CITS3003 Project Part 1

Jake Nelson 20752958

May 29, 2017

1 Overview

The required functionality for this first part of the project has been implemented and tested thoroughly, and there are no known errors in our implementation. Raj Sembi and I worked on the main project tasks together and I implemented current object selection and object grouping for my individual functionality.

2 Implementation

2.a Camera Rotation

In order to enable camera rotation, the display function needed slight modifications. Correct camera rotation was achieved by adding appropriate X and Y rotations using the predefined camera angle variables.

2.b Object Rotation

Implementing object rotation was done by modifying the drawMesh function to account for the necessary x, Y and Z rotations utilising the angles array for each object. Additionally in order to perfectly match the sample solution then the X and Z rotations needed to be inverted.

2.c Lighting and Shine Adjustments

Functions were created to adjust the amount of ambient, specular and diffuse light and the degree of shininess, as well as a new menu item added within the MaterialMenu function.

```
static void adjustAmbientDiffuse(vec2 ambience) {
        // SceneObject *obj = &sceneObjs[toolObj];
        sceneObjs[toolObj].ambient = max(0.0f, sceneObjs[toolObj].ambient +
            ambience[0]);
        sceneObjs[toolObj].diffuse = max(0.0f, sceneObjs[toolObj].diffuse +
            ambience[1]);
```

```
}

static void adjustSpecularShine(vec2 shine) {
        // SceneObject *obj = &sceneObjs[toolObj];
        sceneObjs[toolObj].specular = max(0.0f, sceneObjs[toolObj].specular +
            shine[0]);
        sceneObjs[toolObj].shine = max(0.0f, sceneObjs[toolObj].shine + shine
            [1]);
}
```

2.d Closeup (Clipping)

The reshape callback was fixed and modified to allow for more "close up" views of objects. The near distance was quadrupled to increase the field of view of the camera, and the initial view distance was scaled similarly to accommodate.

2.e Reshape

The reshape function also required modification to scale the viewport similarly when the window is resized horizontally as when it is resized vertically. This was implemented by using the height-to-width ratio to scale the top and bottom planes when the width is less than the height.

2.f Light Reduction

The effective intensity of the first light source is reduced linearly as its distance increases.

2.g Lighting per Fragment

The lighting calculations were moved from the vertex shader to the fragment shader by transplanting the relevant functions and modifying slightly as needed to account for differences between the vertex and fragment shaders.

2.h Shine

In order to change the functionality of the specular lighting to tend toward white, then the brightness and color of the lighting sources are passed into the shaders separately. The specular component of the lighting then uses only the brightness component of the light source not the color, and is the last element applied so it is not impacted by the objects texture's color.

2.i Second Light

A second light was added to the scene by creating a new sphere object positioned adjacent to the first, and just behind. As the second light is required to be directional, the light source is set to be directional rather than positional. Lighting positions are separated and passed into the vertex shader individually, and the relevant lighting calculations duplicated.

2.j Current Object Selection

As my first additional functionality implemented I added the ability to be able to toggle currently selected object using the up and down keys. I chose to implement this functionality to be activated using the arrow keys rather than a menu item due to the lack of an easy naming convention for objects which would distinguish one object from another. If more than one object was created of the same model and with the same texture then the naming convention would become unclear and confuse the user.

The function was added through assigning a callback function for the special keys using and then using a switch statement for which of the keys was pressed.

```
glutSpecialFunc( special );
```

Within the switch function then if the current object can be incremented or decremented without decrementing to select the ground or one of the lights, or incrementing past the number of objects, then the current object and tool object is updated.

```
void special( int key, int x, int y )
{
    switch ( key ) {
    case GLUT_KEY_UP:
        if(currObject != nObjects - 1)
        {
          currObject++;
          toolObj = currObject;
        }
        break;
    case GLUT_KEY_DOWN:
        if(currObject != 3)
        {
          currObject--;
          toolObj = currObject;
        }
        break;
      }
}
```

2.k Object Grouping

As my second additional functionality I added the ability to group and ungroup all the objects in the scene which are not the ground or light sources. The functionality works

such that once grouped then moving and resizing will operate on all the objects at once. Currently rotation is not implemented once grouped, rotation continues to work just upon the individually selected object. This was a result of running out of time but would be an interesting future extension.

The ability to toggle grouping was added as a menu item, selectable from the main menu. There was felt to be no need currently to add an entire sub-menu as within the current implementation there are no options to select. The state of grouped or ungrouped is stored via a boolean variable, which is updated when the menu item is selected to be the opposite of its current state.

If grouped is set to false, then the location and resizing code operates in its initial manner. However a check for the if the grouped variable is true was added, and if it is set then the code to change the location and resize the object was duplicated and loops over all the objects in the scene which are not the ground or the light sources.

```
static void adjustLocXZ(vec2 xz)
{
    // Additional functionality, if grouped then move all objects
    if (grouped)
      for (int i = 3; i < nObjects; i++){
        sceneObjs[i].loc[0]+=xz[0]; sceneObjs[i].loc[2]+=xz[1];
      }
    else {
        sceneObjs[toolObj].loc[0]+=xz[0]; sceneObjs[toolObj].loc[2]+=xz[1];
    }
}

static void adjustScaleY(vec2 sy)
{
    // Additional functionality
    if (grouped)
      for (int i = 3; i < nObjects; i++){
        sceneObjs[i].scale+=sy[0]; sceneObjs[i].loc[1]+=sy[1];
      }
    else {
        sceneObjs[toolObj].scale+=sy[0]; sceneObjs[toolObj].loc[1]+=sy[1];
    }
}
```

3 Reflection

Overall I found the project was achievable, although at times quite tricky to implement or correctly figure out what to change. Having the source code for the project to build upon was both a positive and a negative, as it required a fair amount of time to become familiar with the code but then enabled faster completion of the project. The time given to complete the project was adequate and the resources, teaching materials and instruction supplied were sufficient to fulfill the requirements of the project.