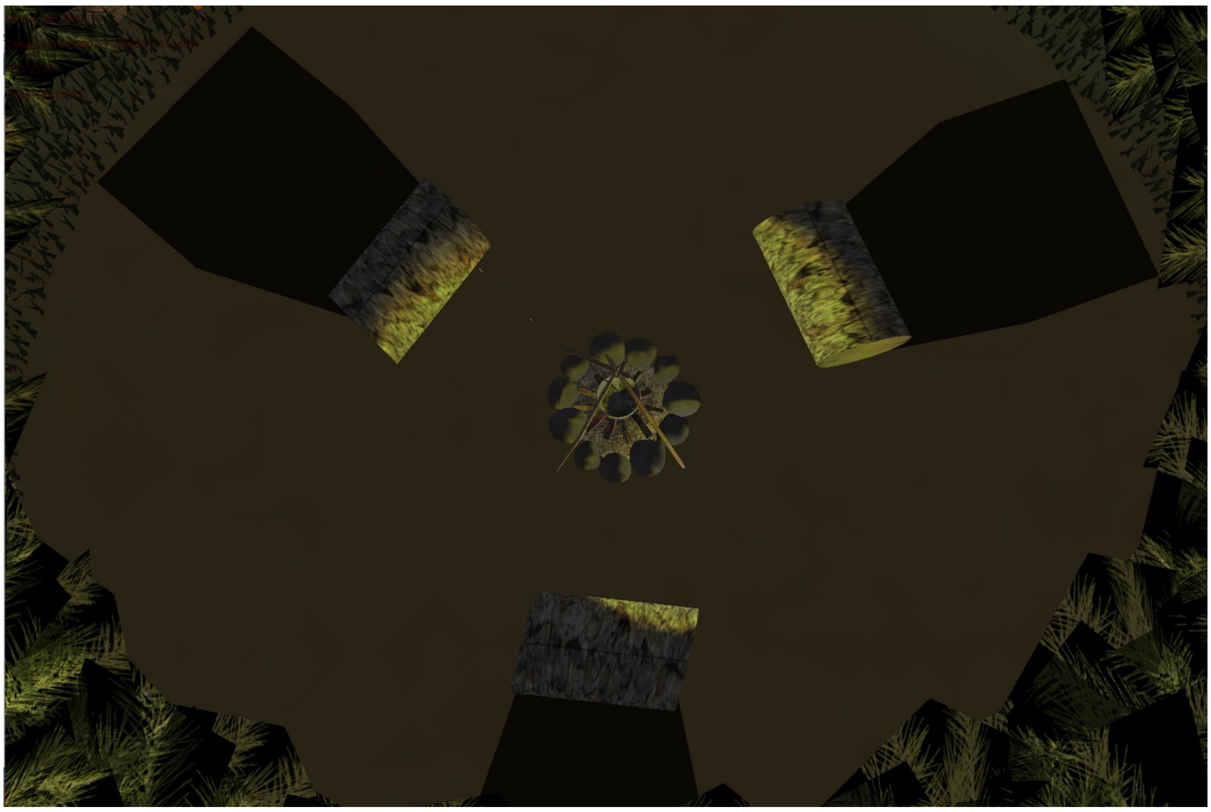
A stroll in the woods



Jamie Haddow

CMP203

0705082

Overview

For the given task, I created a “mostly” calming forest with a few of my favourite characters for games/anime. This open area gave me the space to show off the different weekly techniques, and present the opportunity to show my own originality.

This application was created in C++ using OpenGL2.0(Fixed pipeline) with which a framework was provided by the lecturer. I used a single external library, irrKlang for audio (“https://www.ambiera.com/irrklang/”)

Controls

Camera 1:

W,A,S,D for general camera movement. Z and X for vertical movement.

Mouse movement controls the Pitch and Yaw

Camera 2 and 3

No specific controls as they are fixed positions

User Control:

- and + keys turn wireframe on and off.

M,N,B,V to control the arm above the firepit

K turns on a transparent stencilled version of the world above the scene and a great cost to the fps of the program. Please use with caution!

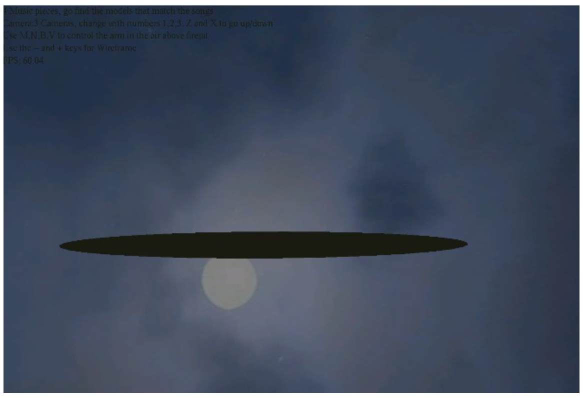
Requirements

Following the project brief, the requirements are:

- Geometry
- Lighting
- Camera and Interaction
- Hierarchical Modeling
- Advance features
- A wire frame mode
- OOP

I will now go into more detail, each of these requirements and how I feel I met the needs of each

Geometry



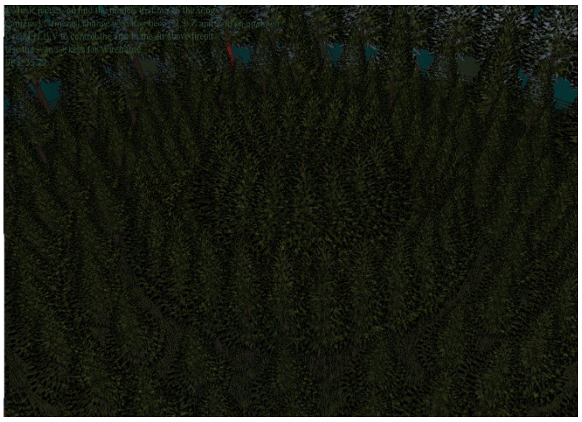
```
4 Music pieces, go find the models that match the songs
Camera:3 Cameras, change with numbers 1,2,3. Z and X to go up/down
Use M,N,B,V to control the arm in the air above firepit
Use the – and + keys for Wireframe
FPS: 60.04
```

SkyBox

Using a sphere with a camera

Placed inside using depth test

Created an endless scene.



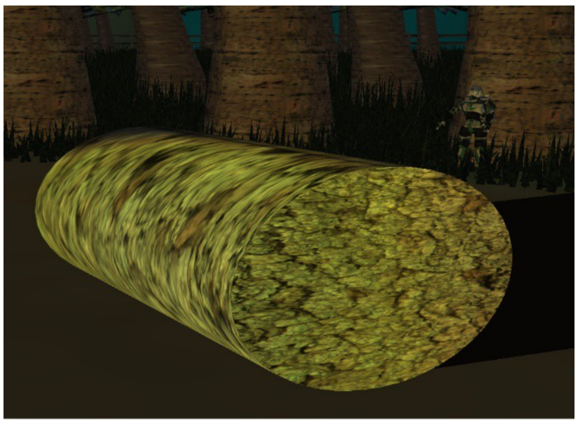
```
4 Music pieces, go find the models that match the songs
Camera:3 Cameras, change with numbers 1,2,3. Z and X to go up/down
Use M,N,B,V to control the arm in the air above firepit
Use the – and + keys for Wireframe
FPS: 35.72
```

Hand crafted scenery

By using models found online

I created my own forest using

Loops and rotations.



Procedural generation

I generated a disc to create the floor.

I then created a cylinder and used 2

of the discs I created to make a log

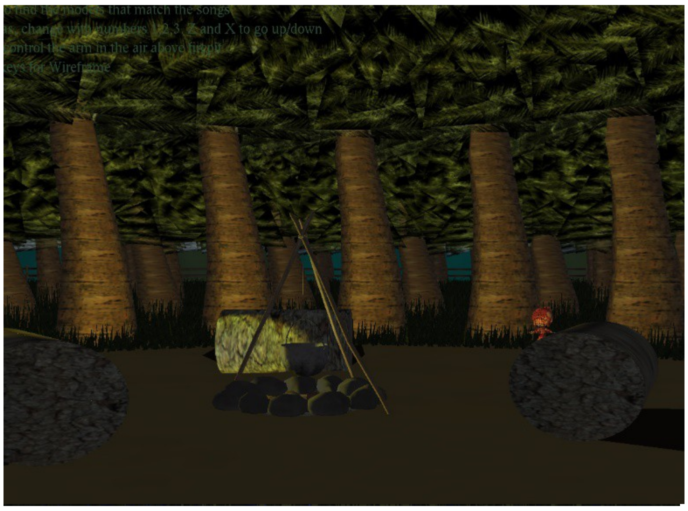
shape used in the scene.

Lighting

This project contains 7 active lights and 1 “world light”. By world light, this is simply a light with an ambient value attached to it to give a general brightness to the scene.

2 point lights were used for the firelight to give a warm glow in the open area in the center of the scene and with the zombie wandering around the scene in the trees.

```
GLfloat Diffuse[] = { 10.0, 7.8, 0.5, 1.0f };
GLfloat Position[] = { 0.0f, 0.0, 0.0, 1.0 };
glLightfv(GL_LIGHT2, GL_DIFFUSE, Diffuse);
glLightfv(GL_LIGHT2, GL_POSITION, Position);;
glLightf(GL_LIGHT2, GL_CONSTANT_ATTENUATION, 0.5);
glLightf(GL_LIGHT2, GL_LINEAR_ATTENUATION, 0.0);
glLightf(GL_LIGHT2, GL_QUADRATIC_ATTENUATION, quad);
glEnable(GL_LIGHT2);
```


o find the models that match the songs
as, change with numbers 1,2,3, Z and X to go up/down
control the arm in the air above firepit
keys for Wireframe

4 Spot lights were used to highlight the models placed in 4 corners of the scene.

```
GLfloat Diffuse[]   = { 2.0f,    0.0f, 0.0f, 1.0f };
GLfloat Position[]  = { -110.0f, 3.0f, 0.0f, 1.0f };
GLfloat Direction[] = { -1.0f,   0.0f, 0.0f };

glLightf(GL_LIGHT4, GL_SPOT_CUTOFF, 25.0f);
glLightfv(GL_LIGHT4, GL_SPOT_DIRECTION, Direction);
glLightf(GL_LIGHT4, GL_SPOT_EXPONENT, 20.0);
glLightfv(GL_LIGHT4, GL_DIFFUSE, Diffuse);
glLightfv(GL_LIGHT4, GL_POSITION, Position);
glEnable(GL_LIGHT4);
```

Camera

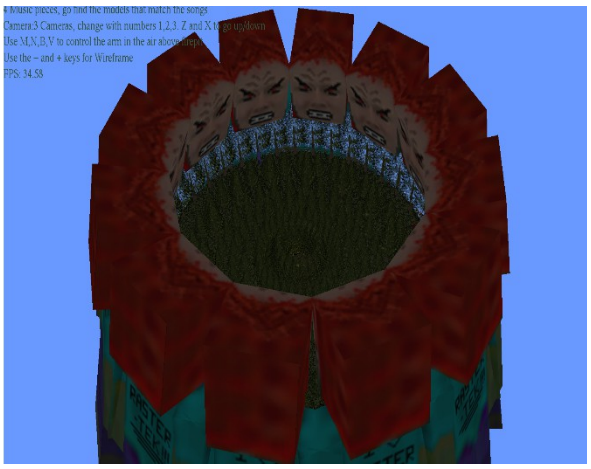
This project contains 3 cameras. Camera 1 is free roaming. Camera 2 is a static location above the scene giving the user an overall view of the forest. Camera 3 is a static camera that rotates around the Y axis following the zombie that wanders in the forest.

I was hoping to use Camera 3 as a 3rd person follow on the zombie, however I was unable to retrieve the objects positions using the worldMatrix so I had to resort to a much simpler solution of setting the camera to a set position just above the origin on the Y axis and then have the camera rotate on the yaw axis using the same variable that the zombie uses to simulate it following the object.

Below is the code I used to initialise the camera positions for use and how I setup camera 3 to follow the zombie with a few images of the different cameras in use.

```
camera.cameraInit(3.0f, 3.0f, 10.0f, 0.0f, 0.0f, 0.0f, 0.0f, 0.0f, 0.0f);
camera2.cameraInit(113.0f, 365.0f, -314.0f, 0.0f, 0.0f, 0.0f, 0.0f, 0.0f, 0.0f);
camera3.cameraInit(0.0f, 5.0f, 0.0f, 0.0f, 0.0f, 1.0f, 0.0f, 0.0f, 0.0f);
```

```
camera3.setPitchYaw(0, rotation);
camera3.update();
```



4 Music pieces, go find the models that match the songs
Camera:3 Cameras, change with numbers 1,2,3. Z and X to go up/down
Use M,N,B,V to control the arm in the air above firepit
Use the – and + keys for Wireframe
FPS: 34.58

Hierarchical modelling

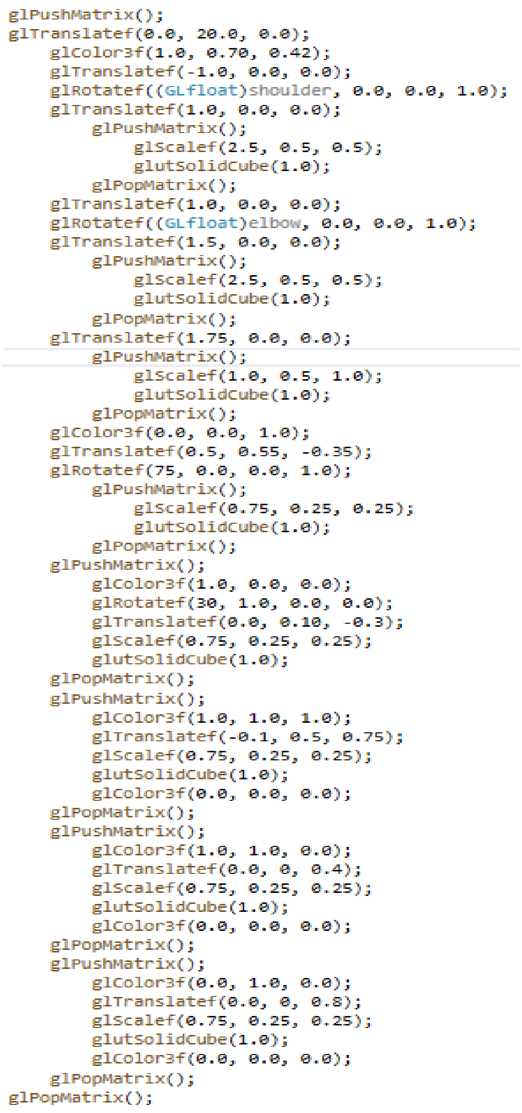
I used matrix stacks throughout the project. Here are a few examples.

1.) Creating my arm object from the openGL red book.

2.) Setting my models locations.

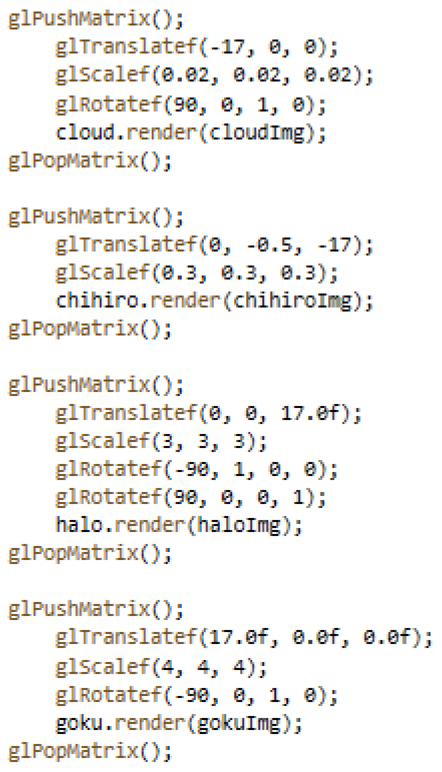
3.) Stencil buffering my objects.

1.)



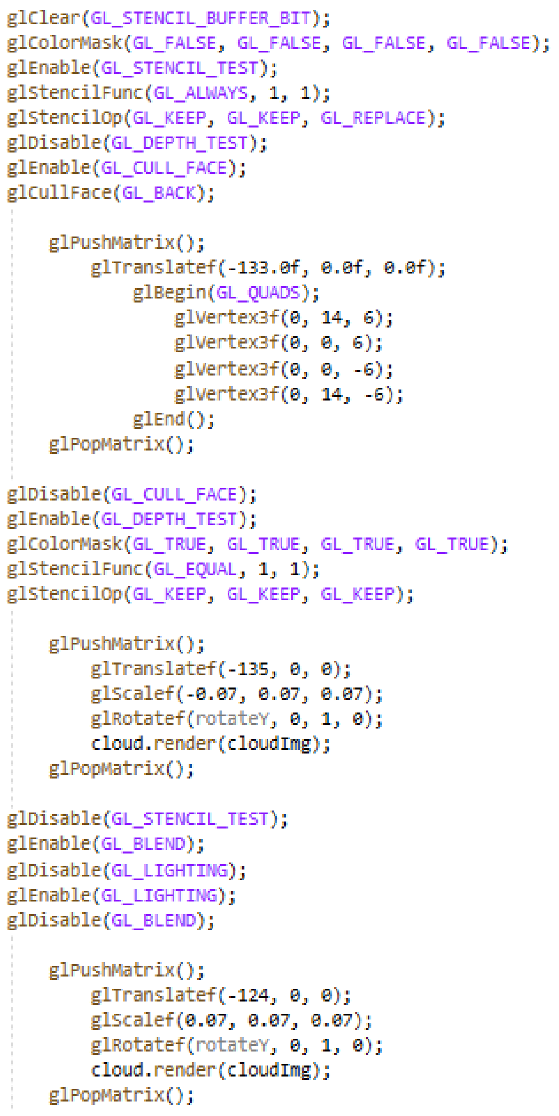
```
glPushMatrix();
glTranslatef(0.0, 20.0, 0.0);
    glColor3f(1.0, 0.70, 0.42);
    glTranslatef(-1.0, 0.0, 0.0);
    glRotatef((GLfloat)shoulder, 0.0, 0.0, 1.0);
    glTranslatef(1.0, 0.0, 0.0);
        glPushMatrix();
            glScalef(2.5, 0.5, 0.5);
            glutSolidCube(1.0);
        glPopMatrix();
    glTranslatef(1.0, 0.0, 0.0);
    glRotatef((GLfloat)elbow, 0.0, 0.0, 1.0);
    glTranslatef(1.5, 0.0, 0.0);
        glPushMatrix();
            glScalef(2.5, 0.5, 0.5);
            glutSolidCube(1.0);
        glPopMatrix();
    glTranslatef(1.75, 0.0, 0.0);
        glPushMatrix();
            glScalef(1.0, 0.5, 1.0);
            glutSolidCube(1.0);
        glPopMatrix();
    glColor3f(0.0, 0.0, 1.0);
    glTranslatef(0.5, 0.55, -0.35);
    glRotatef(75, 0.0, 0.0, 1.0);
        glPushMatrix();
            glScalef(0.75, 0.25, 0.25);
            glutSolidCube(1.0);
        glPopMatrix();
    glPushMatrix();
        glColor3f(1.0, 0.0, 0.0);
        glRotatef(30, 1.0, 0.0, 0.0);
        glTranslatef(0.0, 0.10, -0.3);
        glScalef(0.75, 0.25, 0.25);
        glutSolidCube(1.0);
    glPopMatrix();
    glPushMatrix();
        glColor3f(1.0, 1.0, 1.0);
        glTranslatef(-0.1, 0.5, 0.75);
        glScalef(0.75, 0.25, 0.25);
        glutSolidCube(1.0);
        glColor3f(0.0, 0.0, 0.0);
    glPopMatrix();
    glPushMatrix();
        glColor3f(1.0, 1.0, 0.0);
        glTranslatef(0.0, 0, 0.4);
        glScalef(0.75, 0.25, 0.25);
        glutSolidCube(1.0);
        glColor3f(0.0, 0.0, 0.0);
    glPopMatrix();
    glPushMatrix();
        glColor3f(0.0, 1.0, 0.0);
        glTranslatef(0.0, 0, 0.8);
        glScalef(0.75, 0.25, 0.25);
        glutSolidCube(1.0);
        glColor3f(0.0, 0.0, 0.0);
    glPopMatrix();
glPopMatrix();
```

2.)



```
glPushMatrix();
    glTranslatef(-17, 0, 0);
    glScalef(0.02, 0.02, 0.02);
    glRotatef(90, 0, 1, 0);
    cloud.render(cloudImg);
glPopMatrix();

glPushMatrix();
    glTranslatef(0, -0.5, -17);
    glScalef(0.3, 0.3, 0.3);
    chihiro.render(chihiroImg);
glPopMatrix();

glPushMatrix();
    glTranslatef(0, 0, 17.0f);
    glScalef(3, 3, 3);
    glRotatef(-90, 1, 0, 0);
    glRotatef(90, 0, 0, 1);
    halo.render(haloImg);
glPopMatrix();

glPushMatrix();
    glTranslatef(17.0f, 0.0f, 0.0f);
    glScalef(4, 4, 4);
    glRotatef(-90, 0, 1, 0);
    goku.render(gokuImg);
glPopMatrix();
```

3.)

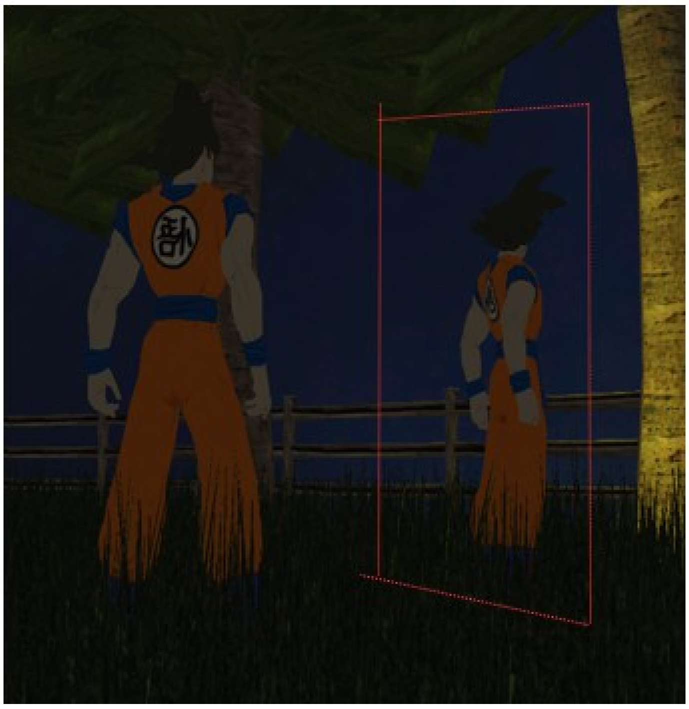


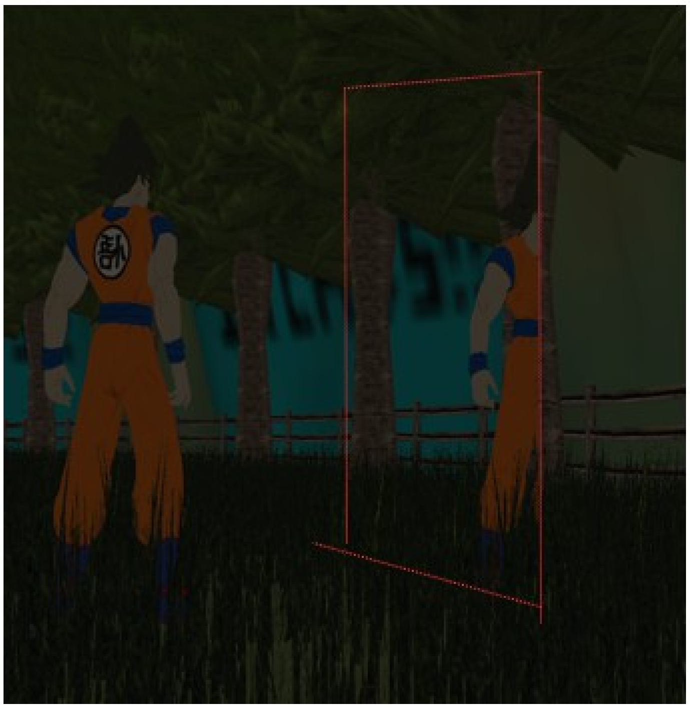
```
glClear(GL_STENCIL_BUFFER_BIT);
glColorMask(GL_FALSE, GL_FALSE, GL_FALSE, GL_FALSE);
glEnable(GL_STENCIL_TEST);
glStencilFunc(GL_ALWAYS, 1, 1);
glStencilOp(GL_KEEP, GL_KEEP, GL_REPLACE);
glDisable(GL_DEPTH_TEST);
glEnable(GL_CULL_FACE);
glCullFace(GL_BACK);

    glPushMatrix();
        glTranslatef(-133.0f, 0.0f, 0.0f);
            glBegin(GL_QUADS);
                glVertex3f(0, 14, 6);
                glVertex3f(0, 0, 6);
                glVertex3f(0, 0, -6);
                glVertex3f(0, 14, -6);
            glEnd();
    glPopMatrix();

glDisable(GL_CULL_FACE);
glEnable(GL_DEPTH_TEST);
glColorMask(GL_TRUE, GL_TRUE, GL_TRUE, GL_TRUE);
glStencilFunc(GL_EQUAL, 1, 1);
glStencilOp(GL_KEEP, GL_KEEP, GL_KEEP);

    glPushMatrix();
        glTranslatef(-135, 0, 0);
        glScalef(-0.07, 0.07, 0.07);
        glRotatef(rotateY, 0, 1, 0);
        cloud.render(cloudImg);
    glPopMatrix();

glDisable(GL_STENCIL_TEST);
glEnable(GL_BLEND);
glDisable(GL_LIGHTING);
glEnable(GL_LIGHTING);
glDisable(GL_BLEND);

    glPushMatrix();
        glTranslatef(-124, 0, 0);
        glScalef(0.07, 0.07, 0.07);
        glRotatef(rotateY, 0, 1, 0);
        cloud.render(cloudImg);
    glPopMatrix();
```

Advance features

Stencil buffer usage

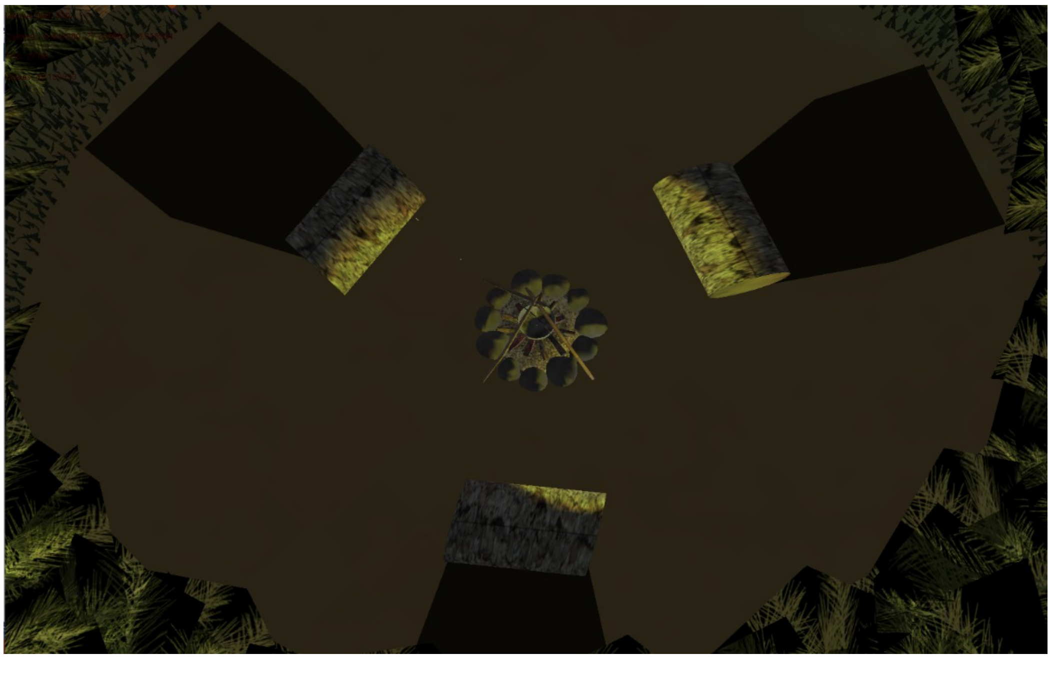
In my scene I used stencil buffering on a number of my objects. You can see this used on the 4 models around the edge of the scene. If you also look up and press K you can also see a silly attempt at stencilling the entire world at a huge fps cost(at least on debug mode, not so much on release).





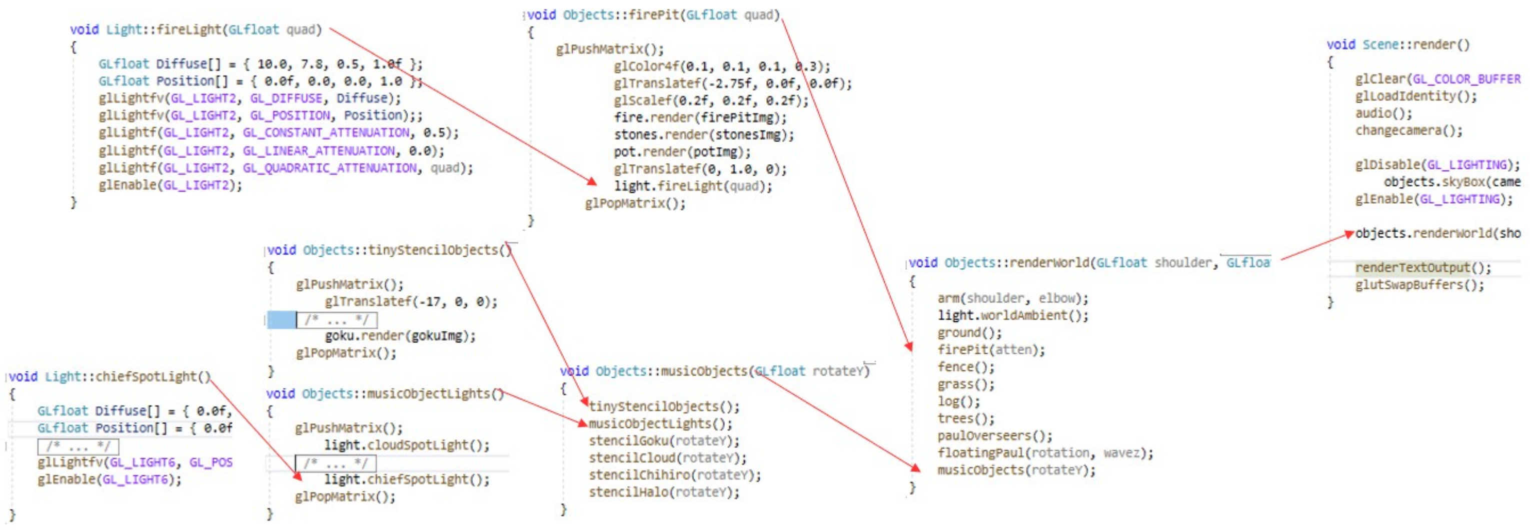
Shadow usage

I used the second approach from the lecture, planar shadows for my scene. I was going to do the same for the front row of trees however, due to the grass obscuring most of the floor,it would be a waste of resources



Object Oriented

I did my best to reduce the number of function calls in render and the following image shows my thought process.



```
void Light::fireLight(GLfloat quad)
{
    GLfloat Diffuse[] = { 10.0, 7.8, 0.5, 1.0f };
    GLfloat Position[] = { 0.0f, 0.0, 0.0, 1.0 };
    glLightfv(GL_LIGHT2, GL_DIFFUSE, Diffuse);
    glLightfv(GL_LIGHT2, GL_POSITION, Position);;
    glLightf(GL_LIGHT2, GL_CONSTANT_ATTENUATION, 0.5);
    glLightf(GL_LIGHT2, GL_LINEAR_ATTENUATION, 0.0);
    glLightf(GL_LIGHT2, GL_QUADRATIC_ATTENUATION, quad);
    glEnable(GL_LIGHT2);
}
```

```
void Objects::firePit(GLfloat quad)
{
    glPushMatrix();
            glColor4f(0.1, 0.1, 0.1, 0.3);
            glTranslatef(-2.75f, 0.0f, 0.0f);
            glScalef(0.2f, 0.2f, 0.2f);
            fire.render(firePitImg);
            stones.render(stonesImg);
            pot.render(potImg);
            glTranslatef(0, 1.0, 0);
            light.fireLight(quad);
        glPopMatrix();
}
```

```
void Scene::render()
{
    glClear(GL_COLOR_BUFFER
    glLoadIdentity();
    audio();
    changecamera();

    glDisable(GL_LIGHTING);
        objects.skyBox(came
    glEnable(GL_LIGHTING);

    objects.renderWorld(sho

    renderTextOutput();
    glutSwapBuffers();
}
```

```
void Objects::tinyStencilObjects()
{
    glPushMatrix();
        glTranslatef(-17, 0, 0);
    /* ... */
        goku.render(gokuImg);
    glPopMatrix();
}
```

```
void Objects::renderWorld(GLfloat shoulder, GLfloa
{
    arm(shoulder, elbow);
    light.worldAmbient();
    ground();
    firePit(atten);
    fence();
    grass();
    log();
    trees();
    paulOverseers();
    floatingPaul(rotation, wavez);
    musicObjects(rotateY);
}
```

```
void Light::chiefSpotLight()
{
    GLfloat Diffuse[] = { 0.0f,
    GLfloat Position[] = { 0.0f
    /* ... */
    glLightfv(GL_LIGHT6, GL_POS
    glEnable(GL_LIGHT6);

}
```

```
void Objects::musicObjectLights()
{
    glPushMatrix();
        light.cloudSpotLight();
    /* ... */
        light.chiefSpotLight();
    glPopMatrix();
}
```

```
void Objects::musicObjects(GLfloat rotateY)
{
    tinyStencilObjects();
    musicObjectLights();
    stencilGoku(rotateY);
    stencilCloud(rotateY);
    stencilChihiro(rotateY);
    stencilHalo(rotateY);
}
```

Code examples

Here are a few extra examples of my code:

Below: my floor generation

Right: my grass functions

```
void Shape::calculatefloor(int floorsize, float floor)
{
    floorVertexCount = 0;
    float theta = ((2 * 3.1415) / floorsize);
    float Theta2 = 0;
    float diameter = 2 * floor;
    for (int i = 0; i < floorsize; i++)
    {
        //the 1st vertex which is he origin. x,y,z. each loop we will be calling
        //x
        floorTexcoords.push_back(0.5f);
        floorVerts.push_back(0.0f);
        floorNorms.push_back(0.0f);
        //y
        floorTexcoords.push_back(0.5f);
        floorVerts.push_back(0.0f);
        floorNorms.push_back(0.0f);
        //z
        floorVerts.push_back(0.0f);
        floorNorms.push_back(1.0f);

        //The 2nd vertex in the loop is the origin + the radius to get our first
        //x
        floorTexcoords.push_back(floor * (cosf(Theta2) / diameter) + 0.5f);
        floorVerts.push_back(floor * (cosf(Theta2)));
        floorNorms.push_back(0.0f);
        //y
        floorTexcoords.push_back(floor * (sinf(Theta2) / diameter) + 0.5f);
        floorVerts.push_back(floor * (sinf(Theta2)));
        floorNorms.push_back(0.0f);
        //z
        floorVerts.push_back(0.0f);
        floorNorms.push_back(1.0f);


        if (Theta2 >= (2 * 3.1415))
        {
            Theta2 = 0;
        }
        else { Theta2 += theta; };

        //Origin + theta  + radius
        //x
        floorTexcoords.push_back(floor * (cosf(Theta2) / diameter) + 0.5f);
        floorVerts.push_back(floor * (cosf(Theta2)));
        floorNorms.push_back(0.0f);
        //y
        floorTexcoords.push_back(floor * (sinf(Theta2) / diameter) + 0.5f);
        floorVerts.push_back(floor * (sinf(Theta2)));
        floorNorms.push_back(0.0f);
        //z
        floorVerts.push_back(0.0f);
        floorNorms.push_back(1.0f);
        floorVertexCount += 3;
    }
}
```

```
void Objects::grass()
{
    for (int i = 0; i < 40; i++)
    {
        glPushMatrix();
        //(360 / 40) giving the rotation amount
            glRotatef(9 * i, 0, 1, 0);
            glTranslatef(0.0f, 0.0f, 20.0f);
            glScalef(0.01f, 0.004f, 0.01f);
            sceneGrass.render(grassImg);
        glPopMatrix();
    }

    for (int i = 0; i < 40; i++)
    {
        glPushMatrix();
            glRotatef(9 * i, 0, 1, 0);
            glTranslatef(0.0f, 0.0f, 26.0f);
            glScalef(0.01f, 0.007f, 0.01f);
            sceneGrass.render(grassImg);
        glPopMatrix();
    }
```

```
float grassSize = 40;
float grassRotate;
int grassTranslate = 34;
grassRotate = 360 / grassSize;

for (int j = 0; j < 17; j++)
{
    for (int i = 0; i < grassSize; i++)
    {

        glPushMatrix();
            glRotatef(grassRotate * i, 0, 1, 0);
            glTranslatef(0.0f, 0.0f, grassTranslate);
            glScalef(0.01f, 0.01f, 0.01f);
            sceneGrass.render(grassImg);
        glPopMatrix();
    }
    grassTranslate = grassTranslate + 7;
    grassSize = grassSize + 5;
    grassRotate = 360 / grassSize;
}
```

References

The websites I used for 3d models were:

https://www.turbosquid.com/ https://www.models-resource.com/ https://free3d.com/

The music I used in my project I found on YouTube, downloaded in MP3 format and then converted to ogg using audacity. The songs I used were:

Halo theme song - https://www.youtube.com/watch?v=0jXTBAGv9ZQ

Dragonball Z 'Chala Hey chala' - https://www.youtube.com/watch?v=pYnLO7MVKno

Spirited away 'Inochi No Namae' - https://www.youtube.com/watch?v=ImPM5IDIYPs

FF7 Aeriths Theme - https://www.youtube.com/watch?v=flqKWLkm2-g

For my skybox, I used a sphere from 'song Ho Ahn' - http://www.songho.ca/opengl/