CITS3003 Graphics & Animation

Part 1 - Project Report

Jason Ankers - 21493118

Scene Editor

The final scene editor fulfils the requirements of the project outline with no known issues or bugs. Additional functionality also works correctly although has not been extensively tested.

Functionality implemented

I completed the first half of the scene editor, steps A-E, while my group partner Tom, completed the final 4 lighting steps. Coming from the labs I had a basic understanding of how OpenGL worked and didn't have too many issues getting started, my process for each step is outlined below.

A (Camera Rotation)

Camera rotation is handled by the two variables camRotUpAndOverDeg and camRotSidewaysDeg which were kindly already implemented. In order to apply these rotations to the view I used the following code

```
mat4 xrot = RotateX(camRotUpAndOverDeg);
mat4 yrot = RotateY(camRotSidewaysDeg);
view = view * xrot * yrot;
```

The only issue I had with this step was the camera seemed to glitch after clicking on the screen. This was kindly solved by an anonymous user on the CITS3003 help forum and involved editing gnatidread.h to change

```
clickPrev = currMouseXYscreen(mouseX, mouseY);
```

to:

```
prevPos = currMouseXYscreen(mouseX, mouseY);
```

B (Object Rotation and Texture Scaling)

Object rotation is controlled by the angles[3] array in the sceneObject struct. These values are modified using setToolCallbacks in the mainmenu function, this is also kindly already setup for us. Rotation is applied to the object in the same way as camera rotation.

```
mat4 xrot = RotateX(-sceneObj.angles[0]);
mat4 yrot = RotateY(sceneObj.angles[1]);
mat4 zrot = RotateZ(sceneObj.angles[2]);
model = model * xrot * yrot * zrot;
```

I had to tweak to x-rotation by negating so it would appear the same way as in the video. Lastly, texture scaling required modifying the fragment shader's call to texture2D.

```
texture2D(texture, texCoord*2.0) to texture2D(texture, texCoord*texScale)
```

C (Material Menu)

The material menu was modified to add a call to setToolCallbacks. Two callback functions were created for this.

```
setToolCallbacks(adjustAmbientDiffuse, mat2(1, 0, 0, 1),
            adjustShineSpecular, mat2(1, 0, 0, 10) );
```

These functions modify the specular, shine, ambient and diffuse attributes of the currently select scene object.

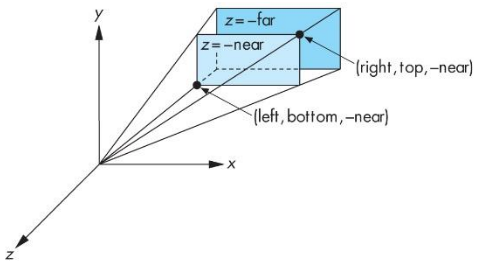
```
static void adjustShineSpecular(vec2 sh_sp)
 {
   sceneObjs[toolObj].specular+=sh_sp[0];
   sceneObjs[toolObj].shine+=sh_sp[1];
}
 static void adjustAmbientDiffuse(vec2 am_df)
 {
   sceneObjs[toolObj].ambient+=am_df[0];
   sceneObjs[toolObj].diffuse+=am_df[1];
 }
```

D (Closeup view)

For this step, I changed the reshape function to allow for closer objects. The original *nearDist* was 0.2, I changed it to 0.01 which fixed the problem, going smaller seemed to have no effect.

E (Window reshaping)

This also required a change in the reshape function. The projection is calculated with a call to Frustrum(left, right, bottom, top, near, far).



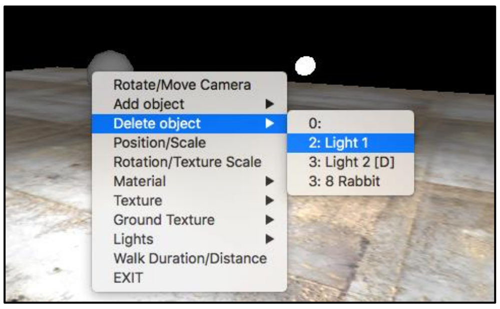
My understanding from the image to the left (taken from lab 5), is that by changing near, the view can be zoomed in and outwards. To keep the object visible like shown in the video, we want the view to zoom out as the width is decreased. To achieve this, I added the following code.

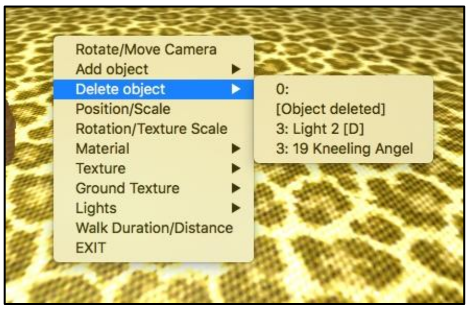
```
GLfloat nearDist = 0.01;
GLfloat near = nearDist;
if(width < height)
   near = near * (float)width/(float)height;

projection = Frustum(-nearDist*(float)width/(float)height,
                     nearDist*(float)width/(float)height,
                     -nearDist, nearDist, near, 100.0);
```

This change varies near based on the width/height, but only when width is less than height.

Extra features implemented

I extended the menu to allow for object deletion and light manipulation. The main issue with the usability of the scene editor is the selection of objects. I did some research on selecting objects using the mouse but this feature would require ray tracing, a considerable undertaking. The solution I found was to modify the menu after a new object is added/deleted. My delete menu contains a list of all the objects in the scene and new objects get added onto the bottom.

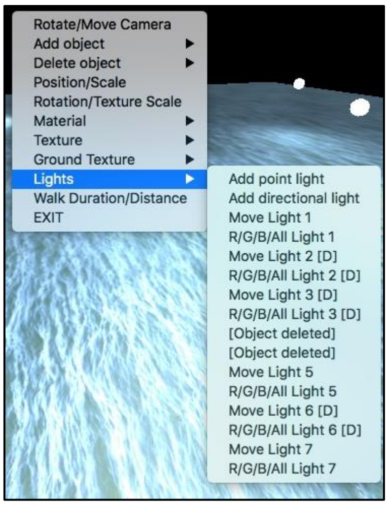




The problem with this approach is that removing menu items throws out the order. The solution was to just replace the item's name with [Object deleted].

After implementing the deletion menu, I worked with Tom who had been trying to get multiple lights to work. He needed a way to select and move around the large number of lights. I suggested we use a similar system to my delete menu. When adding and deleting lights from the scene, the menu is updated with two options for each light; move and RGB. This required a rewrite of the light menu system which can be seen below.

```
static void lightMenu(int id)
{
    deactivateTool();
    if (id == 70) {
        addLight(false);
    }

    else if (id == 71) {
        addLight(true);
    }
    //move
    else if(id % 2 == 0)
    {
        toolObj = lights[id/10];
        setToolCallbacks(adjustLocXZ, camRotZ(),
                         adjustBrightnessY,
                         mat2( 1.0, 0.0, 0.0, 10.0) );
    }
    //RGB
    else if(id % 2 == 1)
    {
        toolObj = lights[(id-1)/10];
        setToolCallbacks(adjustRedGreen,
                         mat2(1.0, 0, 0, 1.0),
                         adjustBlueBrightness,
                         mat2(1.0, 0, 0, 1.0) );
    }
}
```



The ID passed to the menu contains two parts, the first part is the light number and the second part is a one or a zero. This means that even IDs are for movement and odd IDs are for RGB. E.g. for light 5 the move ID would be 50 and the RGB ID would be 51. This mess of a menu is not ideal, as pictured above, however without ray tracing, object selection is difficult.

Personal reflection

This section of the project provided a detailed overview of some of the fundamental features OpenGL has to offer. Despite writing in a language I had rarely used before, I found the ‘step-by-step’ manner of the project to be helpful in preventing things from getting too complicated. Example videos showing the required functionality also helped to clear any confusion in terms of what was required. Working through the questions and solving the issues made it both a challenging and rewarding experience.