Reinforcement Learning

Assignment 1: Procedural Content Generation

M.F.A. Aarnoutse
s2682796

March 11, 2021

1 Introduction

In this document I describe my approach to assignment 1 for the course *Modern Game AI Algorithms*. In this assignment we generate 2D coloured world maps. In Figure 1 is a map shown by Weeden (2015). This example shows three biomes; blue water, green grass, and mountains in beige. The idea is to create something similar with two unique additions. I have chosen to implement beaches and shade the mountains. As an extra feature I will simulate erosion on the heightmap.

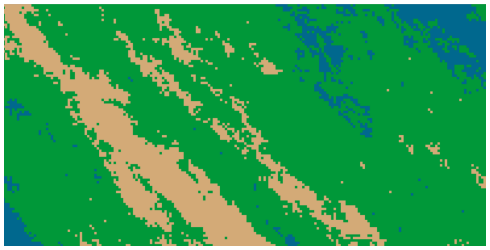
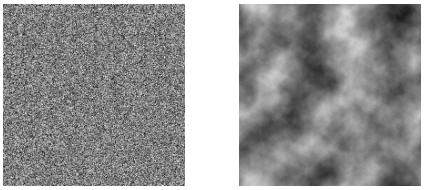


Figure 1: Example from Weeden (2015)

2 Background

Before I get into the results of map generation we take a look at how randomness is implemented within procedural generation.

These maps are created using noise and assigning a colour to a certain threshold. Specifically Perlin noise can be used as a baseline as this is a type of gradient noise. The difference between white noise and Perlin noise is shown in Figure 2. This makes it clear that the Perlin noise is continuous, whereas the white noise is completely random.



(a) White noise (b) Perlin noise

Figure 2: Difference between white and Perlin noise

The implementation of Perlin noise is easier explained in 1 dimension. For every integer $x = 0, 1, 2, ..., n$ we choose a slope with $y = -1 < y < 1$. Between these points we can apply interpolation and we would get randomised terrain where every hill is a parabola (Figure 3). This would

not look much like the terrain we set out to create. What we could do is generate points in between the existing points such as $x = 0, \frac{1}{2}, 1$. By increasing the number of points between 0 and 1 the terrain has a greater resolution and more detail. These are the number of octaves the generator has.

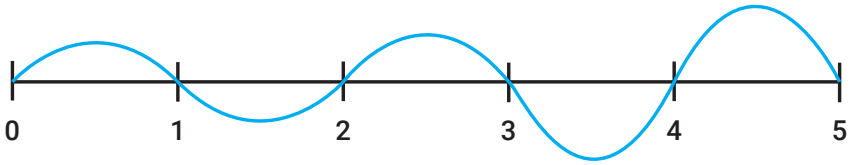


Figure 3: Perlin noise in 1 dimension

Converting this principle to 2D works in the same way as before. Instead of a line we have got a grid with vectors. These vectors are the 'slope' from the 1-dimensional example. Taking the dot product from these vectors results in a distance measure from each point (Eevee, 2016). This is visualised in Figure 4. Applying interpolation between the gradients results in a Perlin noise map such as Figure 2b.

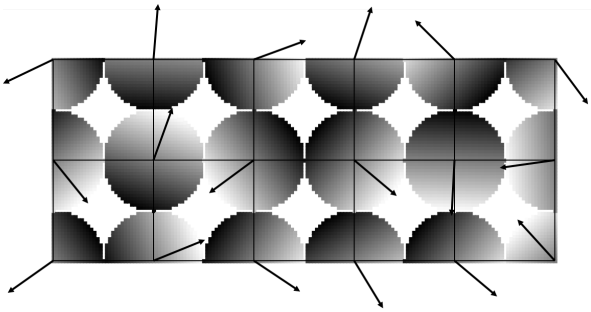


Figure 4: Dot product of vectors result in gradients (Eevee, 2016)

3 Results

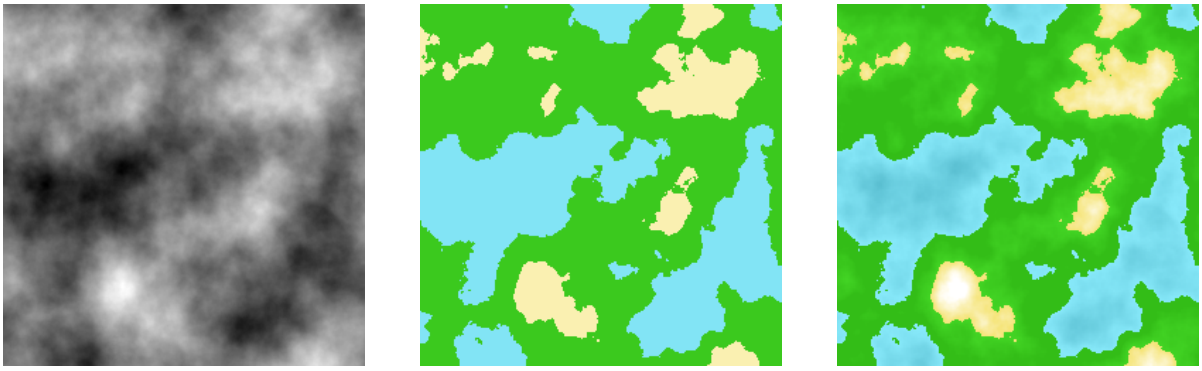
In this section I show examples of the created generator using the two methods from above. As the assignment states I should introduce two new unique features to the already existing green, blue and beige biomes. This choice has been made to add beaches around water bodies and add more believable mountains with a heightmap.

3.1 Generating water, grass and mountains

Using the Perlin noise map we can assign colors to different height thresholds, e.g. everything between 0–0.4 is assigned to the water biome. This results in Figure 5b.

Adding shades can be done using the initial Perlin noise map that seperates the biomes. For each biome I assign two colours; start and end, where end is a darker shade of the starting colour. Using linear interpolation of the two colours I can assign a shade when the height is towards the lower end of the depth interval. The result is shown in Figure 5c.

A simple addition to the previous map is more layers. I added snow on top of the mountains and created to additional grass biomes; lowlands and the highlands. A very small threshold between water and the lowlands is assigned to 'waves'. This gives me the opportunity to assign a white to light blue colour to indicate a very shallow water body. This makes it almost seem like it is possible to walk through the water in the top left corner of Figure 6.



(a) Heightmap (b) Default (c) Shaded

Figure 5: Water, grass, and mountains

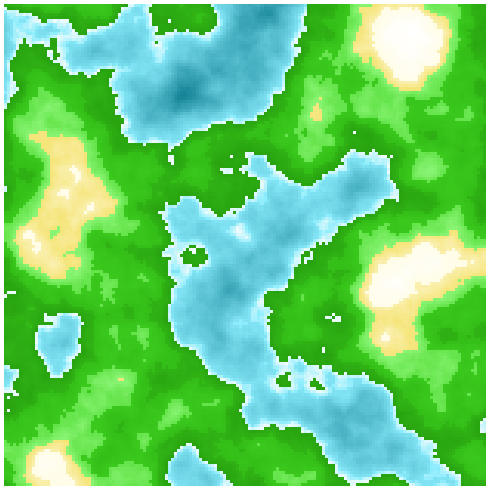
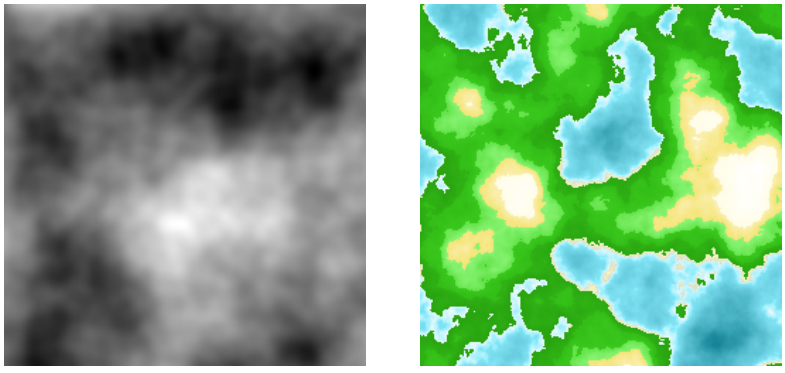


Figure 6: Adding more layers

3.2 Beaches

With the waves in place the only thing we are missing in the map are beaches. I could have easily done the same as the waves and assign an interval between the water and land to the gold-yellow colour for beaches. This is not realistic as not every coastline contains a beach. The way I solved this is by generating another Perlin noise map.



(a) Heightmap (b) Map

Figure 7: Beaches

As the gradients in Perlin noise map are uniformly distributed we can use this as the probability for the beaches. If the x, y in the noise map is > 0.5 we try to draw a beach at that location. This is done by replacing a part of the land and water to make it seamless. The result is shown in Figure 7.

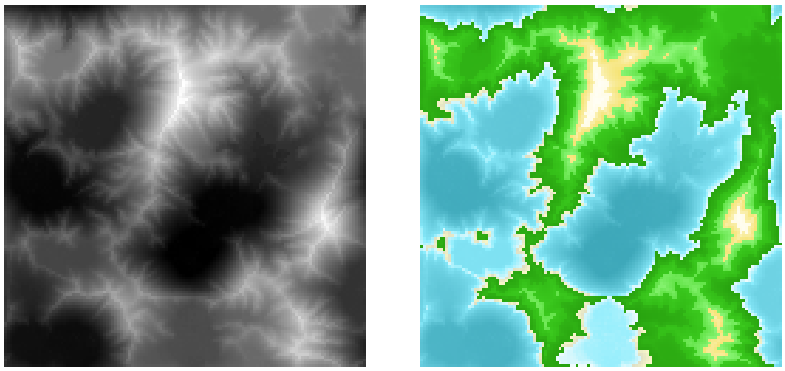
3.3 Improving heightmap

At last I set out to create a more realistic looking coastline and mountains. In real life the mountains and coastlines are not this smooth. This is due a geological process called erosion where water and wind drag down soil to different places. However, we are able to simulate this process by keeping track of three properties of the terrain; terrain height, water level, sediment level. The idea is that water dissolves terrain into sediment, which is transported downhill and deposited (Andrino, 2020).

Andrino (2020) describes the simulation in six steps:

1. Incrementing the water level by simulating precipitation
2. Computing the slope of the terrain to determine how the water flows
3. Determine the sediment capacity
4. Dispose sediment when over capacity, otherwise erode terrain
5. Move water and sediment downhill
6. Evaporate a fraction of the water

The repository of Andrino has a sample available that has been implemented in the project. After simulating the erosion 179 times we get the result in Figure 8. This makes the mountains more realistic and we can see that the bottom of the ocean is a lot smoother whereas it previously still looked cloudy from the Perlin noise.



(a) Heightmap (b) Map

Figure 8: Erosion

4 Conclusion

There is still a lot that can be added to the terrain generation, such as rivers, towns, and forests. Although this seemed like a good place to start as any. With these foundations we can see that the world is starting to take shape. While I implemented different terrain types this would only be one or two biomes in a game like Minecraft. There is still a lot of work to be done to create a diverse world.

References

Andrino, D. (2020, January 29). Three ways of generating terrain with erosion features. Retrieved 2021-03-01, from https://github.com/dandrino/terrain-erosion-3-ways

Eevee. (2016, May 29). Perlin noise. Retrieved 2021-03-01, from https://eev.ee/blog/2016/05/29/perlin-noise/

Weeden, K. (2015, July 26). Procedural generation in python. Retrieved 2021-03-01, from https://medium.com/inspired-to-program-%E3%85%82-%D9%88-%CC%91%CC%91/procedural-generation-in-python-7b75127b2f74