C S 378H Final Project: Flying Virtual Terrain

Deepanshi Sharma

For my final project, I created an infinite virtual terrain on a 2D canvas using ThreeJS and WebGL. The virtual landscape is navigated through the aerial perspective of a flamingo in flight, which can be guided by the user. My goal in terrain generation was to allow the user to freely navigate a never-ending landscape with many customizable features. Some user-controlled features include: camera viewpoint, flight velocity, terrain noise, and map height.

My implementation of the virtual world was highly stylized in a colorful and low-polygonal appearance. This choice of style not only serves an artistic purpose but also saves computational cost and processing power. As a result, my main focus surrounded procedural terrain generation and lacked emphasis on shading/lighting. All objects in the scene are shaded using a flat shading model with some added gradient functionality. In order to achieve some feel of texture, I added a randomly generated offset to each RGB channel, which is also user-customizable. In terms of lighting, the scene is illuminated by two directional lights encapsulated in a sphere mesh "aura" to emulate the Sun, as well as some ambient light to soften the effect.

The objects in the scene are digested in a gltf format through Three.js's GLTFLoader class, which converts a glb models to gltf files. Predefined animation key frames are processed using an AnimationMixer, with frame durations calculated using timestamps. Flight movement is controlled using the W-A-S-D keys, and the arrow keys are used to toggle four different camera angles. Two event listeners keep track of when a key is pressed and when it is released respectively. The wing speed of the bird is controlled by its velocity – faster wing speeds

correlate to higher velocities. Animation key frames are encoded in a two-dimensional array, where each row contains a non-zero entry for the frame to be displayed and each column maps to an individual animation keyframe. I also implemented rotation about the XYZ coordinate system at angles between -0.8 and 0.8 radians. Flamingo object position was also tracked in the parent structure to update the terrain according to the user's current position.

The terrain is procedurally created using a height map that is generated according to a Perlin noise distribution. This height map is then structured according to user-customizable parameters and applied to the vertices of a plane object. Perlin noise is simplified using Simplex noise, which takes a random seed as an input. Simplex noise operates on fewer directional artifacts, which greatly reduces computational complexity and improves performance. Infinite generation is achieved by allocating blocks of terrain as needed according to user position. As the user navigates the scene, blocks are allocated, destroyed, and shifted according to a width-mapped pixel offset. I addressed glitches in object movement due to shifting terrain by updating the model with increased frequency to account for delays.

I was successful in my implementation of a procedurally generated virtual terrain in terms of artistic quality and good performance. A potential next step for this project would be to build a virtual reality application that is immersive for the user. More advanced rendering and shading techniques can also be applied, but those techniques come at the cost of greatly increased computational complexity. Ultimately the future of virtual world-building relies on advancements in real-time rendering performance.