up the previewing process. When you use noise asset in your application, you can request for values at any resolution.



Frequency 0.02
Lacunarity 0
Quality Standard
Octave Count 1
Persistence 0
Seed 1828459

## Nodes

To spawn your first node, click the "Billow" button in the right hand node list. This will spawn a billow module in the graph area. This module creates a gentle rolling noise, as seen in the preview area[1]. Below that is the property area[2], full of values relevant to each noise module. On the left hand side of the node are inputs[3]. These can be used to connect other nodes to your existing ones, or define shared property nodes. Each has an associated type, and will warn you if you try to make an invalid connection. Each node will have an output[4] for connecting it to another node, or to the root node. Nodes can be moved by dragging their headers, and removed by clicking the red "x" above them.

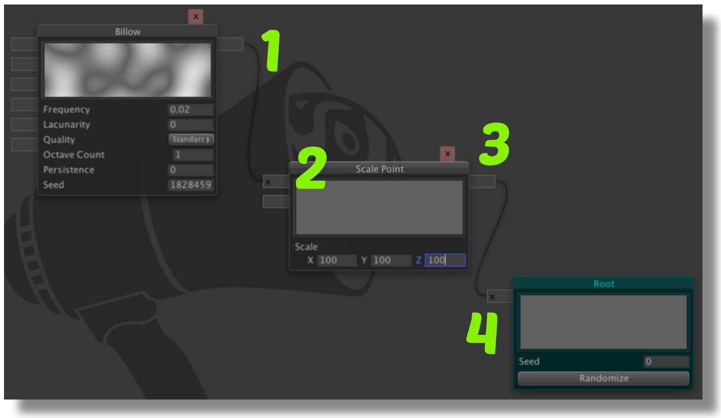


Missing inputs or values.

Scale
X 1 Y 1 Z 1

Next, spawn a "Scale Point" module, which takes a module as an input, and scales the detail of the noise by the provided x, y, z values. If you're using the "Sphere" or "Elevation" previews, this will be important, since the noise made by the billow module is too large to be seen without scaling.

## Connecting Nodes

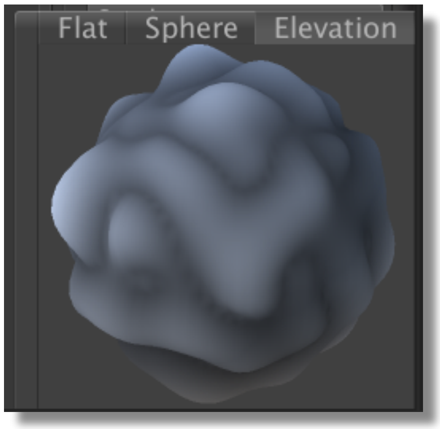


Frequency 0.02
Lacunarity 0
Quality Standard
Octave Count 1
Persistence 0
Seed 1828459

Scale X 100 Y 100 Z 100

Seed 0

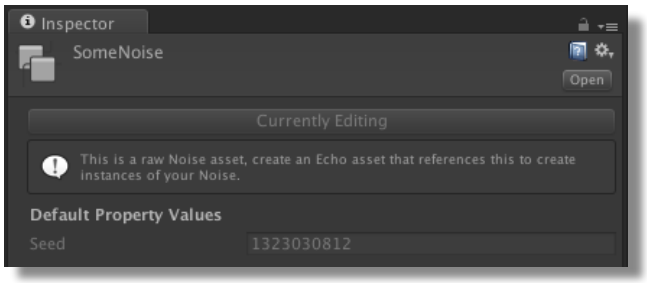
The power of Noise Maker comes from the ability to combine these discrete nodes. To do this, click on the output of the billow node[1] and connect it the first input of the scale point node[2]. After playing around with the values in the scale point node, you'll see it shows a preview of the billow node scaled with the values you specify. Remember, your noise is of an arbitrary number of dimensions, so these previews are only showing you a 2D slice of the actual noise. To use billow noise in a meaningful way with the elevation preview, set the x, y, and z values of the scale point module to 100.



Finally, connect the output of the scale point node[3] with the input of the root node[4]. With the elevation preview selected, you should now see something like the noise generated here. Keep adding new nodes and tweaking their values, that's the beauty of Noise Maker, you can learn by doing! Using the nodes in the "Combiners" section, you can take different noise generators and blend them together in various ways. Using nodes in the "Properties" section allows you to define individual properties of each node, and share them across other nodes.

## External Properties

When you save your noise, and take a look at the asset, you'll see a a list of the the default values exposed by your asset. Currently it will only show one "Seed".



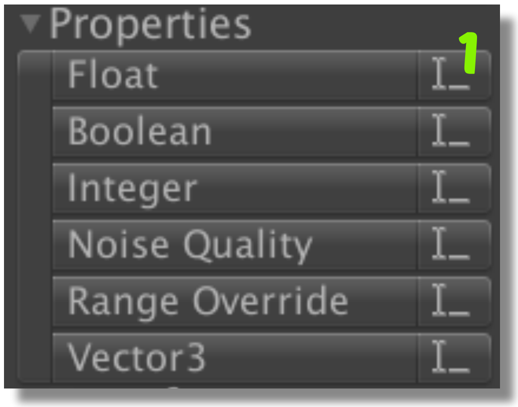
SomeNoise

Currently Editing

This is a raw Noise asset, create an Echo asset that references this to create instances of your Noise.

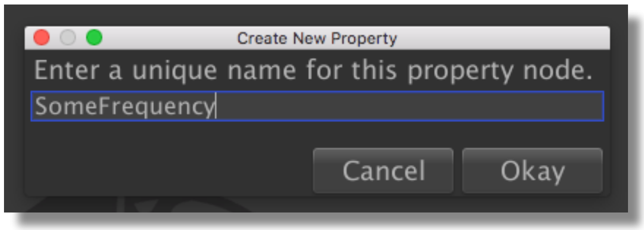
Default Property Values

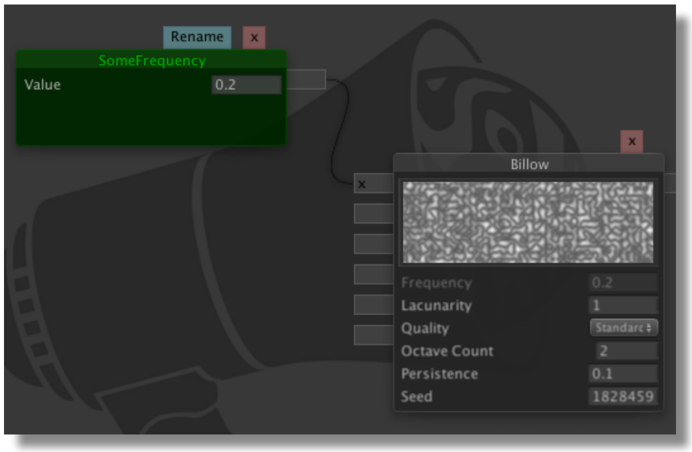
Seed 1323030812



If you want to expose other properties in this menu, simply go to the properties section and click the button to the right of any entry[1].

You'll be prompted to enter a unique name. This is the name that will appear on the asset's inspector, so pick something meaningful.

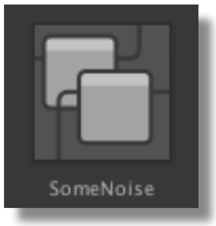


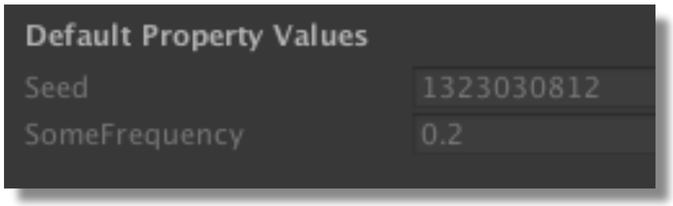


| | |
|---|---|
| Frequency | 0.2 |
| Lacunarity | 1 |
| Quality | Standard |
| Octave Count | 2 |
| Persistence | 0.1 |
| Seed | 1828459 |

Once spawned, external property nodes show up as green, and can be used to override the value of any properties of the same type on other nodes.

Find the instance of your Noise asset in the project window. It will have an icon like the one to the right.





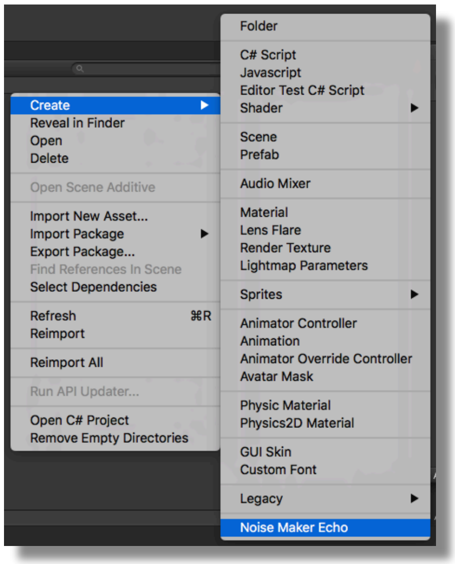
Default Property Values

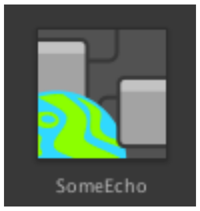
| | |
|---|---|
| Seed | 1323030812 |
| SomeFrequency | 0.2 |

Now you'll see your value exposed along with any others you've defined. However, they will not be editable yet. This is because a Noise asset is abstracted from an actual instance of your noise.

## Creating Echos

***To create a usable asset instance of your noise, you'll need to create a new Echo asset. These allow you to create many different versions of the same noise, where each Echo defines a different set of properties. This relationship is very similar to how shaders and materials work. Some properties, like Texture2Ds can only be set in Echo instances or through C# scripts. Select "Noise Maker Echo" from the "Create" menu to begin.***





***You'll see your Echo asset with an icon like this appear. Click on it and the inspector will show a field for selecting a Noise asset.***

***You can also create Echos from a C# script if you have a reference to a Noise asset.***

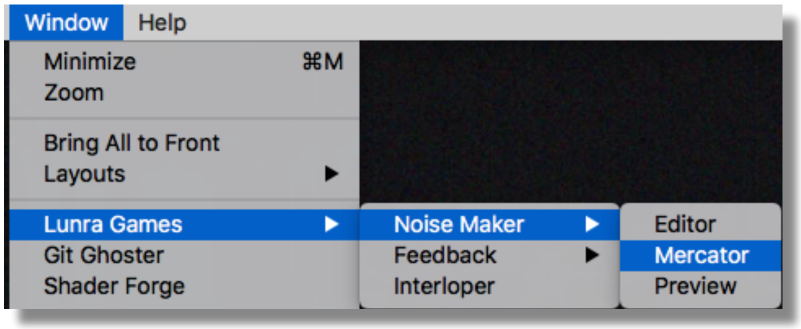
```
// Set in the inspector
public NoiseAsset NoiseAsset;

void Start()
{
    // A new echo for you to request values from
    var echo = new Echo(NoiseAsset.Noise);
```

```
var noise = Noise.FromJson(json);
var echo = new Echo(noise);
```

***If you have the raw Json for a Noise asset, you can also create it with the converter method in the Noise class.***

## Mercator

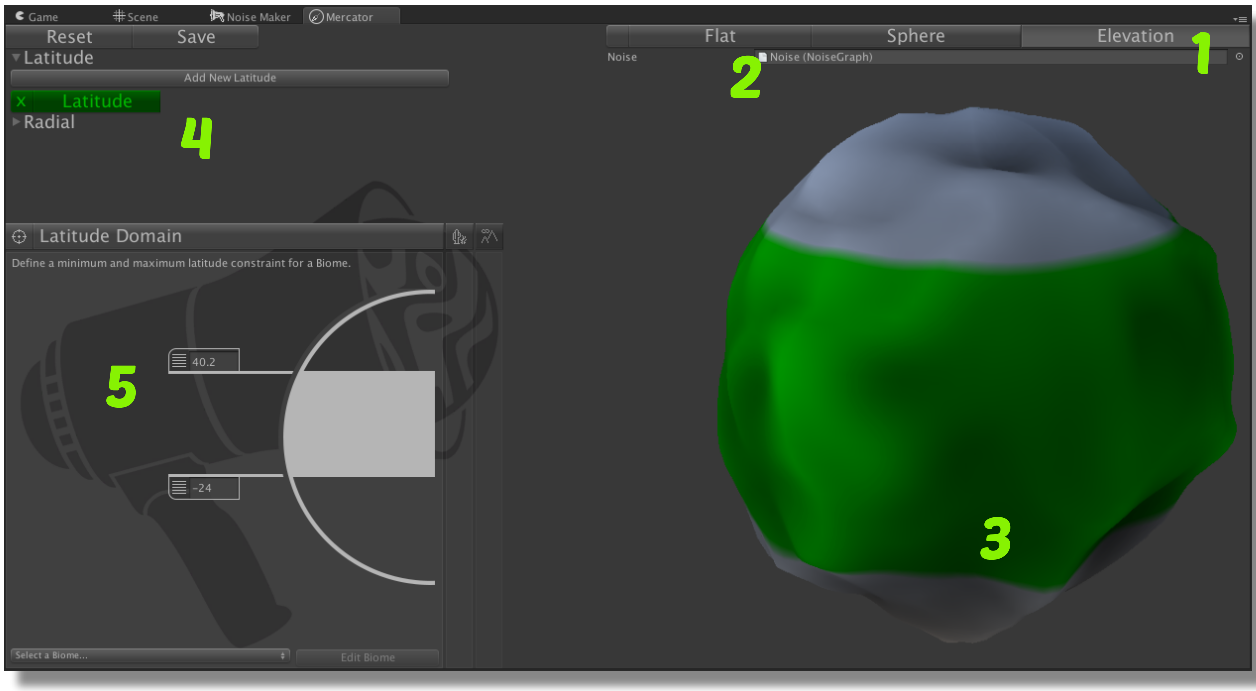


***Procedural content isn't much use without the ability to skin it, and that's where Mercator comes in. Mercator allows you to define a procedural texture for your heightmaps. To get started, open the Mercator window.***

***Once open, create a new map asset and save it to your project. Mercator assets can be used with any Noise asset you make, allowing you to mix and match them as you please, for ultimate flexibility. Upon creating a new Mercator asset, you'll see a much larger preview area to the right. Select "Elevation" in the preview bar[1] to see what your noise would look like applied to a tessellated sphere. Now select a previously made Noise asset to the "Noise" field[2]. You will see a nice large render of your noise with the default visualization applied to it[3].***

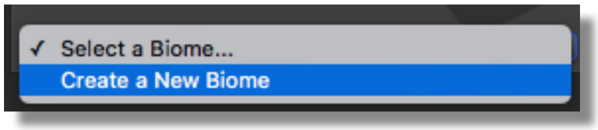
## Domains

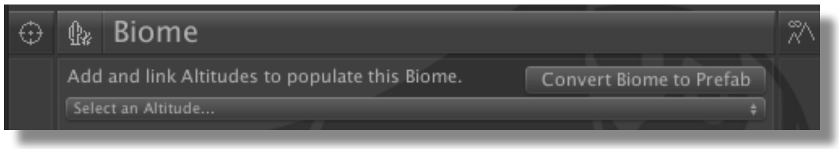
***To the top left is the domain area[4], from which you can create new domains. Domains are Mercator's way of defining an area in which to apply your color ramps. Click "Add New Latitude" to define a new domain between two latitudes in the editor area[5]. This domain is specifically for spherically projected noise, and will not work with flat noise. Multiple domains will blend together if they overlap.***



## Biomes

In Mercator, biomes are used to group collections of altitude based color ramps. To create a new biome, click the dropdown "Select a Biome..." menu, and click "Create a New Biome". The domain editor will now switch to a biome editor. To switch back to the domain view, simple click the crosshair at the top left.

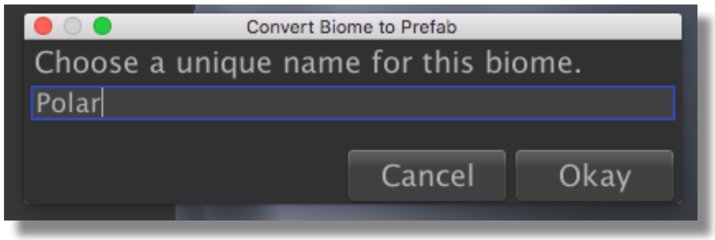




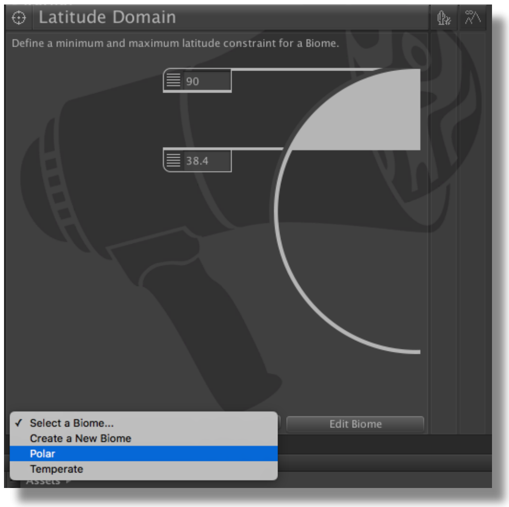
Add and link Altitudes to populate this Biome.

Since you may want to use biomes across various domains, an option to "Convert Biome to Prefab" exists.

Click on it, then give the biome a meaningful name in the popup dialog.



Choose a unique name for this biome.
Polar

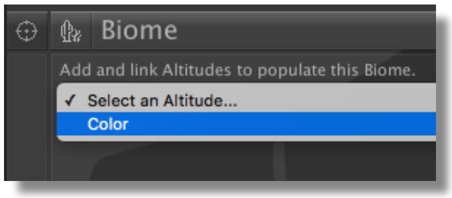


Define a minimum and maximum latitude constraint for a Biome.

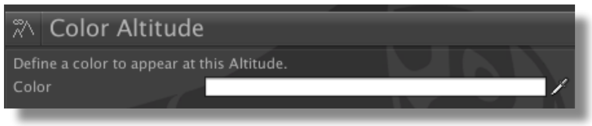
Once you've clicked "Okay", go back to the domain editor, and click "Select a Biome...". You will now see your biome listed. To use this biome in any other domain, simply select it from the drop down menu. Changes you make in any instance of this biome will be reflected in all others. This is perfect for simultaneously changing both polar caps of a world, or all the forests at different latitudes.

## Altitudes

Finally, Mercator uses altitudes to define what pixels are drawn at a specific altitude within a biome. To create a new altitude, go the biome editor of any domain and click "Select an Altitude". You will see an option for "Color", select that and you'll be taken to the Altitude editor.



Add and link Altitudes to populate this Biome.

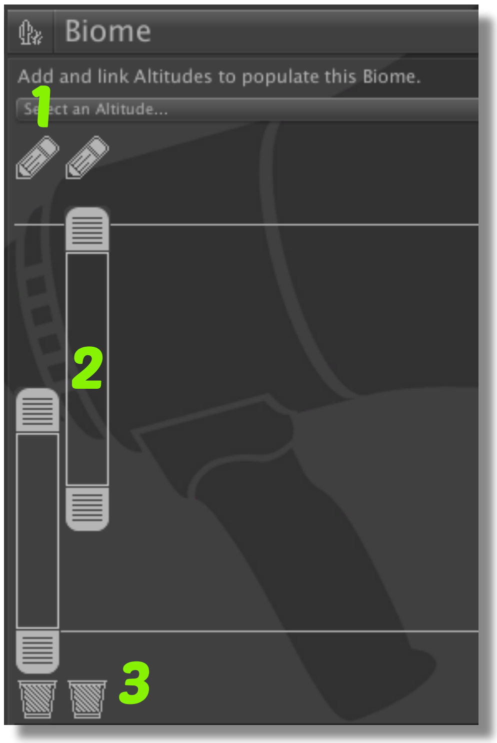


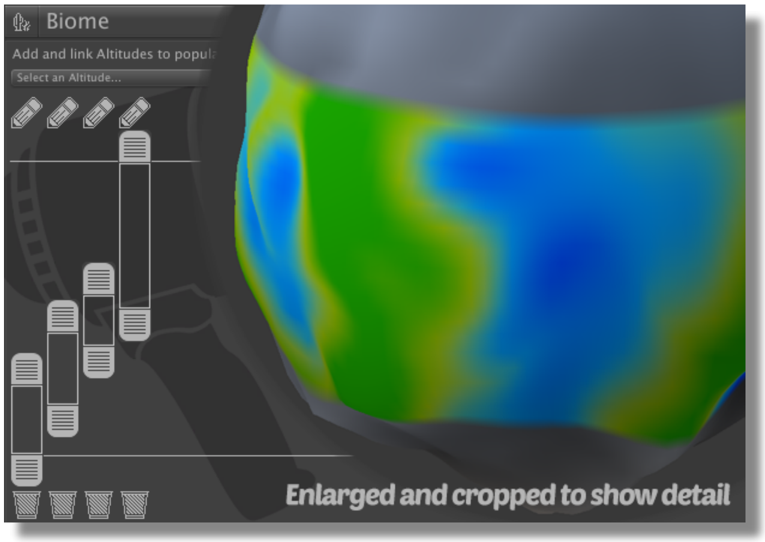
Define a color to appear at this Altitude.

From here, you can select the color that this altitude represents.

## Arranging Altitudes

Within a biome, altitudes are represented as entries going from the left side of the editor to the right. To return to the altitude editor, click the pencil button[1] above an entry. To adjust the range in which this altitude appears, click and drag the altitude's slider[2]. To remove an altitude that you no longer need, click the trash button[2] at the bottom of the entry.

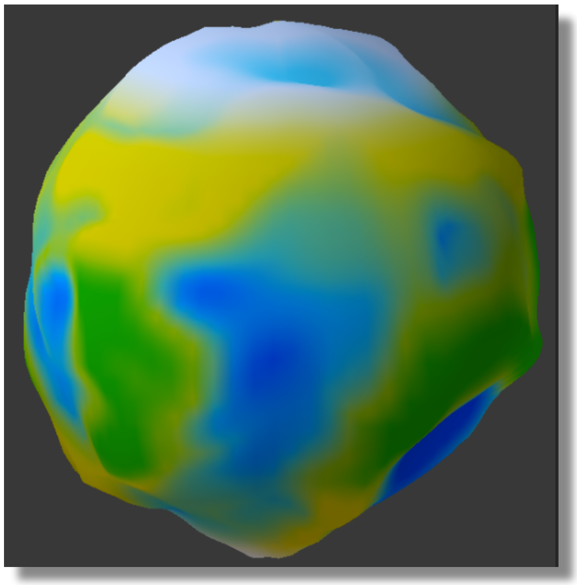



Biome
Add and link Altitudes to popul
Select an Altitude...
Enlarged and cropped to show detail

Keep mixing and matching various altitudes to create the effect you desire, and you'll see the changes reflected live in the preview to the right.

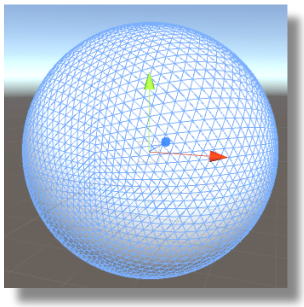
## Complete Preview

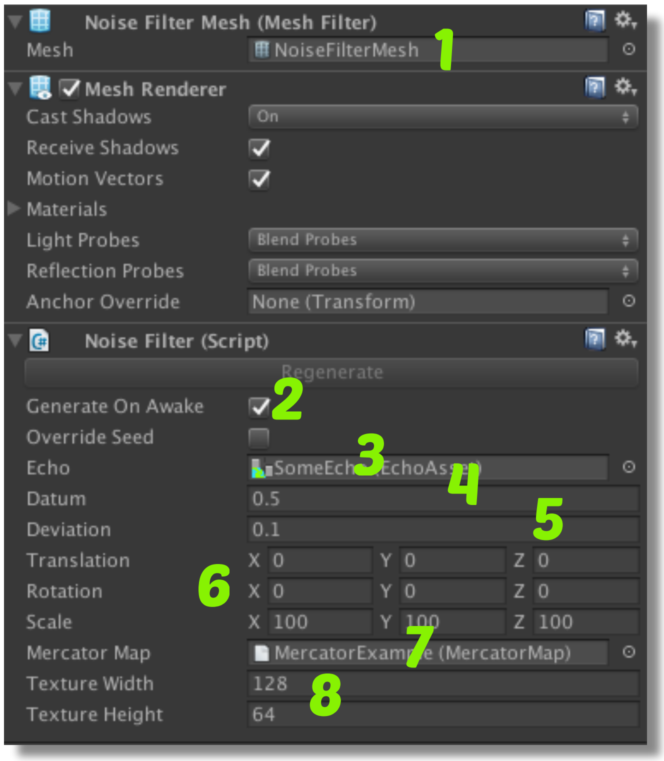
To see the entire preview, deselect any domains you're working on in the top left of the editor. When none are selected, the preview area to the right will show a low resolution preview of your entire map. Remember, Mercator can output textures at any resolution, so any blurriness in the preview is an artifact of the quick render time.



## Runtime Generation

**Finally, it's time to see the fruits of your labor in-game! Start by creating a new game object with a mesh filter, mesh renderer, and the custom "NoiseFilter" component. The noise filter works by copying and modifying the mesh filter's geometry[1], meaning you don't need to worry about your original geometry being destroyed.**

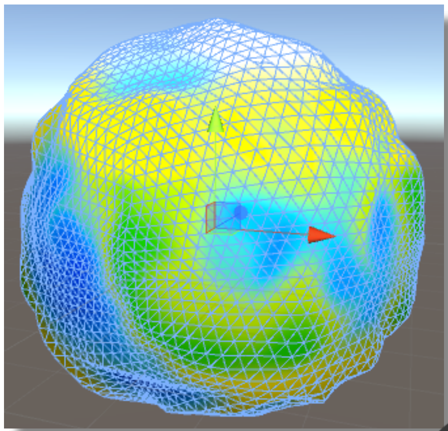




**By checking the "Generate On Awake" box[2], this noise filter will create its mesh and materials upon receiving the awake event. To trigger it manually in code, simply call the noise filter object's "Regenerate()" method. Specify an Echo asset[3] you made earlier, which will be used to generate the offsets for the mesh vertices. You can consider datum[4] as the sea-level of your noise, and deviation[5] as the fraction of datum that your noise will be multiplied by.**

**Datum + (Noise * Datum * Deviation)**

**So if your datum is 1.0, your deviation is 0.5, and the noise graph output is 0.2, the final result is 1.1. Additionally, you can further customize the final geometry of your noise by editing its transform values[6]. Specifying a Mercator asset[7] allows the filter to texture the resulting mesh. It's important the mesh you've supplied is properly UV wrapped to a 2:1 width to height ratio, otherwise distortion will appear around the poles. Included with Noise Maker are several meshes perfect for this purpose.**



## Congratulations!

**Upon hitting run, you'll see your first terrain procedurally generated with Noise Maker and textured by Mercator! If you create anything cool, please email some screenshots of your work to LunraGames@gmal.com, I'm always interested in new images for the store page.**

## What About Tessellated Planes?

Noise Maker is a big project, so not every feature got into this first beta release. This includes noise filter support for planes. However, you can access the LibNoise for NET's data to generate tessellated planes by grabbing your Noise's "Root" IModule property. If you want to vote on this and other features that are important to you, please check out our active tickets here. If you have ideas for new features, or bugs that you'd like to report, please don't hesitate to submit feedback here.

## Special Thanks

The individuals and organizations below were incredibly helpful in the creation of Noise Maker, thank you so much!

- Creators of LibNoise for NET, who made such a kick-ass open source library
- Newtonsoft Json.NET
- Bailey James, for removing all the exclamation points from my marketing copy
- Ian Earle, who provided feedback for logos and designs on countless occasions
- The members of r/gamedev's IRC channel, for tolerating my never-ending sessions of show and tell
- And to the real heroes, those with the patience to answer my inane questions in the Unity IRC channel!