Perlin Noise Map Generation

For the Assignment 2 project, I decided to focus my efforts on procedurally generated content that has consistency upon generation. This wold primarily rely on computer based pseudorandom number generation in order to generate Perlin noise maps in which data could be extracted. The applications of using procedurally generated content in games allows for a longer effective shelf life of a product, wherein each game may act as a relatively new experience while having consistency within the gameplay loop and mechanics. However, the usual drawback for using procedural content is that there is a lower overall quality for level design, wherein without specific precautions, it is possible to create unplayable maps or have all terrain elements be clustered in one specific area creating a blank space in design that would be bothersome to players.

The main programming technique used in this project is the use of a Perlin noise generator for both the water effects by the islands generated and the island generation in itself. Normally, Perlin noise would generate a random heightmap of realistic looking terrain without any particular features being above any of the others, however, by adding a base level of a paraboloid, it is possible to force the generation on top of a dome, rather than a flat surface and through manipulation of constants such as the lacunarity and persistence of the Perlin noise it is possible to ensure that the heightmap would almost always create an island with at least one mountain peak at the maxima of the heightmap. However, the current approach will never produce any form of cavern or cave system without further work, though it is possible for other features such as small islands, bays, and mountain ranges to be prominent.

Following this I have also added a system that reads the generated heightmap and creates a 2d array of the maximum gradients at each point on the map to be stored in the terrain data. This could then be used to generate roads that follow the path of the lowest gradient as well as the shortest route to create a more natural looking environment, this would be possible with the application of an A\* pathfinding script that reads steeper gradients as inaccessible points on the 2D array and tracking the sum of gradients traversed to determine which current traversal is closest to the destination.

Beyond this I have also created a modifiable shader that reads the height of each point on the map in order to determine an appropriate colour scheme, blending the snow of the mountain peak into the rock layer, and slowly blending into grass before suddenly becoming the shoreline beach. The shader itself could be used to apply a variety of textures to the terrain, however I preferred the simpler aesthetic of plain colours. The reading of the terrain used in the shader could also be extended to the placement of foliage, such that at low level of grasslands it is possible for forests to spawn, while the more mountainous regions will consist of rock formations.

For the project I have submitted I would aim to implement this terrain generation into a “Battle Royale” game akin to Player Unknown Battlegrounds and Fortnite Battle Royale, the benefit being that the map may contain the same base elements in each game session by adding buildings to the map upon creation, but the player must consistently adapt to a new map layout each game, preventing the ability to have a set metagame in effect.