Adventures In Graphics Programming

Author: Austin Veino

After the first presentation, we set out to implement a flythrough within our Qt application. We thought this was going to be an easy task after Grinstein allowed the use of code from other groups. We investigated a few solutions for the flythrough and attempted to modify our code accordingly. These solutions were almost working with our existing code. Hours of debugging and attempting various fixes had gone by and nothing of a true "fly-through" nature was working. At the time we could look around but our movement stayed relative to the model. We decided to throw out the Qt code and try a flythrough implementation using straight OpenGL and shaders. It was thought that this might prove to be easier to implement and work with.

We loaded up Dan Brook's color cube code as a basis, and massaged it another group's flythrough implementation. Again, hours went by, but this time we succeeded. We modified their flythrough to fit our code and added some of our own code to smooth out the camera movement. At this point we had a color cube and some ground and we could fly around it. It also worked on Linux and Mac thanks to the use of creative preprocessor directives and makefile. Hurray for progress, now to add some "wow" factor. Lights were the next step for our cube. The code for the lights seemed simple enough and at one point we had a messed up color cube with the colors all sorts of inverted based on the base triangles. After a few hours of fiddling with the lights, we found out we needed normal vectors to get them to work correctly. This was considered pointless on our color cube since we wanted to be loading an outside model anyway. Therefore, we went forward with trying to load models, which would be able to have their normals provided to us or calculated with an object loading library.

We decided to use the assimp library to add model loading functionality. Documentation and tutorials for assimp are on the internet but it was tough to find code that was usable with what we wanted to do. Eventually, we found a tutorial that seemed promising. We decided we should try to compile and run the tutorial code just to see if it works. This started the rabbit hole of chasing down compiler errors and fixing linking errors. We actually learned a lot about linking errors and where these libraries are located. To make things more interesting we were trying to keep platform independence. We eventually got the tutorial to compile on Austin's Mac. We were extremely excited to run and build this tutorial because if this worked we had basis to start tweaking. Our dreams were shattered when a run time error occurred. Our shader language version was to low, great. So that tutorial is out the window. More internet browsing then and fiddling with our code. Hours later again, we had some thing show up on the screen. Except it wasn't the model we were hoping for. We fooled around with assimp for a while and nothing was working, so back to the drawing board.

We debated what to do now that assimp was not accomplishing what we wanted. We looked into what another group did to load objects but we didn't want to copy their code. We then stumbled onto lib3ds a library that parses .3ds objects. We didn't really want to go down another library rabbit hole, but it seemed useful. We actually found a tutorial using lib3ds and Qt, finally a ray of hope. We pulled up our old Qt code and

started merging the tutorial code. The first attempt at running the code proved promising as a shadow figure of the model we were loading appeared. And not to long after we had a program that could load any 3ds model and rotate it around apply lights and change view mode. We needed to fix some of the rotation controls and we added the ability to move the model around and zoom into it. We also added the ability to zoom in Ortho view by changing horizontal and vertical viewing dimensions, and overall, things were looking better.

Now that we can load models we thought we would attempt some textures because they would add some wow factor. We were actually able to get code to grab texture coordinates off the model but we weren't sure how to load a texture onto them. A couple more hours of Googling and code plugging we had a model with some very messed up textures. At this point we were exhausted and it had been around five days straight working on this graphics project. So we reverted back to our basic model loader without textures.