Project 02 – Flight Sim

Due: Friday Nov 7
Total: 100 pts

You will be creating a flight simulator with randomly generated, and infinite, terrain.

LIGHTING

The lighting for the simulator will be fairly complex, and consist of:

- General ambient lighting
- A background sky including a sun
- Sun light from the sun that casts shadows on everything
- The world will go through a day-night cycle every minute: at 0 seconds it is midnight, 15 seconds it is sunrise, at 30 seconds it is noon, and at 45 seconds it is sunset. The following must change with the "time of day":
    - sun must change position
    - shadows must change with the sun position
    - light intensities and colors must change appropriately
- Shadows must be fairly good (minimal banding/acne) and must be visible even as the plane moves across the infinite terrain
- It should be a relatively clear day, but fog is used to occlude terrain further than 2 km away (which may not be generated yet) and its color must change with the time of day

GROUND

The ground is generated in squares of 2km by 2km. It is strongly recommended that you keep the units in the system in meters, thus the mesh will be 2000 x 2000 for a single square. The terrain is generated with the diamond-square algorithm. Code that runs the diamond-square algorithm is provided, but it only generates a 2D array of heights and not a geometry.

You will need to have a grid of these generated squares. You initially generate a 3x3 grid of squares. As you visit squares, additional squares are added to the grid to make sure that there is always all terrain within 2km of the plane. This must be done carefully to make sure the new grid squares match their edges to the existing squares using the arguments to provided code for matching. Note that they will still have some seams, but they will be very small and not very apparent unless viewing from a distance.

AIRCRAFT

The aircraft must be fairly complex. It can either be made from several built-in shapes (at least 10 shapes) or loaded from an external model file. The propellor on the airplane must spin based on the current thrust (a percent). The current speed and altitude of the plane must be displayed in the provided HTML GUI and updated continuously. The size of the plane must be realistically set in in meters. The plane spawns at $200\ \mathrm{m}$ above the ground at the origin.

FLYING

The plane maintains a thrust (initially 50%), velocity (initially $[-20, 0, 0]\ ^{m}/_{s}$), and angular velocity (initially $[0, 0, 0]\ ^{rad}/_{s}$) overtime and must fly according to:

- W/S – Pitch down/up (effects angular velocity by delta time times $2\ ^{rad}/_{s^2}$)
- A/D – Roll left/right (effects angular velocity by delta time times $2\ ^{rad}/_{s^2}$)
- Q/E – Yaw left/right (effects angular velocity by delta time times $2\ ^{rad}/_{s^2}$)
- Shift/Ctrl – Increase/Decrease throttle (by delta time, remember there is a range of 0 to 1)
- Space – Greatly decrease throttle (by delta time times 3, remember there is a range of 0 to 1)

The angular velocity also has a dampening factor of 3 times the delta time (minimum of 0), meaning that each component of the angular velocity is multiplied `1-3*deltaTime`.

The physics of the aircraft will be simplified for this project:

- Acceleration will be initialized to the direction that plane is facing multiplied by the current throttle times 10.
- The plane's y acceleration is affected by gravity and lift, so subtract gravity ($9.81\ ^{m}/_{s^2}$) and add the plane's x/z speed times 0.003.
- Compute the velocity from the acceleration, the acceleration is the added to the current velocity and then overall velocity is multiplied by 99% to account for air resistance.
- The overall speed must be capped to a maximum speed of $55\ ^{m}/_{s}$.
- The position is updated with the current velocity times delta time, with the altitude capped at $4200\ \text{m}$.
- The aircraft is rotated by its angular velocity times delta time.

After updating the plane's position and rotation, the camera is updated to be behind and looking at the plane.

COLLISION DETECTION

The collision detection algorithm will be published later, but in any case, once the plane lands on the ground, the simulation is stopped in its entirety except that the light cycling still occurs. The reset button will reset the position, velocity, angular velocity, and thrust of the plane to the original values allowing the plane to fly again. The ground stays the same.

CHECK-INS

There are weekly check-ins on the following dates where you will be expected to show the following progress:

- **Oct 27:** Day/night cycling and lighting along with ground. Recommend you use an orbit controller or similar at this stage so it is easier to examine the environment. The initial 3x3 grid of ground should be made (and the seams should be minimal) but the auto-generation does not need to be working.
- **Nov 3:** Rendering of the aircraft along with flying and infinite ground generation. No collision detection.

Each check-in is worth up to negative 10 pts if not successfully completed.

CODE QUALITY NOTES

- All geometry and materials reused as possible
- No new objects should be created during regular animation iterations (expanding the terrain which only happens infrequently can allocate new objects).
- Magic numbers should be reduced as possible.
- General code quality: good documentation, good variable names, good division of functions, etc.

RUBRIC

20 pts	Lighting: sky, time-based positioning, dynamic light intensities/colors, good shadows, fog and its color
20 pts	Ground: generation of initial squares, alignment of squares, infinite terrain
20 pts	Aircraft: model with realistic scaling, spinning propellor, UI updates, plane spawning
20 pts	Flying: working controls, working physics
20 pts	Collision Detection: efficient computation of “landing”

Bad code style, including not reusing geometry and materials as possible, are deductions on top of completing features. Each check-in is worth up to negative 10 pts if not successfully completed.