CITS3003

Scene Editor Report

Joel Frewin — 21306458

October 2014

Program Functionality

Our implementation of the project code performs to the fullest functionality as specified in the project outline. We have successfully completed parts A through I as a group and have each completed both the individual and credit additions. In completing the tasks we only changed minimal amounts of code, meaning it will not have a detrimental effect on the programs original performance. After testing all parts of the program we observed little to no lagging and no bugs.

Building Process

Parts A and B involved adding missing code by extrapolating on existing code. The added camera rotation was done by multiplying the view by x rotation in the same way as the y rotation already implemented. The functionality for object rotation was also added in this same way in part B. this also resulted in fixing the plane orientation problem.
Part C was done by first observing the change in rbg values. The program changes the object attributes based on the mouse's changing x and y positions. This code was duplicated, replacing the rbg attributes with the ambient, diffuse, specular, and shine attributes of the object.
Part D was done by changing the nearDist value in the reshape function. Allowing the camera to become closer before the object was behind the camera.
Part E also involved the reshape function. We took the reshape code for the vertical component and reapplied it the horizontal reshaping when the width was smaller than the height.
We started with Part G in the vertex and fragment shaders, moving the calculations of the light vectors into the fragment shader. This was done by passing the parameters from the vertex shader to the fragment shader via in/out variables. By calculating light per fragment rather than per vertex, the light model is much more detailed and specific.
After this we moved on to part F, using the Lvec vector to create a distance variable from the light source to the fragment. This distance vector would then affect the intensity of light reaching the camera. Points with light further away are darker.
Originally in the code all of the light types were multiplied by the texture matrix, meaning brightness would subtend toward the texture colour. For Part H, In order to make specular light remain white we separated it from this multiplication and added specular source calculation in afterward. This resulted in glowing white objects when specular light was prominent.
For the final part of the group assignment we added a second light (directional) to the code and the fragment shader. We added a new light object in the main code, and passed its coordinate position to the fragment shader. In the fragment shader we defined the vector from a point to

the light source as being equal to the light position relative to the origin. We then repeated the lighting calculations as in the first light and added it to the original light variables. Each point now displays the sum of both lights.

Personal Contributions

My partner Lance Brockway and I decided the best way to learn the content was to each do the assignment separately in its entirety. Once we each did parts A through I we combined our code based on whoever did each part best (clarity, simplicity, functionality etc.). We then collaborated on the motion controls at described below, allowing multiple objects to move throughout the scene. For my individual extension I introduced a delete function, also described below. It is a simple addition which removes the current object. The final code contains the collective work of the group including Lances individual extension: a duplicate function.

Individual and Additional Functionality

The delete function allows the user to remove the current object in the program, object control is then given to the previous object. I first added a new glut menu item to the program called Delete, with a unique id. In the main menu when this command is recognised it performs the delete function. In the delete function the number of objects, current object and tool object ids are decreased by one.
Motion allows the user to give a path to the current object or light which it will move along over time. The available paths are: circle, square, zigzag, and none. This was done by first giving all objects three new attributes: rootLoc, startTime, and motion. rootLoc is the root location of the object, allowing the object to be moved and rotated while it is in motion. startTime is the time since the motion was added, allowing for circle calculations. And motion is an integer id referring to the type of motion.
Every frame, the display function calls the move function, which goes through every object and updates its current position based on its root position and how far along the path it should be at the current time.
This motion can be added to multiple objects at one time and runs smoothly on the machine

Project Reflection

The project was relatively straightforward, with a clear indication of what needed to be done. Each part looked at a different aspect of OpenGL and the program code and expanded my comprehension of it. The only criticism I can give is that there was a lack of initial instructions. The starting point gave a directory of files which took awhile to decipher and understand. Even something as simple as a readme document or some very basic step by step instructions to begin would have been helpful.