Stephen Cardone

CS-330 Computer Graphics and Visualization

Professor Whitworth

Final Project Reflection

This reflection will discuss the contents of my final project. It will contain three sections. First, I will discuss development choices, second, I will explain how to use my project, and third, I will explain custom functions in my project.

Development Choices:

I had to make many decisions during this project, regarding functionality, and detail. Throughout the project, I believe I made some good decisions, and there are some decisions I would go back and change if I could. As far as deciding what object to use, this was not difficult for me. The requirements specifically mentioned using simple kitchenware as an object. I have 2 trophies that are just painted cast-iron pans, and that was all I needed. The complexity of the object was not too extreme, and it had a golden, painted texture that allowed me to explore texturing in this project. And finally, It had enough sentimental value so I would be excited to work on it.

**Textures:** I was expecting texture to be very simple for my project since I would be using 1 golden texture, but I was naive. My first plan was to simply take photographs of my actual object and use the photograph for the texture. I ran into many issues with this. I had to zoom in quite far to avoid ridges and words stamped into the pan. Also, it was nearly impossible to get consistent lighting on my photographs, so when I would include the texture in my project, you would easily be able to identify various triangles. In the end, I went with golden foil texture from the internet that had consistent lighting, which I believe was a good choice. Even after choosing the texture, I was not happy how the pan appeared. Because I used “pie slice” triangles to generate the circular portion of my pan, each pan was repeating the same texture and I didn’t like it. I went through two iterations to resolve this. First, I decided to randomly generate points between 0-1 and use those to hopefully create enough complexity so the triangles would disappear. This was better, but because of the random aspect to it, it was possible to get some odd-looking stretched textures. I decided to attempt to calculate the points on a circle using trigonometry to identify what texture coordinates to use make the circle look exactly like the template, and this turned out to be the best option for me.

**Data structure selection:** I also ran into an issue with the method I chose to organize my triangle objects. Fairly early in the project, I made the decision to use a single growing vertices array object and each 3 vertices would be a triangle. The reason I chose this approach was because I had already decided that I was going to be identifying vertices programmatically for many (if not all) of my triangles. Because of this, it made sense to have a single object that I could keep appending to without having to worry about any other structures. I determined that the greatest challenge I would face was the need to give the pan thickness which meant connecting the bottom and top circles to have show the edges of the pan, and I had already come up with an algorithm to do this, so I was not concerned. Eventually I hit a point when I was adding a bit more detail, and I had to use the same points many times over. At this point I realized It would have been a better approach to use a vertices and indices structure to easily access previous points. Unfortunately, at this point, changing to this format would have been a complete rework, so I continued with my initial strategy.

**Lighting the sides:** I also had to make the choice about how I would light the tapered edges of the pan. The edges of the pan contain many triangles and each one is at a unique angle relative to the x,y,z axis. When considering the normal for these objects, I had a decision to make. I could look in to identifying each object’s normal individually, but this didn’t seem like a good approach to me. Using trigonometry, I was able to have the normal for the object change depending on where the triangle fell on the circle. Using this I was able to come up with a decent lighting scheme for this complicated shape. I do not believe it is perfect, and there are some obvious flaws if you look at the right angle, but I am happy with the state for this project given the time and complexity expectations. There were many more decisions I had to make during this project, but those are a few stand-out decisions that had significant impact on my project.

Navigating the project

Interacting with the final project is fairly straightforward, however some basic explanation is needed. When the project is running, you will see the pan object, the lighting souce, and the grey background. The project has 2 states: “Manual” and “Automated” movement. The state is determined by whether the mouse is on the window or not. If the mouse is not on the window, the project will self-animate wherever the user left the camera. If the mouse is returned to the screen, the user now has manual control of the camera orbiting the object and can move around as they please within the restrictions that are set in the code. The user can also press the “P” key to toggle bool flag between perspective and orthographic displays. Pressing “P” also logs that the change has occurred. To create the user interaction functionality, I used the following OpenGL api’s, and set their callbacks to my own code. glutPassiveMotionFunc(), glutEntryFunc(), glutKeyboardFunc(), glutKeyboardUpFunc().

Custom Functions

I used many custom functions in my project. From the beginning I had the mentality of allowing the project to be scalable (as I do with all code I write). I will not be going in to detail on all of my custom functions, but I will go into detail on 2 specific functions that are modular and reusable. The first function is a function called “generateCircleVerts()” The function’s purpose is to return the vertices of a triangulated circle object. Parameters are passed in to the function for location, radius, number of points etc. The function calculates how many triangles will need to be drawn, determines the theta (angle) of each triangle, and uses trigonometry to calculate the locations of the points at radius length away from the passed in center point. The vector object is then then returned to the calling codepath. The second function is called “drawConnectedCircles()”. It takes parameters of 2 circles that have the same number of points, and determines all the triangles needed to draw the “walls” between the points of corresponding points on each triangle. This was used to draw the walls of the pan. This function then appends the triangles to the vector object that is used to determine our array object for displaying the objects. These 2 functions could be extracted from this project and function in any other 3d animation graphics project that uses c++ with an x,y,z system. There are many other functions I defined in this project, but these two stood out. Some other functions are also modular, and some are unique to this project.

Conclusion

Overall, this project creates a 3d graphics rendering of a frying pan object with lighting, texturing, object/camera movement. I encountered and solved many issues when creating this project, and created modular, code to solve those problems. Working with computer graphics was a challenge, but I found that the deeper into it I got, the more fun it was. I am very happy to know how it functions, and would be happy to look deeper in to graphics in the future if the opportunity arises.