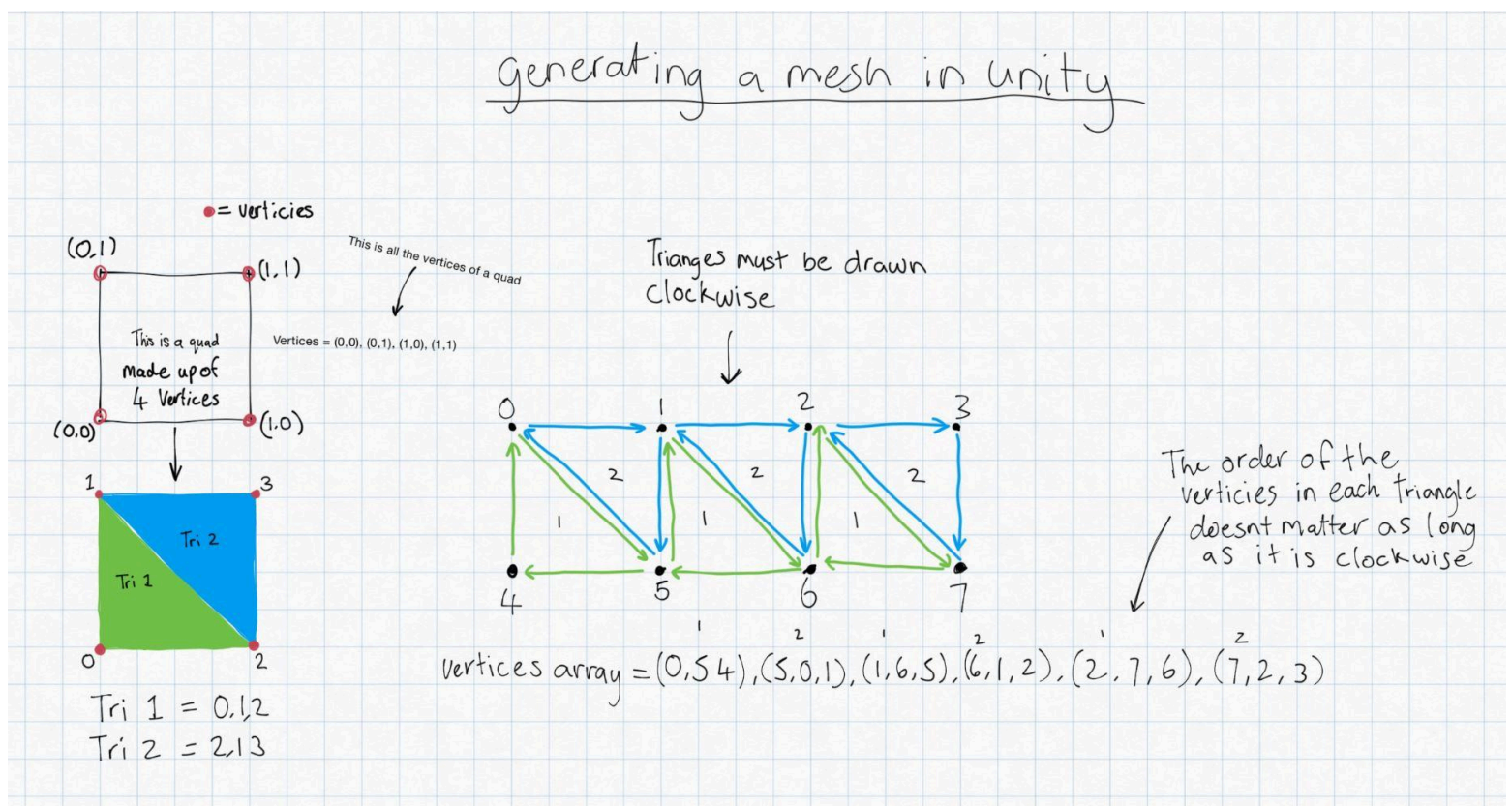


Gad 176.2 Feature Spec

Reuben Poole

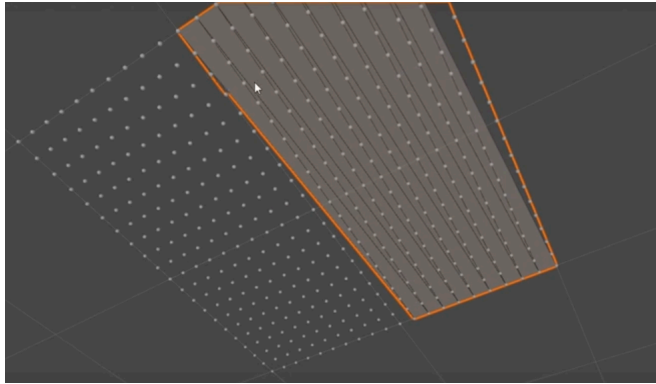
Terrain Mesh Generation:



In order to generate the mesh for my terrain in unity I did a ton of research on how a mesh is generated. In my project the mesh is made up of Quads, and each Quad consists of 2 triangles.

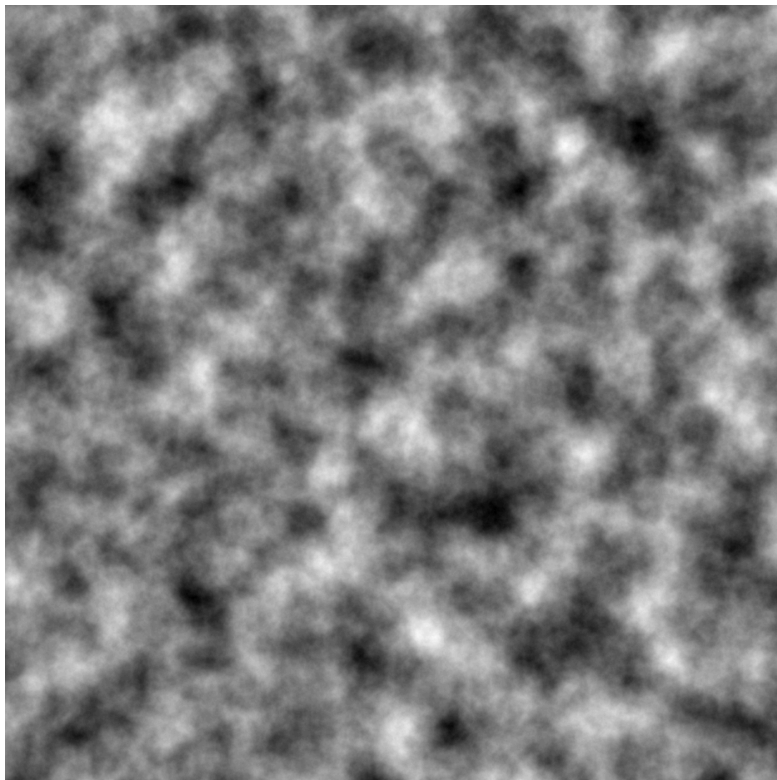
- Each triangle in my mesh consists of 3 vertices.
 - When generating triangles you must add each of the vertices to an array.
 - It does not matter which vertex starts the array, however the proceeding vertices must be placed in a clockwise direction. (As seen in the diagram above).

- It is also important to note that when generating a grid of triangles. You need to make sure that you don't generate the last triangle in the array because this will cause it to be generated twice. Resulting in unwanted results with the normals of your mesh.



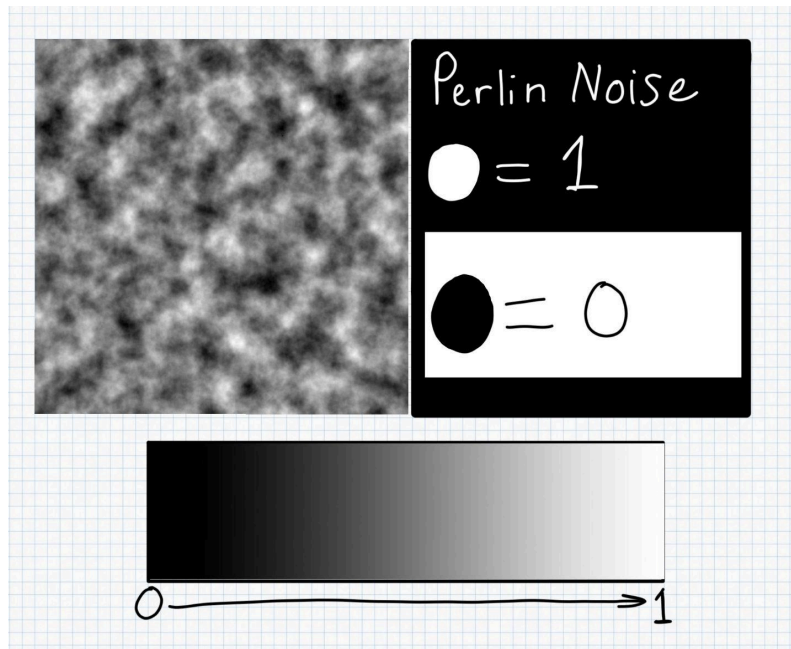
This can be seen here where the last triangle is being generated using the vertices from the next row along in the grid.

Perlin Noise:



In order to transform my square mesh grid into a natural mountainous terrain. I used perlin noise in order to control the y value of each of the vertices in the mesh.

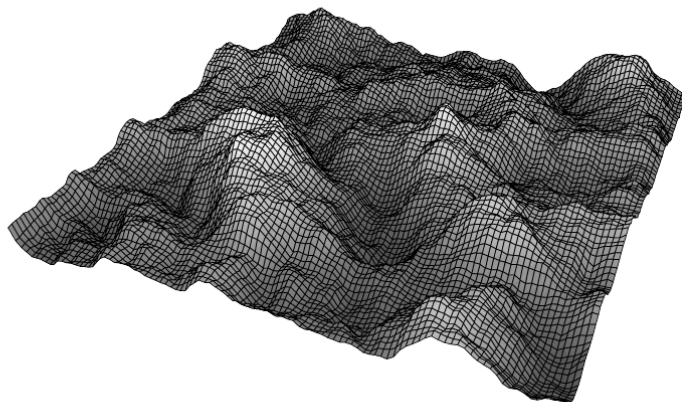
This is an example of some perlin noise I generated whilst following Sebastian Lague's tutorial on procedural landmass generation (Lague, 2016).



A perlin noise image consists of 2 shades, Black, and White.

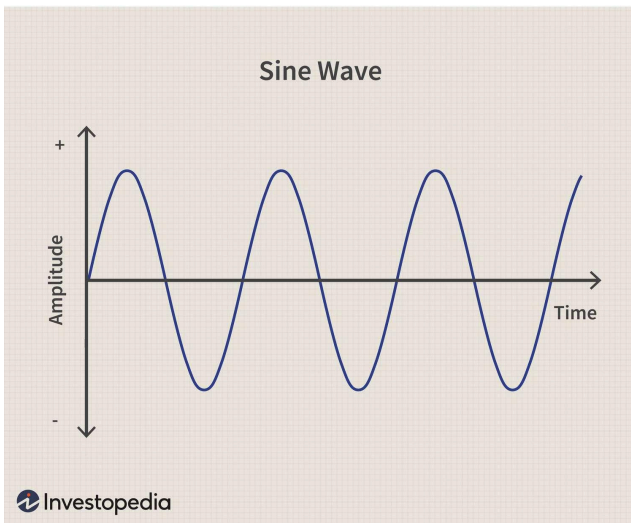
In order to convert Perlin noise into a height map to generate a mesh I am using the two shades as a gradient. With 0 being black, and white being 1. This means that I can now use the float value from 0 to 1 in order to control the y value of my mesh.

This is done by multiplying the y value of each vertex by the perlin noise value.



Here is a simple grid mesh that has had each of its vertices y values multiplied by Perlin noise.

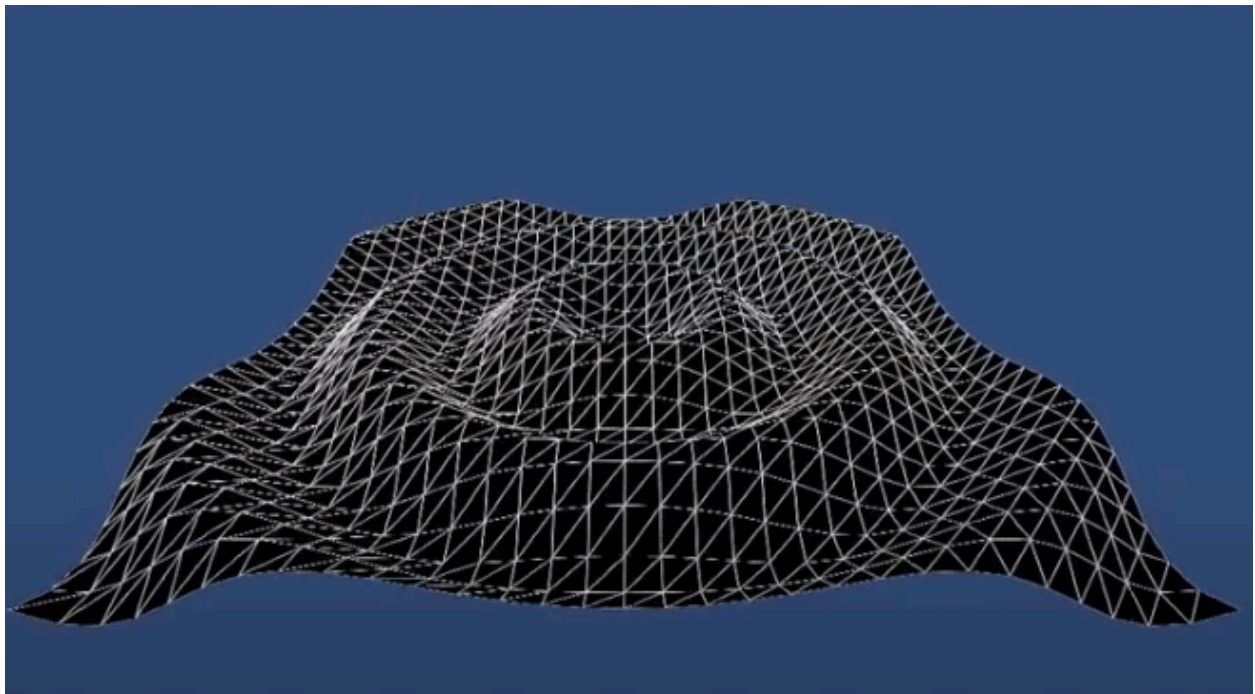
Sine Wave



In my package I am creating an island. Therefore it is important to have some kind of fall off to the perlin noise value.

To create a fall off for the noise in my terrain I decided on using a sine wave. The Perlin value is then multiplied by the value created by the sine function on all vertices.

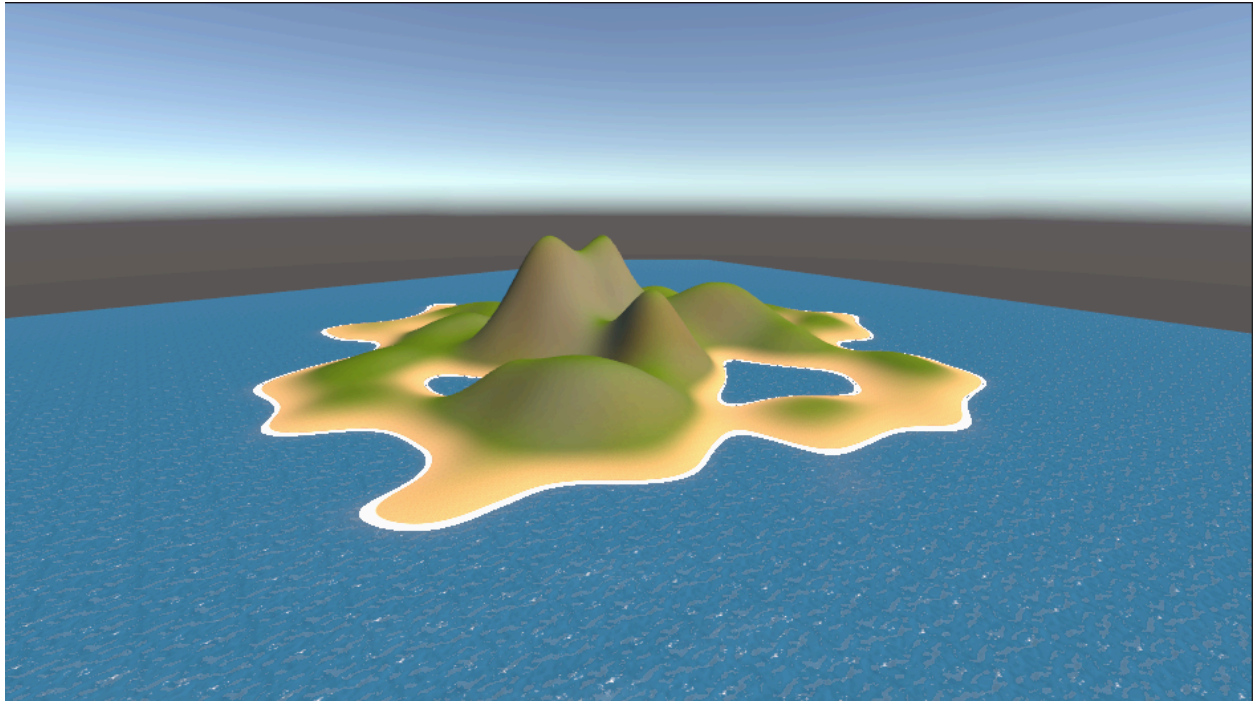
In order to multiply the Perlin value by a sine wave it is necessary to find the center of the generated mesh.



After multiplying the y values of a mesh by sin the mesh looks like this. After tweaking the amplitude of the wave it is possible to get a more island result by decreasing the frequency between each high and low of the sine wave.



And then when I combine the sine wave and the perlin noise the result looks like this:



Features in my island generator package:



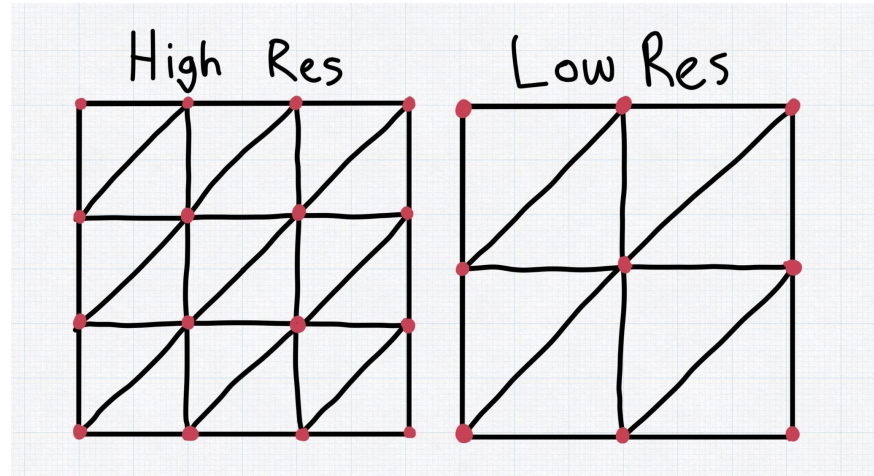
World Size	<p>The world size is a drop down that controls how many chunks are generated in the terrain.</p> <p>World size is a vector 2 and I have set it up to be consist of 3 different size options:</p> <ul style="list-style-type: none">- Small = 4 x 4- Medium = 8 x 8- Large = 16 x 16
------------	---

Resolution

The resolution controls how many vertices are in the mesh.

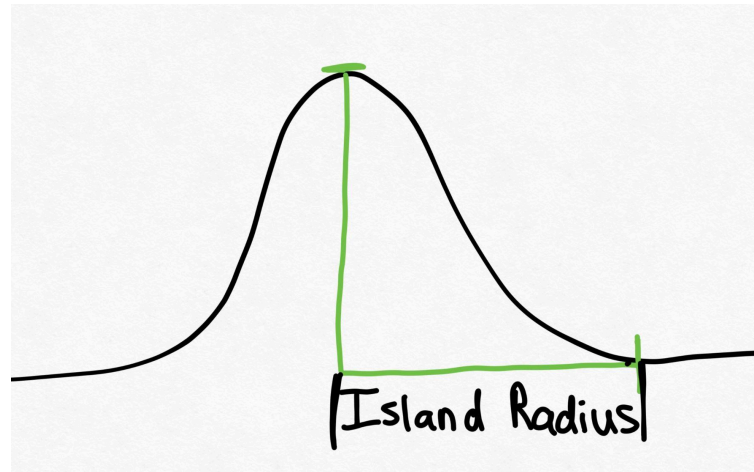
I have set up the resolution so that it is a dropdown of low, medium, and high.

- Low = 16
- Medium = 64
- High = 128

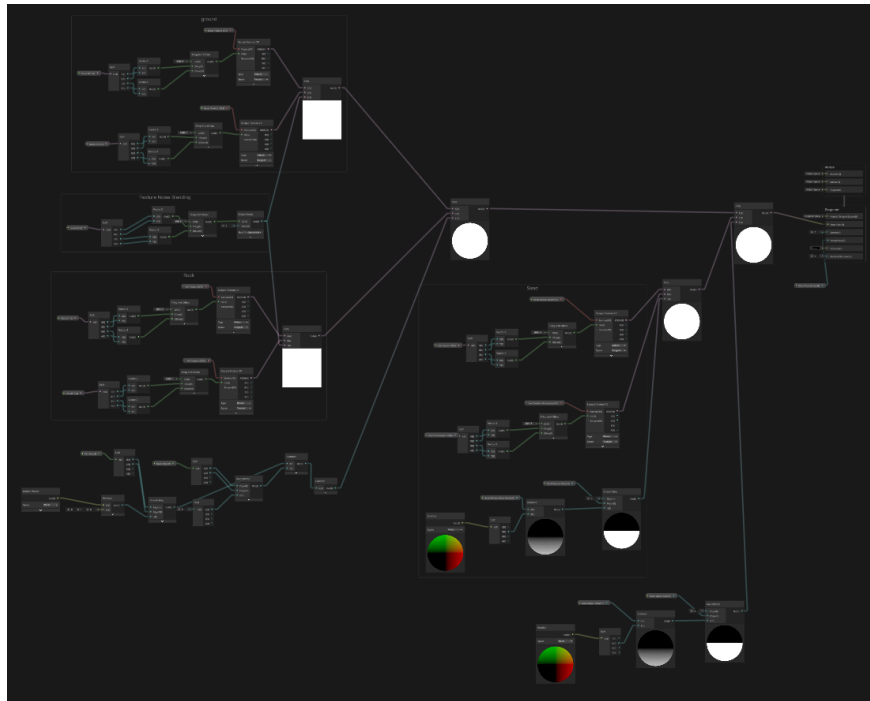


Island Radius

The island Radius controls the distance between the top and the bottom of the sine wave.

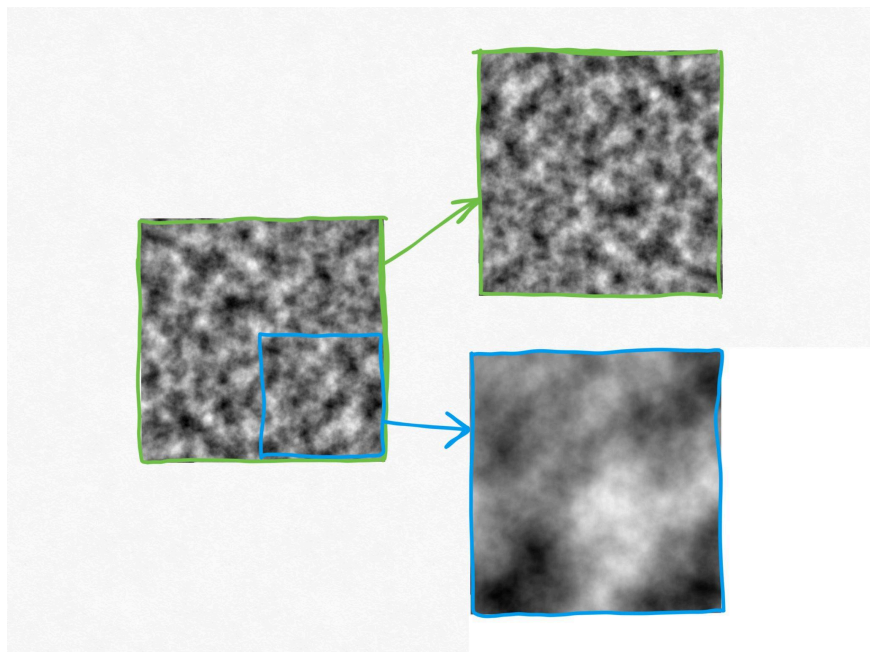


Terrain Material

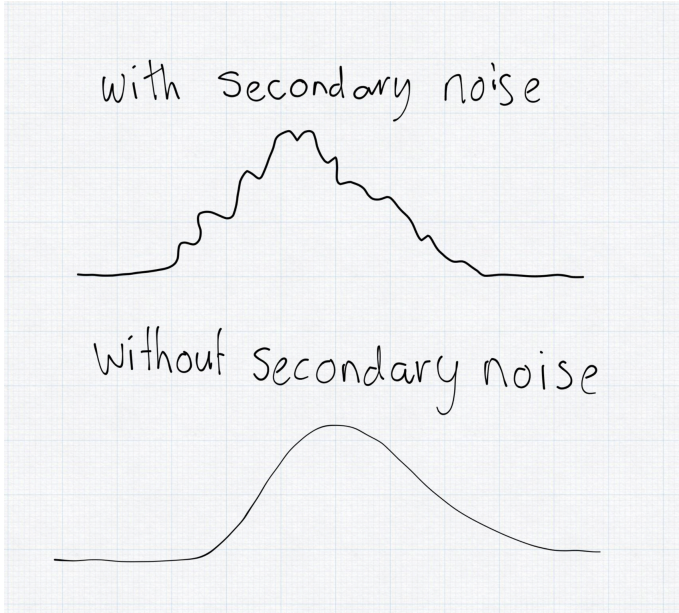


The terrain material is created with this shader graph that I made. If users wanted too they would be able to change out the textures used in order to generate a different effect, eg a snowy island.

Noise Intensity



The noise intensity parameter is actually controlling the scale of the Perlin noise. As you can see in the diagram above the blue zoomed in noise will be less intense, and the green zoomed out noise will be more intense.

Seed	<p>The seed offsets the location at which the perlin noise is generated. By changing the seed the terrain generation will result in a completely different result.</p> <p>I have also included a randomize button in the inspector. This changes the seed to a random number between 0 and 10,000.</p>
Height	<p>The height value controls the height of the terrain.</p>
Secondary Noise	<pre data-bbox="560 506 1414 800"> if (useSecondaryNoise) { islandMultiplier += Mathf.PerlinNoise(islandMultiplier += Mathf.PerlinNoise(islandMultiplier += Mathf.PerlinNoise(} </pre> <p>The secondary noise parameter is a bool that when enabled, adds three new levels of perlin noise to the terrain. This results in a more bumpy and noisy terrain.</p> 

Resources

During the process of learning how to create a procedurally generated landmass in unity I used many different resources and followed many different tutorials. Some of the tutorials I followed can be seen in the original github repo. (I decided to start a new repo due to the fact that it was getting annoying attempting to name my scripts due to duplicate classes already existing).

Here are some of the resources that I used during the process of learning procedural generation.

[Complete Guide To Unity Procedural Generation - GameDev Academy](#)

[Start NOW!!! Beginners guide to Procedural Generation!](#)

[Procedural Terrain Generator! UNITY](#)

[The Theory of Noise: An Overview of Perlin Noise](#)

[how to make a procedural grid world in under 2 minutes in unity \(part 1\)](#) (Whole series)

[GENERATING TERRAIN in Unity - Procedural Generation Tutorial](#)

<https://docs.unity3d.com/ScriptReference/Mesh.html>

[Procedural Landmass Generation \(E01: Introduction\)](#) (Whole series)

[How Minecraft ACTUALLY Works](#) 

[How does procedural generation work? | Bitwise](#)

[Why I'm Using Wave Function Collapse for Procedural Terrain | Unity Devlog](#)

[How to create and modify a plane mesh in Unity \(Procedural mesh generation tutorial\)](#)

Reflections

Mid-Point Reflection

Proficiency:

Throughout this project I was able to develop my technical skills and knowledge simply by watching videos and reading papers on procedural generation, I probably watched at least 40 videos on the topic of procedural generation, and I even branched out into other PCG fields such as procedural animation, and procedural storytelling, in order to ensure that I get a good grasp on procedural generation as a whole. For example, a paper I read on procedural dungeon generation was able to provide a really good definition that clarified exactly what procedural generation actually is: "Procedural content generation (PCG) refers to the algorithmic creation of content. It allows content to be generated automatically, and can therefore greatly reduce the increasing workload of artists."(**van der Linden et al., 2014**). The more I researched procedural generation the more I was able to understand the concepts, however I also found that the task of procedurally generating an island with trees and other things was simply too complicated for me to achieve with no prior learning and within a short period of time.

Process:

During this project the process that I followed was pretty much the same the whole time, I simply researched and followed tutorials from youtube, I continued to do this quite a few times until I finally felt confident enough to create my own procedural mesh. This definitely took a few attempts, and I had to rewrite and reorganize my code quite a few times, however I feel like in the end I was able to create some pretty clean code and a very nice and easy to use inspector interface. After I had created a procedurally generated island mesh that I was finally happy with, I faced quite a few problems with creating a shader for the island. Originally I wanted to just download a shader from somewhere, however I couldn't find any free shaders for island meshes. So in the end I had to create my own island shader, this process ended up being quite time consuming because I found it very difficult to find good tutorials on the shader graph in unity.

Person:

This project was a group project however it certainly didn't feel like one. Throughout the whole project myself and one other student seemed to be the only ones who contributed any type of

work towards this project. This made it quite difficult for me to practice my collaboration skills. We were however, able to have some good group discussions about how we would like the game to turn out, this was great and it was fun to discuss how we could put the game together, however sadly I never saw any of the work we allocated to each other completed, apart from mine and Mitchells. Despite the lack of participation I feel that Mitchell and I were still able to give good feedback to each other, for example I was able to help with things such as assessing how easy it was to tell what time it was in Mitchell's Day/Night Cycle.

Project-Completion Reflection

Appraisal:

Overall I am pretty pleased with how my project turned out, however I would have loved to add a few more ease of use things into the package such as a method of resizing the water asset for the different size islands. And I would have loved to figure out some sort of procedural tree placing method using the normals of the mesh, however I simply did not have enough time to do this. Overall I do not think that the project matched my initial expectations, however after actually researching this topic I feel that I actually did quite well, and I am quite proud of my little islands.

Challenges:

During this project I had to overcome so many obstacles and challenges, however the main challenge that I faced was figuring out how to apply a fall off to the generation of my mesh. When attempting to overcome this challenge I looked at and tried many different methods. One method that yielded really good results was Sebastian Lague's tutorial on creating a fall off map. I was able to get this method to work really well when I followed his tutorial, however I tried for so long and I simply couldn't get the fall off map to work with my own project. I believe this was due to the way that I was generating separate chunks for my terrain. In the end I came across a method of using sine waves to control the shape of a mesh and after a good amount of tweaking I was able to get this method working for my island. Through this I learnt a ton about the importance of researching and finding out what other people do in order to gain information on how to tackle personal coding problems.

Future goals:

In future projects I will continue to improve my skills by researching all areas of game development, Unity, and C#. It is my goal to eventually have an extremely in-depth understanding of each of these topics, and I believe that I am on the right track to achieve this goal. In future projects something that I will do differently is that I will attempt to be more careful about the group that I find myself in, however I am aware that this is often unavoidable, so I will attempt to figure out new ways to try and overcome the challenges that arise from lack of group participation. Something that I am keen to repeat in all future projects is to set myself tasks that I have no idea about because I really enjoyed the process of learning and unpacking procedural generation, and I believe that I have greatly improved my coding skills and knowledge due to the fact that I set a large challenge for myself.

References

Lague, S. (2016). Procedural Landmass Generation (E02: Noise Map) [YouTube Video].

In *YouTube*.

https://www.youtube.com/watch?v=WP-Bm65Q-1Y&list=PLFt_AvWsXl0eBW2EiBtl_sxmDtSgZBxB3&index=2

van der Linden, R., Lopes, R., & Bidarra, R. (2014). Procedural Generation of Dungeons.

IEEE Transactions on Computational Intelligence and AI in Games, 6(1), 78–89.

<https://doi.org/10.1109/tciaig.2013.2290371>