CS367 Homework 5
Cityscape Flyover

Objectives

1. Generate basic geometric shapes of different shapes and dimensions
2. Implement user interactions for **camera control**
3. Use coordinate frames to manipulate object / camera pose
4. Use a JavaScript matrix library to manipulate matrices and vectors

Overview

The provided starter code includes JavaScript files for creating geometric shapes (`Object3D.js`, `Cone.js`, and `PolygonalPrism.js`). Using these classes you can create objects of different shapes and sizes. For instance, the following snippet creates a hexagonal prism and a cone of specific dimensions and colors.

```
let t1 = new PolygonalPrism (gl,
  {
    topRadius: 0.5,
    bottomRadius: 0.5,
    numSides: 6,
    height: 1,
    topColor: vec3.fromValues(1, 0, 0),    // optional argument
    bottomColor: vec3.fromValues(1,1,1)    // optional argument
  });

let t2 = new Cone (gl, {
  radius: 0.5, height: 0.8,
  tipColor: vec3.fromValues(0.03,0.4,0.29), // optional arg
  baseColor: vec3.fromValues(0,1,0)          // optional arg
  });
```

The provided sample file `render.js` has been setup to show two objects (a cone and a octagonal prism) (lines 73–89) viewed from a camera positioned at $(2, 4, 2)$ with a gaze point at $(0, 0, 1)$ (lines 48–49). The two objects are places on the ground (the XY-plane) so the camera up direction is the positive Z-axis (line 50).

Your Assignment

To make a more realistic cityscape, you must add more "buildings" to the currently bare downtown.

Build The City

- Use nested loops to systematically place at least 100 buildings to the scenery. Be sure to use loops; copy-and-pasting 100 chunks of code **will not be accepted** as a solution neither will it be considered for grading.
- These buildings must be placed in a grid, like what you expect to see in big cities.
- Use random numbers to vary the shape, height, colors, dimensions so you will obtain a more interesting scenery

Fly Over The Buildings

Imagine you have camera attached to an airplane. Now use keyboard listener to control the airplane (hence the camera) to fly over the city just built:

- Move forward or backward
- Pitch (nose up or down)
- Roll (bang left or right)
- Yaw (turn on its vertical axis)

You may have to initialize `viewMat` to view the entire cityscape (lines 47–51).

Recall that the optical axis of the WebGL camera is the Z-axis, thus the above four operations can be implemented as follow:

Move	translation along the camera Z-axis
Pitch	rotation around the camera X-axis
Roll	rotation around the camera Z-axis
Yaw	rotation around the camera Y-axis

Code Setup

1. Create a new directory for this homework assignment
2. Download `index.html` and `render.js` to this directory
3. Copy `webgl-utils.js` and `shader-utils.js` from your previous homework assignment
4. Download `gl-matrix-min.js` (in a ZIP file) from `https://github.com/toji/gl-matrix/releases`
5. Create a subdirectory `geometry`
6. Download `Object3D.js`, `PolygonalPrism.js`, `Cone.js`, and `Sphere.js` into the `geometry` subdirectory.
7. Complete the code for `Sphere.js`, use recursive subdivision explain in class to subdivide the initial tetrahedron to a sphere.

Extra credit

- The default setup above assume that the airplane flies along its Z-axis and the camera Z-axis is parallel to the airplane Z-axis. Sometimes, surveillance cameras are mounted at an angle, i.e. the camera Z-axis is no longer parallel with the airplane flight path.