Sam Willems COSC363 Assignment 1 Report

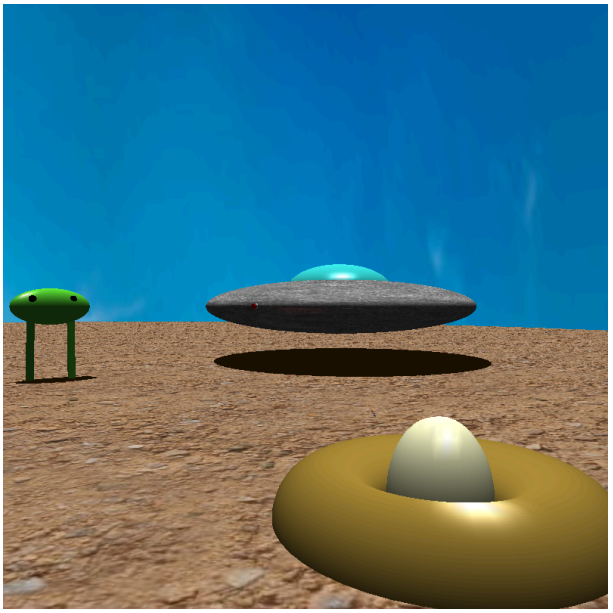
swi126 - 31643284

Declaration

I declare that this assignment submission represents my own work (except for allowed material provided in the course), and that ideas or extracts from other sources are properly acknowledged in the report. I have not allowed anyone to copy my work with the intention of passing it off as their own work.
Name: Sam Willems
Student ID: 31643284
Date: 28 March 2024

Scene Description

The scene is set in a desert climate, with an orange-brown dirt ground. The sky is sunny and blue, with a few clouds scattered about. Within this terrain lies a ufo, with coloured lights and spinning, seemingly hovering in the air despite no obvious means of propulsion. Guarding the ufo is a green, strange looking alien, walking around the ufo as if on lookout for danger. Off to the side of the ufo is a nest with a large egg in it. Perhaps the alien is protecting its soon-to-be child…



Controls

LEFT ARROW KEY - Turn camera 5º to the left
RIGHT ARROW KEY - Turn camera 5º to the right
UP ARROW KEY - Move forward
DOWN ARROW KEY - Move backward
SPACEBAR - Watch the ufo take off (toggleable)

Build Instructions

This program was primarily developed on a Windows computer using Visual Studio. To build and run the project using Visual Studio on Windows, follow the commands described on the COSC363 Learn page in the file Setting up Visual Studio with OpenGL on Windows. To build and run the project on Linux with the terminal, use the following steps.

1. Navigate to the root directory of the project.
2. Run the CMake file with the following command:

```
cmake CMakeLists.txt
```

3. Now build the project using the make command:

```
make
```

4. To run the program, use the following command:

```
./Scene.out
```

Extra Features Implemented

1. Planar shadows cast by the ufo and the alien.
2. A spotlight on a moving/rotating object. There are two of these, one on each side of the ufo (red and green), pointing back at the ufo for dramatic effect.
3. A sky dome.
4. A custom-built surface of revolution (more details below).
5. Spaceship liftoff.

Sweep Surface Explanation

The implementation for the torus that holds the egg in the scene was developed by my own working out and trial and error, however the example provided in the lecture 5 notes was used as a reference when I needed some guidance.

Firstly two initial rings are generated containing the x, y and z coordinates within a slice, based on the major radius given. These are the points on the slices that will be used to generate the Quads. The second ring is generated from the first ring, with the height and radius of the slice adjusted based on the angle around the minor radius.

```
    double nextVAngle = vAngle + 180. / numSlices;
    double nextHRadius = initHRadius - vRadius * cosf(toRad(nextVAngle));
    for (int i = 0; i < numPoints; i++) {

        vx[i] = hRadius * sinf(toRad(hAngle));
        vy[i] = vRadius * sinf(toRad(vAngle));
        vz[i] = hRadius * cosf(toRad(hAngle));

        wx[i] = nextHRadius * sinf(toRad(hAngle));
        wy[i] = vRadius * sinf(toRad(nextVAngle));
        wz[i] = nextHRadius * cosf(toRad(hAngle));

        hAngle += 360. / numPoints;
    }
```

Following this, the quads are generated based on these points, along with the normal vectors at each vertex. The normal vectors are calculated based on the points relative to the origin and the angle parameters, as opposed to using a vertex's neighbouring vertices, which is possible due to the symmetry that each ring has around the origin.

```
for (int i = 0; i < numPoints; i++) {
    glNormal3f(-vx[i] * -cosf(toRad(hAngle)), sin(toRad(vAngle)), -vz[i] * -cosf(toRad(hAngle)));
    glVertex3f(vx[i], vy[i], vz[i]);

    glNormal3f(-wx[i] * -cosf(toRad(hAngle)), sin(toRad(nextVAngle)), -wz[i] * -cosf(toRad(hAngle)));
    glVertex3f(wx[i], wy[i], wz[i]);

    glNormal3f(-wx[(i + 1) % (numPoints)] * -cosf(toRad(hAngle)), sin(toRad(nextVAngle)), -wz[(i + 1) % (numPoints)] * -cosf(toRad(hAngle)));
    glVertex3f(wx[(i + 1) % (numPoints)], wy[(i + 1) % (numPoints)], wz[(i + 1) % (numPoints)]);

    glNormal3f(-vx[(i + 1) % (numPoints)] * -cosf(toRad(hAngle)), sin(toRad(vAngle)), -vz[(i + 1) % (numPoints)] * -cosf(toRad(hAngle)));
    glVertex3f(vx[(i + 1) % (numPoints)], vy[(i + 1) % (numPoints)], vz[(i + 1) % (numPoints)]);
}
```

After this, the points held in the first ring are replaced by the points in the following ring.

```
    for (int i = 0; i < numPoints; i++) {
        vx[i] = wx[i];
        vy[i] = wy[i];
        vz[i] = wz[i];
    }

    vAngle = nextVAngle;
    hRadius = nextHRadius;
```

These steps are then repeated for the number of slices needed in the torus, completing a half circle. (ie, a donut cut in half along the xz plane).

References

COSC363 Staff for notes about relevant topics and example pieces of code.
No third party models were used.

Dirt texture: https://www.deviantart.com/ditto209/art/Dirt-texture-803218777
Metal texture on ufo:
https://www.rawpixel.com/image/5921174/photo-image-background-texture-public-domain
Skydome texture: https://stock.adobe.com/search?k=%22seamless+sky+background%22