Final Project Report

Feature Implementation

I intended to implement shadow mapping for my project, and shadows has been successfully implemented now, with Poisson distribution to make edges of shadows look softer.

There are 3 light sources, blue, green, and magenta respectively. The blue light is slightly behind the camera and moving with it, the green one rotates around the objects, while the magenta light source is a still one. Each light source has its own shadow for the whole scene, and 3 shadows are composited and can overlap each other. This program can read in 3 .smf files consequently, and display 3 objects in a row. It is able to read any 3 .smf files, however I just preset the code only to read a cow, bunny with 5000 faces and a lo-sphere into the screen. There is a mat, a square with edge length of 10, located slightly below objects.

I mentioned light mapping in project description, however it is not implemented. I learned from several references and found that light mapping is intrinsically one special kind of 2-D texture mapping. The procedure of mapping a 2-D texture is very simple, however the process to create textures for lights can be quite annoying, since it can take a long time to approximate and test for the best composition of light color, position, and intensity for light sources. Instead, I just generated an interesting 2-D procedural texture and applied to the mat. Also, I implemented a better shadow model, which adapts Poisson distribution, to generate softer shadows to get higher level of scene reality.

Critical Techniques

Techniques used in this project are: 3D mesh model objects, phong shading, use of FBOs, 2-D procedural texture, shadow mapping, and vertex coordinate transformation.

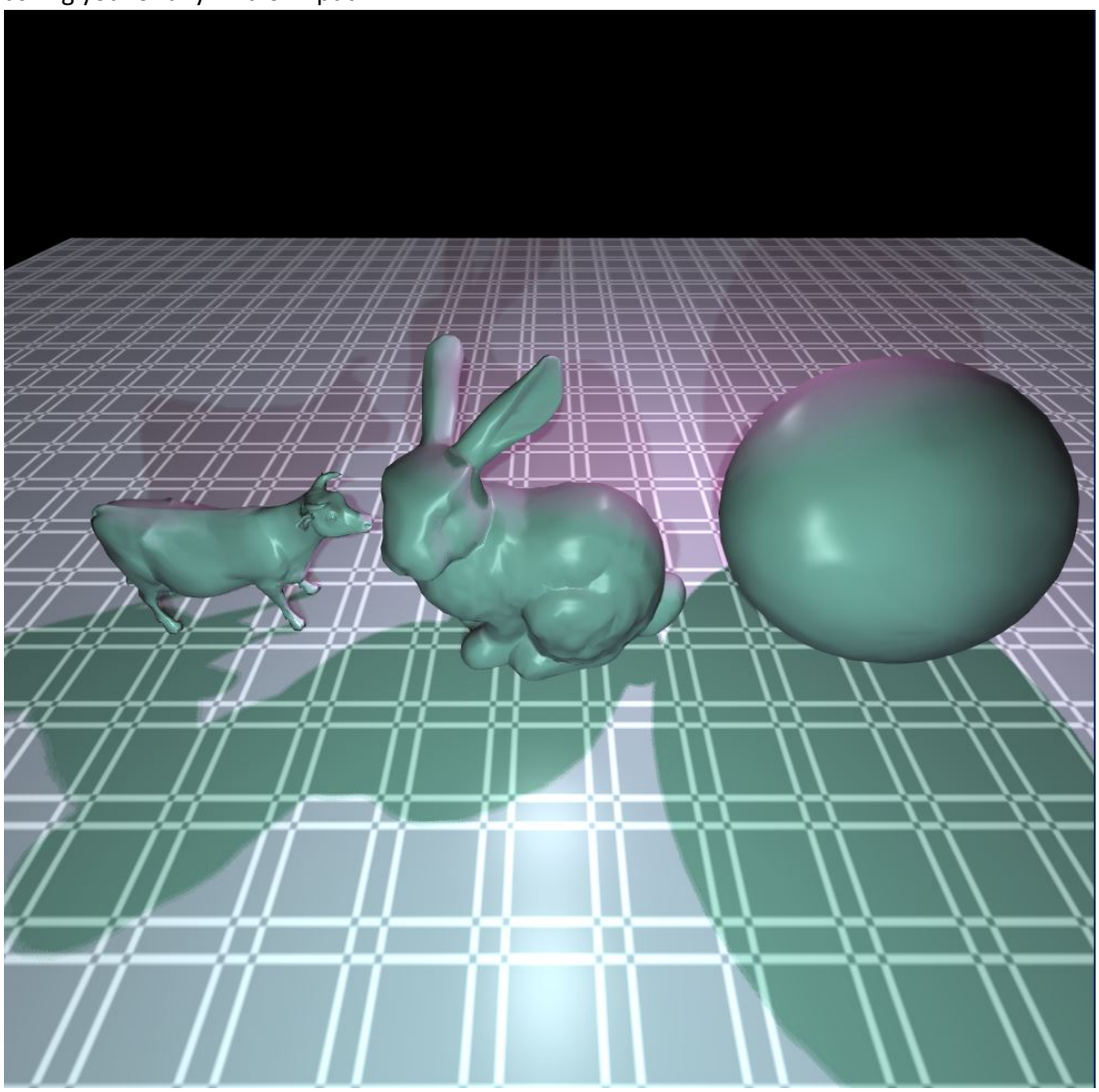
The most difficult part of shadow mapping is the coordinate conversion. I read several references and tutorials, and was confused by different ways for coordinate transformation and inconsistent procedure descriptions. It is straightforward to understand how to generate depth texture for each light source, and I have already been proficient in applying textures to objects using fragment shader after the texture mapping class. After nearly one-week trying different parameters and understanding the behind mathematical principles, I finally found the way to apply shadow texture correctly. Now I understand the role of bias matrix for coordinates in light source view, and I am able to create one and use it properly. I also get better understand homogenous coordinates by finding I actually need to divide first 3 coordinates of a 3-D vertex with the last coordinate.

I developed this program with Visual Studio initially, which worked perfectly fine. After moved codes to virtual machine, the shadows don’t work at all. Soon I realized the openGL version difference between the 2 platforms. I installed latest version of openGL on my own PC, while the

virtual machine has the earlier version, resulting a situation where built-in functions inputs and results are not always the same. So that my shadows are not displayed. I had to re-write the most part of the program to make sure it worked with virtual machine provided.

Image Output

After extracting the zip file and compiling the program, it reads “bound-cow.smf”, “bound-bunny_5k.smf”, and “bound-lo-sphere.smf”. The program generates following scene without asking you for any kind of input.



As the image above, the shadow in the front is of the still magenta light, shadow behind is of the light source behind the camera, while the shadow for rotating light source is not obvious right behind objects, but still be able to see its dark magenta outline. The outlines of shadows are quite soft, as a consequence of adaption of Poisson distribution. This screenshot is captured with my local Windows OS, you may find that MSAA is also implemented.

For this program, you can also change the view of camera, projection mode, material property for shading, and also the light position control of rotating light. You can get the full key board and

mouse menu instruction by reading the readme file inside the zip file.