CSCI-331 **Spring 2023**

Project 3: Flight Simulator

Directions

For this homework assignment we will be making a basic flight simulator over a finite, random, terrain. We will be flying something more like a quadcopter drone than an airplane as the camera won't fly forward on its own. This project will make use of meshes, perspective projections, lighting, and collision detection.

For this project you are given empty HTML and JS files. You must write these from scratch, copying the necessary parts from other examples as necessary. However, any unnecessary parts should not be included and will cause deductions. You are also given 2 additional JS files that include code for various algorithms (including some functions done in class). Feel free to import this code (not copy-paste it but include the script).

The terrain is generated using the diamond-square algorithm which is a recursive algorithm that is already implemented for you in terrain-generation.js. However, you must read the documentation of the provided functions to be able to use it properly. The first function generates the data and the second function turns it into a set of vertices and indices to be used on the GPU. The terrain should be drawn with lighting so the contours and shapes of the terrain can be easily seen (however do not worry about shadows). Exact colors and shading are up to you, but it should look somewhat like a realistic landscape (although that landscape could be on Earth, under the sea, or on Mars).

The flyer should start in the middle of the random terrain slightly above the surface. Note that the random terrain is generated in such a way that the middle of it may be negative or positive thus you can't just start at a height of 0, you have to find the height of the middle of the terrain and place the flyer slightly above that. The controls for the flyer are based on the aircraft principal axes as follows:

- Up/Down arrow: moves forward/backward based on the direction it is facing
- Left/Right arrow: rotates to the left/right (yaw)
- W/S: rotates down/up (pitch)
- A/D: rotates clockwise/counter-clockwise (roll)

The flyer is not allowed to pass through the terrain but nothing bad happens when it "hits" the terrain. Make sure that it cannot become stuck inside a piece of the terrain either. To assist you with collision detection you are given a function to detect if a line segment and a triangle intersect in 3D. Your collision detection should be efficient (i.e. do not examine every triangle to see if you will intersect it but instead just the ones that could possibly be intersected). Additionally, the flyer is not allowed to leave the area with terrain below it.

The view from the flyer is to use a perspective such that no part of the terrain is ever clipped if the flyer is above the defined region and not absurdly far off the ground. The canvas should be full-screen and maintain the proper aspect ratio regardless of the dimensions of the window. Pick values that make it look "real" without a significant fish-eye effect.

Grading

For full credit, be sure to follow the directions above. There are 100 pts and they are broken down as follows:

- 15 pts for code formatting (although really bad code can go negative here)
- 10 pts for rendering the random terrain mesh
- 10 pts for providing appropriate colors and lighting to the terrain
- 15 pts for using an appropriate perspective view of the world
- 20 pts for appropriate movement of the flyer
- 5 pts for not allowing the flyer to pass beyond the edges of the terrain
- 25 pts for not allowing the flyer to pass through the terrain
- +5 EC: terrain colors depend on terrain heights and/or the slope at the current location
- +5 EC: lighting depends on actual current time (e.g. position and intensity of light changes with time)

Guidance

Recommended order of working on this project is as follows so that you don't get overwhelmed:

Mesh

- Draw the terrain mesh
  - The code gives you a 2D array of heights. You need to make this into a set of triangles. You will need to draw this out on paper to be able to figure out how to turn a grid into a set of triangles.
- Add in lighting (standard)
- Adjust colors so that it looks more like terrain (colors should be dependent on terrain height, likely easiest to do this within the fragment shader itself)

Flyer

- Switch to using a perspective view that works appropriately (standard)
- Remove the mouse handlers (if you have them) and add the keyboard controls of the flyer
  - Note: this will take ***significant*** time to get correct – the order of matrices is a challenge
- Start the flyer in the middle and slightly above the terrain making sure it moves correctly
- Add in collision detection so the flyer cannot pass through the terrain
  - Note: this will take ***significant*** time to get correct – ***like an entire week!***
  - Get the current position (transformed) and direction of travel (transformed) as a vector with the length of the movement being made
  - Go through all triangles with a similar loop as was used in `calc_normals()`
  - For each triangle, check if the line segment defined by the current position and vector of travel (from the first bullet point) intersect it
  - If you find an intersection, shorten your vector of travel to meet up with the intersection point (actually, just shy of that to avoid getting stuck in walls) and keep on searching (there can be more then one collision detected)
  - At the end you have the appropriate movement to be made that won't go through anything
  - One that is working, speed everything up by making the loop that goes through all triangles only iterate through triangles currently nearby the flyer
- Check your code and make sure it follows all of the directions