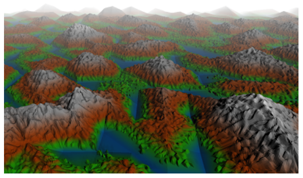
# MP2: Flight Simulator

## Due: October 19 at 11:55pm



Write a simple flight simulator using WebGL to handle the display. The airplane should automatically move forward at a fixed speed.
The user will control the bank and tilt of the airplane through the arrow keys.

- Pressing the left (right) arrow key will make the plane roll to its left (right).
- Pressing the up (down) arrow key will cause the airplane to pitch up (down).

You will need to implement the following

1. A quaternion based viewing system.
2. A terrain rendering algorithm so that you have something to fly over. At a minimum, implement the diamond-square algorithm. You can implement something more complex if you wish.
3. A vertex shader that implements either Phong or Blinn-Phong shading. You can use demo code from the course webpage.
   You can position your light source anywhere in the scene as long as the rendered images are well-lit.
4. You should shade the terrain using a colormap related to the height of the vertex. For example...your colors could go from blue to green to brown to white as in the image above. But please...be creative and don't feel you have to mimic the colormap used here...a simpler one from green to brown is fine, for example. Also. the above image implements fog which you don't need to do....so don't try to match it exactly
5. A user interface that minimally implements the arrow-key controls described above. You can add additional controls to affect speed and yaw if you wish.
6. Your webpage should include text instructions describing how the user interface works.
7. **Commenting**: You should comment each file with an author comment and comment each function you write with a header. Use JSDoc comments with the appropriate tags and types.
   Details can be found in the Google JavaScript Style Guide.

**4-Credit Students:**

In addition to above requirements, your code will also:

1. Implement floating spheres and/or cubes that are randomly placed thoughout the scene. The should remain stationary, although you can rotate or scale to make them more visually interesting.
2. Implement simple collision detection...if your flying camera collides with a floating object, restart the simulator....

**Submission:**

You will upload your files to compass in a zipped folder with name **${NetID}_MP2.zip**. Include all of the files necessary for your application to run locally. Name your webpage *Flight.html*

**3 Credit Rubric:**

| Feature | Points | Description |
|---|---|---|
| Interactive UI | 2pts | Using keyboard to control your plane interactively. |
| Translation | 2pts | The plane should move forward automatically. |
| Roll | 2pts | Apply correct rotation when your plane rolls left or right. |
| Pitch | 2pts | Apply correct rotation when your plane pitch up and down. |
| Terrain Generation | 3pts | Generate a finite mesh of triangles that resembles a terrain |
| Lighting | 2pt | Implement Phong/Blinn-Phong and render a well-lit and correctly shaded scene |
| Performance | 1pt | Your simulator should run interactively and should be reasonably efficient. |
| Documentation | 1pt | Your code should be commented and your user interface explained on your webpage |

**4 Credit Rubric:**

| Feature | Points | Description |
|---|---|---|

| | | |
|---|---|---|
| All the 3 Credit Features | 15pts | See above |
| Random Floating Spheres | 2pts | Render spheres or cubes…make them look interesting |
| Simple Collision Detection | 3pts | Detect collison between plane and sphere (or cubes) and restart |