CITS3003 Graphics & Animation
Project report

Ka Hou Hong(22085304)

May 22, 2020

1 Overview

Parts A to I of the required functions of the project have been implemented, and the spotlight of part J have not been completed. The project will clearly explain the code of each part and the idea of implementing a spotlight.

2 Implementation

Part A: cameraRotate

For part A, the display(void) was modified. Both X and Y rotations were used with *camRotUpAndOverDeg* and *camRotSidewaysDeg* to achieve the correct camera rotation.

```
mat4 rotationA = RotateX(camRotUpAndOverDeg)*RotateY(camRotSidewaysDeg);
view = Translate(0.0,0.0,-viewDist) * rotationA;
```

Moreover, I encountered a problem that caused the camera to jump to a new position when I clicked the mouse button. This problem had to be fixed in activateTool().

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

Part B: objRotateXYZ

Object rotation was completed by multiplying the model with X, Y and Z rotation in the drawMesh function. The angles array stored 0, 1 and 2 which implies X, Y and Z separately. The x axis was negated because the positive x axis was not matched to the sample video.

```
mat4 rotationB = RotateX(-sceneObj.angles[0])*RotateY(sceneObj.angles[1])
*RotateZ(sceneObj.angles[2]);
mat4 model = Translate(sceneObj.loc) * Scale(sceneObj.scale) * rotationB;
```

Part C: Materials

Two functions were created for adjusting ambient, diffuse, specular and shine values of objects.

```
static void adjustAmbientAndDiffuse(vec2 AmbientAndDiffuse)
{
   sceneObjs[toolObj].ambient+=AmbientAndDiffuse[0];
   sceneObjs[toolObj].diffuse+=AmbientAndDiffuse[1];
}
static void adjustSpecularAndShine(vec2 SpecularAndShine)
{
   sceneObjs[toolObj].specular+=SpecularAndShine[0];
   sceneObjs[toolObj].shine+=SpecularAndShine[1];
}
```

Removed the *UNIMPLEMENTED* in glutAddMenuEntry() from makeMenu()

```
glutAddMenuEntry("Ambient/Diffuse/Specular/Shine",20);
```

Two functions were added to materialMenu

```
else if (id == 20)
    {
        toolObj = currObject;
        setToolCallbacks(adjustAmbientAndDiffuse, mat2(1, 0, 0, 1),
                                    adjustSpecularAndShine,  mat2(1, 0, 0, 1));
    }
```

Part D: Closeup

The *reshape* callback function was fixed. When the camera closes the object by reducing the value of the variable *nearDist*, cropping could be avoided. The improved *nearDist* enables the camera to perform more close-ups.

```
GLfloat nearDist = 0.005;
```

Part E: Reshape

Modified the reshape function to ensure that when the window is square, as the width or height decreases, anything visible should continue to be visible. A new projection was applied when the window's weight is less than height.

```
if(width<height){
   projection = Frustum(-nearDist, nearDist,
                       -nearDist*(float)height/(float)width,
                        nearDist*(float)height/(float)width,
                        nearDist, 100.0);
   }
else{
   projection = Frustum(-nearDist*(float)width/(float)height,
                        nearDist*(float)width/(float)height,
                       -nearDist, nearDist,
                        nearDist, 100.0);
   }
```

Part F: Light Reduction

The variable *dist* was modified in the vertex shader so the first light reduces with distance.

```
float dist = 0.01 + length(Lvec);
color = vec4(globalAmbient + ((ambient + diffuse) / dist);
```

Part G: Per Fragment Lighting

The lighting calculations were performed in the fragment shader so that the directions are calculated for individual fragments. Some of the global variables in the vertex shader were moved to the fragment shader.

Part H: Specular Highlights

Light Brightness and LightColor were added to the display callback function in order to pass them to the shader. These variables make the specular light tend to be white, which is calculated in the fragment shader.

In scene-start.cpp:

```
glUniform1f(glGetUniformLocation(shaderProgram, "LightBrightness"),
            lightObj1.brightness);
CheckError();
```

In fStart.glsl:

```
vec3 specular = Ks * LightBrightness * SpecularProduct;
float dist = 0.01 + length(Lvec);
color = vec4(globalAmbient + ((ambient + diffuse) / dist), 1.0);
color.a = 1.0;
gl_FragColor = (color * texture2D(texture, texCoord * 2.0)) + vec4((specular / dist), 1.0);
```

Part I: Second Light

The second light was added through modifying both the C++ code and the fragment shader. Some new variables (LightPosition2, LightColour2 and LightBrightness2) were added .The code added in scene-start.cpp is as follow:

void init(void)

```
addObject(55);
sceneObjs[2].loc = vec4(-2.0, 1.0, -1.0, 1.0);
sceneObjs[2].scale = 0.2;
sceneObjs[2].texId = 0;  // Plain texture
sceneObjs[2].brightness = 0.2;
```

void display(void)

```
SceneObject lightObj2 = sceneObjs[2];
vec4 lightPosition2 = rotationA * lightObj2.loc;
glUniform4fv(glGetUniformLocation(shaderProgram, "LightPosition2"),1, lightPosition2);
CheckError();
glUniform1f(glGetUniformLocation(shaderProgram, "LightBrightness2"), lightObj2.brightness);
CheckError();
glUniform3fv(glGetUniformLocation(shaderProgram, "LightColor2"), 1, lightObj2.rgb);
CheckError();
```

```
static void lightMenu(int id)
else if (id == 80)
   {
      toolObj = 2;
      setToolCallbacks(adjustLocXZ, camRotZ(),
                       adjustBrightnessY, mat2(1.0, 0.0, 0.0, 10.0));
   }
```

The second light is directional and its light calculations(ambient, diffuse and specular) are similar to the first light. The gl_FragColor was also modified:

```
gl_FragColor = (color * texture2D(texture, texCoord * 2.0)) + vec4((specular / dist) + specular2 ,
1.0);
```

Part J: Deletion, Duplication and Spot light

Deletion:

As every object has its own meshID, the idea of object deletion is to change the meshId and currObject to NULL and -1 separately. In addition, the deletion function and its ID were also added in the main menu.

```
static void deleteObject(int id) {
   sceneObjs[currObject].meshId = NULL;
   currObject = -1;
}
static void mainmenu(int id)
else if (id == 00 && currObject >= 0) {deleteObject(currObject);}

static void makeMenu()
glutAddMenuEntry("Delete object", 00);
```

Duplication:

By changing the *nObjects* to the ID of the object in sceneObjs [] and adding the same object similar to the addObject function, the object can be copied exactly the same.

```
static void duplicateObject(int id) {
   if (nObjects < maxObjects){ //THE MAX OBJECTS IS 1024
            sceneObjs[nObjects] = sceneObjs[id];
            toolObj = currObject = nObjects++;
            setToolCallbacks(adjustLocXZ, camRotZ(),
                             adjustScaleY, mat2(0.05, 0, 0, 10.0));
            glutPostRedisplay();
            }
}
```

```
static void mainmenu(int id)
else if (id == 01 && currObject > 0) {duplicateObject(currObject);}
```

```
static void makeMenu()
glutAddMenuEntry("Duplicate object", 01);
```

SpotLight:

This part is not yet complete. The idea to achieve this part is to add a spotlight to the scene by modifying the display () function and provide variables for the implementation of the spotlight. Then, the spotlight is modeled as a cone whose vertices, direction, angle and intensity all have variables. These allow interactively changing its lighting direction.