Computer Graphics Game Report

Introduction:

This program is a simple flight simulator which provides the player with a mountainous map to explore. The player starts in flat runway and is placed back at the same location every time a collision is detected. Engine power is increased by holding the spacebar and is decreased by left alt. The level of current engine power is indicated by the red bar to the right of the screen. The plane will pitch up and down by holding the “w” and “s” key respectively, roll is changed by the “a” and “d” keys and yaw is changed by the “q” and “e” keys. However, roll only changes the camera angle and the angle of the plane and does not contribute to turning. To take off, the power button should be held down so that the plane can reach sufficient speed on the runway. The player must then pitch up to leave the ground. To view the plane at different angles, hold and drag with the left mouse button. While not holding the left mouse button, the camera will simply follow the plane from behind.

Implemented features:

- Directional light: The sun was implemented as a directional light source.
- Loading .obj files: loaded plane, variable amount of trees and light poles.
- Multiple cameras: camera can either be fixed behind the plane or rotated freely around the plane by using the mouse.
- Multiple shaders: a shader was implemented for the skybox, 3D scene and the HUD.
- Collision detection: when the plane crashes it is reset to the airport, this is implemented by comparing the y position of the model to the corresponding y value in the terrain mesh, if the y position of the model is less than the y value of the terrain, a collision occurs.
- Texture mapping: the terrain and trees have textures.
- Depth cue: the terrain texture changes between snow, rock and grass with three specified height levels.
- Skybox: a skybox was implemented using a cube map texture.
- Height mapping: a ppm image file was generated with a Perlin noise function and used to generate the terrain grid.
- Procedural terrain: the location and size of trees is generated randomly; the terrain was generated from using a Perlin noise function.
- Spotlight: spotlights were implemented around the runway with multiple light poles.
- Point light: the plane has a headlight which is a point light.
- Light attenuation: the point and spot lights drop off with distance per an inverse square law.

Extra features

- HUD: 2 bars are implemented to record the player's altitude and engine power.
- Third person camera with z axis roll: this was implemented to simulate the view of the plane when it is rolling.
- Day-night cycle: a day night cycle was implemented, the direction of the light from the sun changes over the day. A lower level of light was added to simulate the moon. The sun rises correctly in the east and sets in the west. The brightness was also varied during the day with respect to time. Lighting was added to the skybox to create a more realistic feeling night time experience. The lighting equations on the skybox were also modified to give a red tinge during the sunset and sunrise.
- Physics: gravity was implemented by subtracting the inverse of speed from the current y position. A number was added to the speed to not divide by 0 when stationary. This results in a faster decrease in altitude as speed is reduced. Drag was implemented by a steady reduce in power if spacebar is not held down. The plane will crash into the ground if no power is given for an extended period as speed will reach close to 0.

Design decisions/challenges:

Height map:

The terrain was generated by stacking multiple Perlin noise generations on top of each other. One of the challenges faced was integrating an airport and raised mountain edges naturally into the procedural terrain. Simply adding the two caused unnaturally large cliffs and stretched textures. To overcome this, the GIMP image editor was used to edit the airport and raised edges in smoothly.

Day-night cycle:

The original day-night cycle involved the use of 2 different skybox textures, one for day and the other for night. This proved to be very immersion breaking as the skybox did not dim gradually but suddenly change after “sunset”. This was solved by using lighting on the skybox textures itself. The lighting equation sets a base brightness of 0.2 (equal to night time brightness) and changes throughout the day by adding the sine of the time multiplied by Pi; this is divided by length of day divided by two. The red part of the colour vector for sunlight also has an extra component which is defined by the cosine of time multiplied by pi divided by length of day divided by two. This results in a red tinge in the sky during the sunrise and sunset periods and white light for most the day, a factor is multiplied to this component to reduce the intensity of the red tinge for realism purposes. This same lighting equation is applied to the “sun”.

Camera:

Since planes rotate in all three dimensions, the camera also needed to tilt with the z axis roll of the plane. This was achieved by changing the up vector in the inbuilt glm::lookAt function. The y value of the up vector was kept at a constant 1.0, and the x and z values were varied per the current yaw, pitch and roll to achieve the correct tilt.

Physics:

A challenge faced while attempting to add realistic physics was the roll function. The plane should turn left if it is rolling left and pitching up, whereas pitching down while rolling left should turn the plane right. This was all achieved, however the plane’s movement through the world was very unintuitive. Due to this, we decided to make rolling the plane have no effect on its movement. Instead, the plane is moved by yawing and pitching to move left/right and up/down respectively.

Heads up display:

The decision was made to draw the HUD in a separate shader. This is because it did not share many attributes with the 3d scene, and would have required many uniform variables to be used as control signals to draw correctly. To correctly make the HUD appear over the screen, an orthographic projection was used with depth testing disabled.

Team Roles:

Michael Vincent (a1670261):

- HUD
- Lighting (3)
- Texturing (1)
- Height map (1)
- Collision detection (1)
- Loading .obj files (3)
- Cameras (2)
- Terrain mesh generation from height map

Zhongfan Zhang (a1670764):

- Day-night cycle
- Physics
- Depth cue (1)
- Skybox (1)
- Report
- Multiple shaders (2)
- Procedural generation of trees on terrain (2)