Tyler Petrochko

CPSC 478 Final Project

**Deliverables**

For my final project, I created a space-racing game in which the player tries to stay alive as long as possible while dodging obstacles. The player flies above an asteroid-like planetary body, which includes a procedurally generated terrain. Movement is restricted two axes (X and Y), and movement is clamped within a certain range on the Y-axis.

To run, run a local webserver

python -m SimpleHTTPServer 8080

And navigate to

http://localhost:8080/star\_marauder/main.html

Gameplay is simple – use the WASD keys to maneuver the ship up and down, and avoid hitting obstacles. As game continues, the score accumulates at the bottom of the screen. I tested my assignment using MacOS Sierra 10.12.6 and Google Chrome.

**Challenging Aspects**

My game includes a few challenging components, namely procedural terrain generation, collision detection, ship animations, and a fog-based shader. Procedural terrain generation uses an optimized simplex-noise library (see Sourced material) and Bezier spline patches to generate a 2D ground surface. The most challenging component of this, however, was re-using procedurally generated “tiles” so that as the ship “moves,” spline patches are moved rather than deleted and re-generated. Because of this, a large number of tiles can be used without severely decreasing frames-per-second.

Collision detection is fairly straightforward – each object has a simple sphere collider proportional to its scaling factor. Spline patches also have a collision function, which is triggered when the spaceship falls below the average of the lowest and highest vertex. This severely lowers computation time as well – preventing expensive collision detection.

As the ship moves up and down, the ship’s pitch and roll are linearly interpolated based on the ship’s linear velocity. This gives a smooth animation as the ship maneuvers up and down.

Lastly, a simple Lambertian shader with linear fog falloff is used to make far-off objects obscured. This reduces the harsh effect of the procedural terrain generation giving a visibly jagged “horizon.” If I had more time, I’d like to blend the fog into the skybox in a less harsh manner, or procedurally generate fog-clouds as well.

**Sourced material**

Space Skyboxes from <https://opengameart.org/content/space-skyboxes-0>

Simplex noise generation from <https://github.com/jwagner/simplex-noise.js/>

Base code taken from chapter 7 of <http://www.envymycarbook.com/>