CS6610 FINAL PROJECT REPORT
NON-PHOTOREALISTIC RENDERING TECHNIQUES

Varsha Alangar

NOTE:

A video of the output are available in the .zip folder. Extract the .zip folder and navigate to the "Documentation" folder to find .mp4 files of the output.

A copy of the class presentation is also available in the "Documentation" folder.

Implementation:

Intermediate implementation: (as mentioned in the intermediate report)

Cel shading involves shading the objects with lesser shading colors to provide a more comic and cartoon like look. It contains sharp change in color on the object. I used 4 distinct colors in this project.

For *silhouette creation*, I rendered an enlarged version of the object by shifting the vertices along its normal and applied a constant color to the vertices.

In addition to the above, I also performed *Contour detection*. Here are the steps I followed:

1. Generate the object, perform toon shading and silhouette creation as mentioned above.
2. Instead of rendering step 1 onto the screen directly, render it into a texture buffer.
3. Generate a plane and apply the rendered texture from the texture buffer onto the plane.
4. In the plane shader, while applying the texture to the plane, calculate the edge and contour: if there is a sharp change in color, it could be a change in depth, in which case, return vec3(0.0, 0.0, 0.0) // shading the contour black. Otherwise, return the color (could be the texture or just the plane) at the given pixel.

Final implementation:

Gooch shading is used to generate objects that look like technical illustrations. Two tone shading is used on the object to denote surface curvature. One of the tones is a "warm color" like yellow, orange and red to denote the parts of the object that are closer to the light source, and the other tone is usually a "cool color" like blue (or shades of blue) to denote parts of the object away from the light source.

Gooch shading involves two passes:

- First pass where the object is colored with the two tones.
- Second pass where the outline of the object is determined.

Formulae used:

$$K_{cool\_diff} = min(k_{cool} + k_{diffuse}, 1.0)$$
$$K_{warm\_diff} = min(k_{warm} + k_{diffuse}, 1.0)$$
$$K_{final} = mix(k_{cool\_diff}, k_{warm\_diff}, NdotL)$$
$$Color = min(k_{final} + k_{specular}, 1.0)$$

Hatching involves a group of strokes with spatially-coherent direction. It helps denote the material property and shape of the object. This is another form of Non-photorealistic rendering that generates an object with a pencil stroke shading like effect.

When the object is away from the light source, we use cross-hatching (non-parallel strokes) to achieve darkening of the object surface. For parts of the object that is close to the light source, parallel strokes of different distances apart are drawn.

Having generated textures for various levels of hatching (each texture representing the various levels of light intensity), we use these textures to color each pixel of the object. Given the intensity of light at a pixel, we choose the hatching texture that best suits that intensity and determine the texture to be applied at that pixel.

Using the implementations so far:

Left mouse button: Hold the left mouse button down and drag the cursor to change the camera angle (rotation) for the object.

Right mouse button: Hold the right mouse button down and drag the cursor to change the camera distance (zoom-in and zoom-out) for the object.

'T' or 't': Press this key to toggle between Toon shading and Gooch shading for the object.

'H' or 'h': Press this key to toggle between Hatching shading and Toon/Gooch shading for the object.

Project Progress:

The goal of the entire project was to perform Toon shading, Silhouette shading, Contour detection, Gooch shading and Hatching. I have been able to implement a basic version of all these techniques to the best of my abilities.

OS Used: Windows 10

IDE: Visual Studio 2013

Libraries and dependencies:

All the libraries used in the project are included under the *lib* folder within the zip file. They include: opengl32.lib, glu32.lib and freeglut.lib

All the header files are included within the GL folder contained in an include folder within the zip file. They have been included as *#include <GL/gl.h>*, *#include <GL/freeglut.h>, #include<GL/vmath.h>, #include<GL/glfw3.h>* in the code. All the header files from cyCodeBase recommended for use are included in the cyCodeBase folder in the same include folder that holds the GL folder.

All the DLLs required are placed in the Debug folder of the zip file. The source code itself was created and compiled in Visual Studio and is available as *main.cpp* in Final Project folder along with the solution.

The executable is available in the Debug folder.

Requirements to compile the project:

Unzip the project zip file and open the solution in Visual Studio.

In the properties of the project, link to the libs, dlls and header files. Make sure to choose “All configurations” in the properties window before adding the dependencies.

As mentioned earlier, all the required libraries, dlls and header files are available in the lib, Debug and include folders respectively of the zip file.

Add the *_CRT_SECURE_NO_WARNINGS;* to the preprocessor additional requirements in the project properties.

Please let me know if there is any issue in running the code.

Screenshot of output:

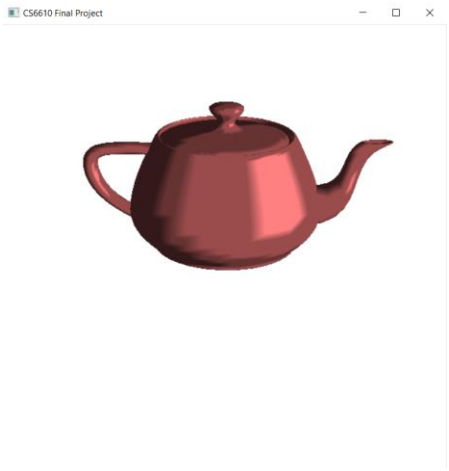


Figure 1: Teapot with toon shading

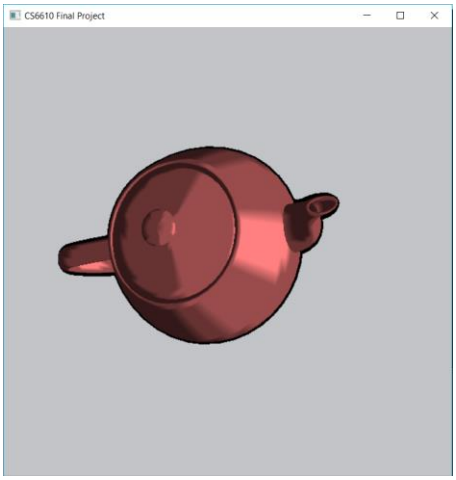


Figure 2: The toon shaded teapot with a silhouette

Figure 3: With toon shading, silhouette and contour

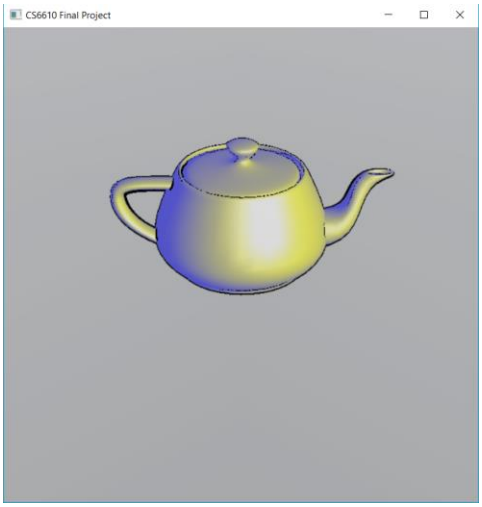


Figure 4: Gooch shading

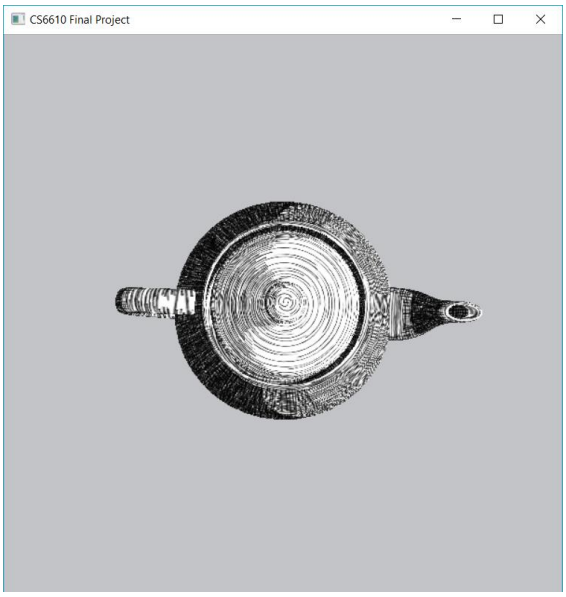


Figure 5: Teapot with Hatching

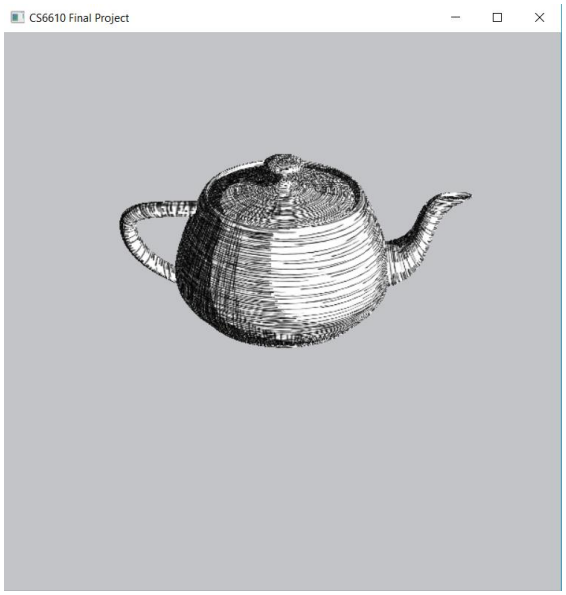


Figure 6: Teapot with Hatching

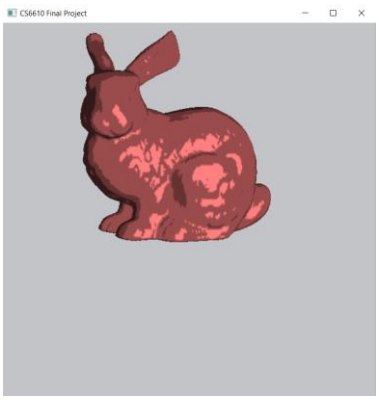


Figure 7: Toon shaded Stanford bunny without outline

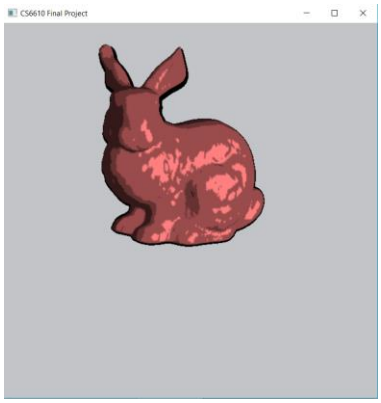


Figure 8: With silhouette

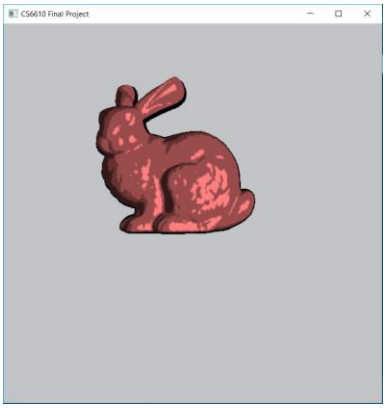


Figure 9: With silhouette (different camera angle)

Figure 10: Gooch shading on Stanford bunny

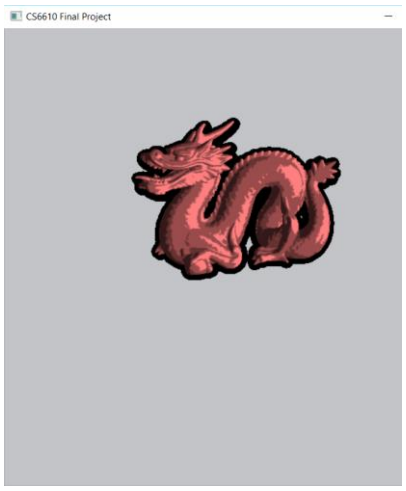


Figure 11: Stanford dragon with toon shading and silhouette

Figure 12: With toon shading and contour detection

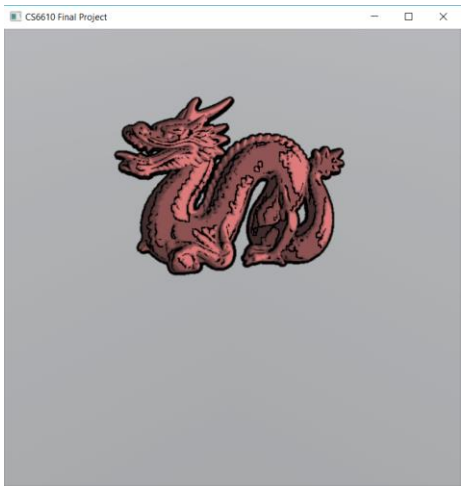
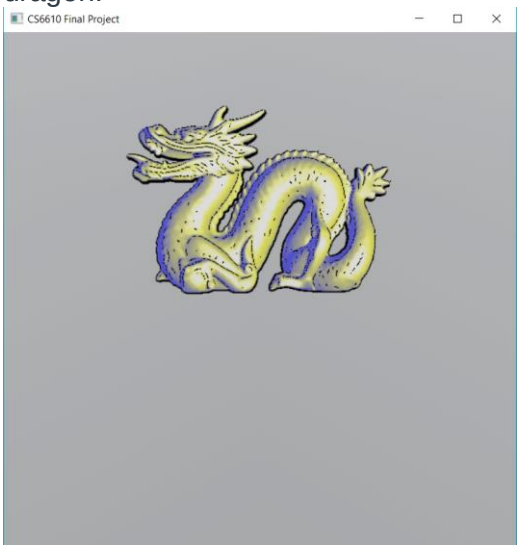


Figure 13: With both silhouette and contour detection

Gooch shading on Stanford dragon:



References:

[1] Artistic Rendering with Graphics Shaders by Lukas Lang; Eastern Michigan University, Department of Computer Science

[2] Non-Photorealistic Rendering with Pixel and Vertex Shaders; Drew Card, Jason L. Mitchell

[3] https://www.cs.rutgers.edu/~decarlo/contour.html

[4] Online tutorials such as http://www.lighthouse3d.com/tutorials/ and http://sunandblackcat.com/ were also helpful.

[5] GOOCH SHADING: A Non-Photorealistic Lighting Model For Automatic Technical Illustration, Amy Gooch Bruce Gooch Peter Shirley Elaine Cohen, Department of Computer Science University of Utah

[6] HATCHING: Real-Time Hatching, Emil Praun Hugues Hoppe Matthew Webb Adam Finkelstein

[7] http://bentonian.com/Lectures/AdvGraph1314/7.%20OpenGL%20and%20shaders%202.pdf