# Biome Generation using Perlin Noise and Cellular Automata

*Author: Duncan Grubbs*

## Introduction

For my final project, I implemented a biome and terrain generator that uses Perlin noise and cellular automata. I was immediately interested in Perlin noise after implementing the lab that focused on using noise for height maps, and I wanted to see if I could do more with it. While there wasn't a cellular automata lab in which we implemented the full algorithm, I was fascinated by the power behind such a simple model. I wanted to combine these two interests and I landed on height map generation using Perlin noise followed by using cellular automata to "grow" biomes across the height map in a "natural way". I wanted to create semi-realistic biome growth by forming rivers out of the snow on the mountains, and then having the rivers, in turn, create lakes. Wherever there was sufficient water, I wanted plains and forests to grow depending on the elevation dictated by the height map. I planned on first generating the height map, and then slowly adding complexity to the algorithm through cellular automata update rules. I knew that the hardest part was going to be creating the update rules for the automata, so I first wanted to build the infrastructure to make that process of experimenting as easy as possible.

## The Algorithm

My implementation works in the following way. First, it creates a 500x500 2D array. Next, it populates the array with automata cells that just contain an elevation in their state. Then, a number (by default 1500) of random pixels are assigned the mountain (snow) biome based purely on their elevation (i.e. if their elevation is greater than 180). At this point, there is enough information in the map to start updating cells based on the automata rules. The biome (cell) that has the most complex update rules and starts growing first is *RIVER*. In one update iteration, each cell is updated based on the states of the cells in its Moore neighborhood. The most common update rule checks if there is a neighboring cell with a biome defined and the elevation. If the biome is compatible, the cell takes on that biome. When drawn, the color of each pixel on the map visualizes its cell's biome which could be *LAKE*, *RIVER*, *MOUNTAIN*, *FOREST*, or *PLAIN*. *MOUNTAIN* (snow) is white, *FOREST* is a brown, *PLAIN* is green, and *RIVER* and *LAKE* are blue. The colors are also adjusted according to the cell's elevation so that lower elevations are darkened, and higher elevations are brightened.

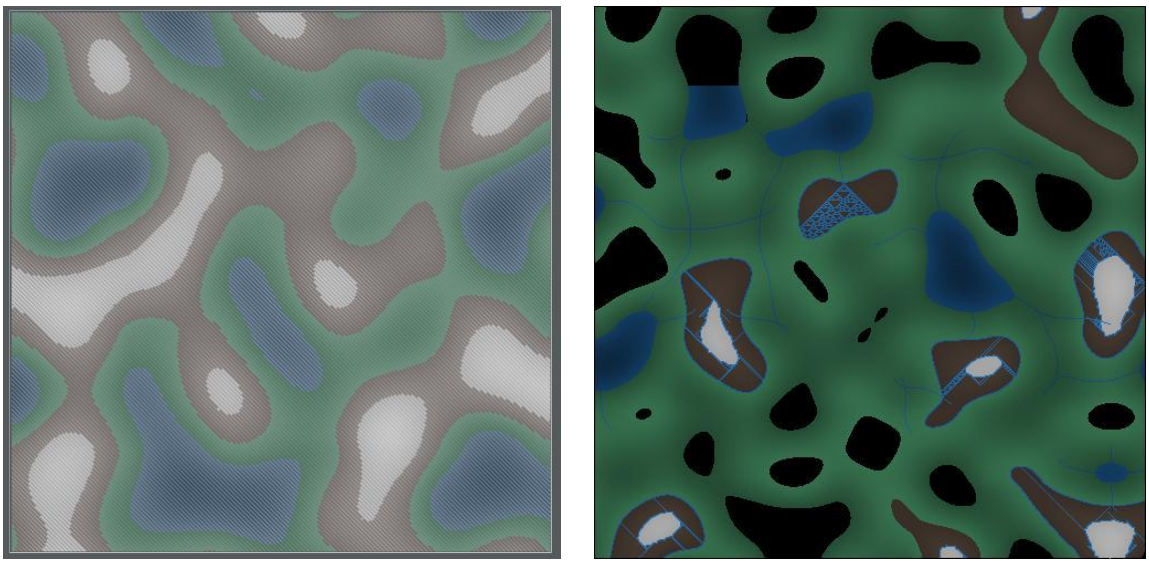
## Analysis

While I achieved many of my goals for the project, there are still many things I would like to fix and improve. On a basic level, the generated map looks good and there is an interesting progression of biome growth over time that can be visualized. That being said, there are some key areas of potential improvement.

1. Perlin noise generates fairly uniform bumpiness, which is not entirely accurate to the real world, where there might be extended flat or mountainous areas. Using a combination of noise functions, or even predefined rules about elevation ranges for areas of the map could be a future improvement.
2. Rivers do not *always* flow downhill. Given the complexity of each cell's state, it is very hard to write update rules that allow for super realistic water flow. However, it is easy to add more rules, so in the future, it would be very possible to improve the realism of water flow.
3. There aren't that many biomes. In the future, I could add biomes such as desert or ocean, which could have interesting impacts on their surrounding biomes.

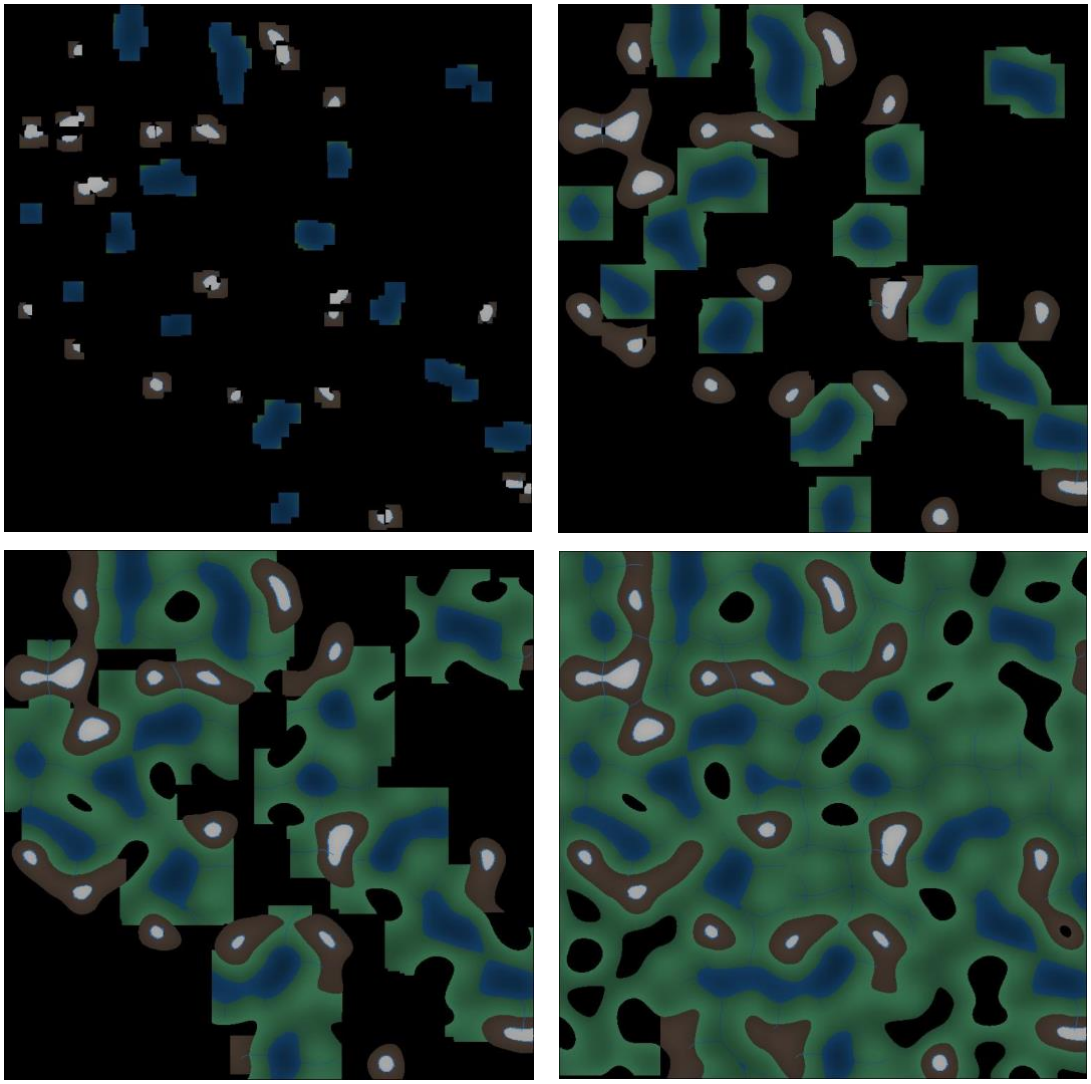
The biggest challenge I faced stemmed from the complexity of individual cell states. This made the process of creating update rules incredibly difficult. The number of possible combinations of neighborhood cell states grows exponentially with the number of possible cell states. Because of this, I could only account for a tiny fraction of possible cell state combinations. When testing my update rules, some of them would never apply and I couldn't even see their impact, while others would cause a biome to take over the entire map. This unpredictable behavior of cellular automata is a well-known difficulty of working with them, and trial and error is still one of the best approaches used even at high levels.

## Results



***Left**: Initial biome generation just based on elevation **Right**: Classic automata pattern emerging during testing*

***Below:*** *4 stages of biome growth*



***Below:*** *Larger Demo (1000px by 1000px for 2500 iterations and 3000 seed cells)*