Computer Science Department

Final Project: Perlin Noise in Terrain & Animation

Course Instructor: Kevin Smith

Martin Bojinov

Introduction

Noise functions are a popular tool used to generate “coherent” randomness. Perlin noise, developed by Ken Perlin in 1983, is the most popular of these noise functions [2][3].

Noise functions, such as Perlin noise, can be used to randomly generate terrain. This is very handy in procedurally generating realistic looking maps in computer graphics. This is particularly handy in media such as video games. Hand crafting a map takes a lot of time so many games resort to procedural generation to speed up the process[4].

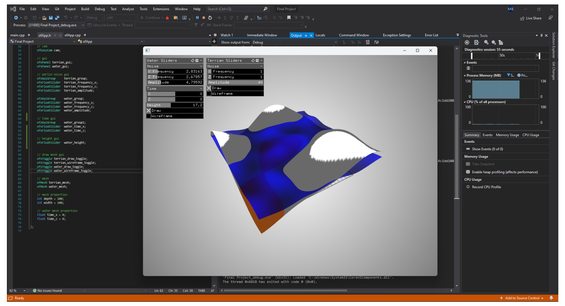
The procedural generation of terrain can be done beforehand and imported into the media it is used. This project aims to create a system where terrain is created using Perlin noise and updated live. Specifically in the use of animating water to create waves and updating textures based on noise values.

These features can be used live in video games to create modifiable terrain or create convincing terrain with minimal effort. The main goal of this project is to be able to animate a portion of a world map, containing variable islands / continents.

Methodology

Open Frameworks

Open Frameworks, an open-source graphics toolkit, was used to develop this application from scratch. It combines many commonly used graphics and math libraries for ease of use.

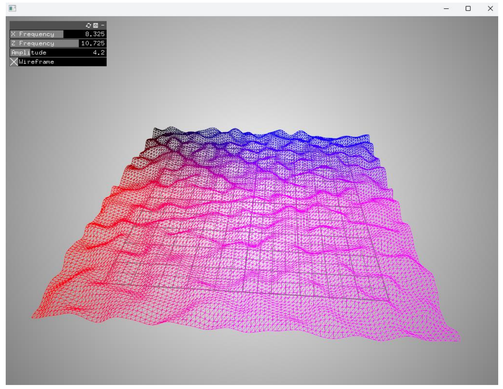


Open Frameworks has built in support with Visual Studio and C++. This fosters a simple, clean environment to develop the project. Some special features include a built-in compiler, debugger, and autofill text for library functions.

Methodology

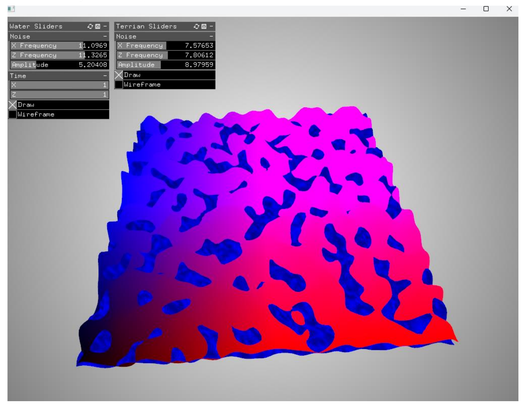
Procedurally Generating Terrain (and Water)

To procedurally generate terrain using noise, we must first create a canvas. We start by implementing a simple 2D mesh in open frameworks and inputting (x, z) values into the noise function. The output is then fed into the y component of the mesh.



From there we hook up several GUI sliders to control the parameters of the noise function. This allows us to update our terrain live within our simulation.

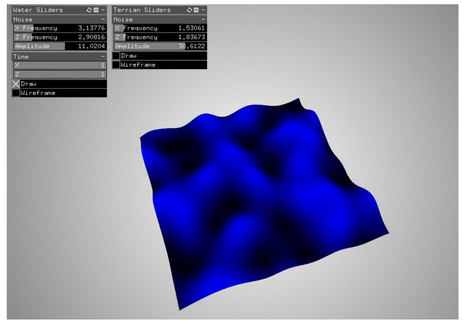
Now that we have terrain generation done, we can look to create an identical system to generate water. We create a second mesh layer for water generation and define many similar parameters as the terrain mesh. The two meshes initially overlap in an awkward manner.



Animation & Live Terrain Coloring

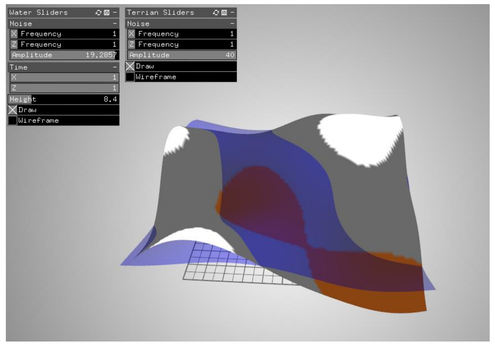
Now that our two meshes are setup, we fine tune their parameters. From there we implement animation and terrain coloring.

Terrain coloring is the first to be implemented by modifying the color value of each vertex on the mesh based on its y value (which we got as an output from the noise function). This allows us to dynamically color the mesh based on the parameters passed to the noise function.



Animation is then set up by adding a “time” value to the x and z components passed into the noise function. By making time positive or negative, we can change the direction that the ocean waves move.

Combining all these features into the two meshes we get a procedurally generated terrain that can be updated live with parameters. The water is animated live, and the textures of the terrain update based on noise function parameters.



Analysis and Results

The main parameters for the noise function of each mesh are amplitude, frequency(x), and frequency(z). This affects how high the terrain or water is, and how many waves / mounds there are in the x and z component, respectively.

The time parameters of the water mesh affect how fast the waves animate and in which direction. The wave’s height can also be adjusted to simulate how tides raise and lower during the day.

Conclusion / Future Work

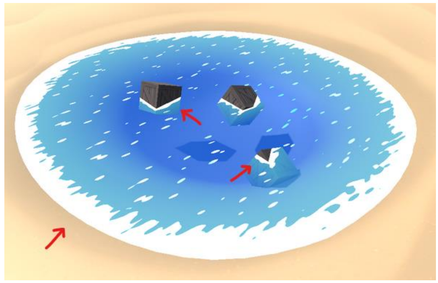
Conclusion

This live generate terrain can be used for quick backdrops in CG media or as terrain in video games. It offers quick, live, world generation that looks live instead of being completely static.

Future Work

In the future I would like to use a third layer of Perlin noise. This will be used to create “biomes” that designate areas on the world map. This will allow for different regions to be colored in different manners. This will also allow for the parameters to be modified by the biome. For example, a desert biome would have a low amplitude to create sand mounds / dunes. The terrain would be colored sandy-yellow in this region.

Further work would be to make the animated water more realistic. In his blog, Daniel Pokladek outlined a method that Perlin noise can be used to create foam in water [1].



Key References

[1] D. Pokladek, “Toon water shader: Perlin noise, Animation & Foam,” DANIEL POKLADEK BLOG, https://danielpokladek.wordpress.com/2020/03/24/toon-water-shader-perlin-noise-animation-foam/ (accessed May 15, 2024).

[2] K. Perlin, “An image synthesizer,” *ACM SIGGRAPH Computer Graphics*, vol. 19, no. 3, pp. 287–296, Jul. 1985. doi:10.1145/325165.325247

[3] K. Perlin, “Improving Noise,” *New York University*, Jul. 2002. https://rmarcus.info/blog/assets/perlin/improved_perlin.pdf

[4] Smith, Kevin. “Lecture 8 - Ray Marching – SDF’s and Noise Functions” Geometric Modeling, 9 April 2024, San Jose State University. Lecture.

Acknowledgements

The project could not be accomplished without the support and guidance of our professor, Kevin Smith.

I would also like the acknowledge my dad and older brother for getting me into computers and gaming. This spawned my love and fascination for computer graphics.