CS 3920/5920 - Computer Graphics Fall 2017

Program: 4 – Lighting, Shading, Shaders, etc.

Points: 20

Due Date: November 23

Grace Date: November 28 - demonstrated no later than 5:00 p.m

You will continue to work in your groups. You must put the project files in a Prog4 folder in your group S share. Sign up for a 15 minute demo time. No extra credit on this program.

In this program, you will be making significant modifications to Program 3 to bring it up to OpenGL 3.2 and later, with no allowed deprecated/removed functions. You will lose 2 points for each Deprecated OpenGL function used, up to a maximum of 10 points.

**Changes to Form1, the Figure class, the Axes class, and the VertexDataList class:**

1. Remove and Replace all GL methods such as MatrixMode, LoadMatrix, VertextPointer, ColorPointer, EnableClientState, etc. See opengl-quick-reference-card.pdf for a list of the deprecated/removed methods (shown in blue). I suggest that you use “Find in files” in Visual Studio to search for those functions to assure you have replaced them all.
2. Replace the items in #1 with GL methods such GetUniformLocation, UniformMatrix4, EnableVertexAttribArray, VertexAttribPointer, etc., as discussed in class. This time, the ModelView matrix is kept as two separate pieces, the ModelMatrix and the View Matrix, instead of combining them. The ModelMatrix is the aggregate of all the rotations, scalings, and translations on the figure. The ViewMatrix is the LookAt matrix. Therefore, these will be associated with the Uniform variables in two different places (Form1 and Figure class).
3. Add a FormClosing event handler to Form1 and use it to Unload the Shaders.
4. Finish the CalculateNormals method in the VertexDataList class.
5. Make changes to the Axes class changes similar to the above. Experiment with the shininess.
6. Add the following to the Figure class to get and set the shininess:

public float Shininess { get; set; }

You will use this to set the Shininess uniform variable in the Show method of Figure.

1. Have a way to set the Shininess for each figure. One way would be to add it as a parameter to the Load method in Figure and then have FigureList LoadFigures method specify different values for different figures (0.0 means not shiny). **You must have at least 2 shiny figures (assuming there are 2 in the folder) and at most 50% shiny figures (rounded down).**
2. For the Figure class, calculate the Normal matrix in Show using the inverse transpose of the calculated Modelview matrix (use both the Model matrix and the passed View matrix – multiply them together – but which way?). Matrix4 has methods for Invert and Transpose.
3. On the form, have a Groupbox for the Lighting Settings with the following capabilities:
   1. Set the X, Y, Z coordinates for the Light. I should be able to easy place it anywhere.
   2. Set the Light Color (e.g., like in Program 1 for the Line/Rectangle color and then change it to RGB to set LightColor. You must have at least 7 colors and 5 required colors are: White, Black, Red, Green, and Blue.
   3. A way to set GlobalAmbient from 0.0 to 1.0 in increments of 0.1

You will use these values in the Form1 ShowFigs to set the GlobalAmbient, LightPosition, and LightColor uniform variables. **Changing any of these should be immediately visible, even if the Figure Timer is “off”**. This allows us to move the light around with the figure stopped and see the effects.

**Loading Shaders:**

I have created the shell for a Singleton ShaderLoader class. It is an amalgamation with pieces taken from tutorials and other web sources. Comment it so I can tell that you understand what it does.

**Vertex Shader and Fragment Shader Programs:**

As a starter, use the examples from the “Using functions in shaders” in

<http://www.gamedev.net/page/resources/_/technical/opengl/the-basics-of-glsl-40-shaders-r2861>

This is similar to what is in the book on page 478 except the book uses many deprecated functions.

I have placed those in files in the Prog4 folder (called Prog4\_xS.glsl, where x = F or V).

I have placed the GLSL Spec: <https://www.opengl.org/registry/doc/GLSLangSpec.4.40.pdf> in the 1CourseMaterials folder.

Make the following changes to the Prog4\_VS.glsl:

1. Add an “in” for VertexColor. You can reorder the locations to match the order in the VertexData if you want but it isn’t necessary.
2. Change NormalMatrix to mat4 (makes it easier to set in the program) and make the corresponding change the first line in getEyeSpace:

norm = normalize(vec3(NormalMatrix \* vec4(VertexNormal,0.0)));

1. Remove the MVP and ModelView uniform variables
2. Remove the MaterialInfo and LightInfo structs and their corresponding uniform variables.
3. Add the following **uniform** variables:
   1. float Shininess
   2. float GlobalAmbient
   3. vec3 LightPosition (in world coordinates, not eye coordinates)
   4. vec3 LightColor
   5. mat4 ModelMatrix (will be aggregate of the figure rotations, scaling, translations)
   6. mat4 Viewmatrix (will be the Lookat matrix)
4. Modify the phongModel function using the uniforms from #5 and the following rules:
   1. Use ViewMatrix \* ModelMatrix instead of ModelViewMatrix
   2. Use ProjectionMatrix \* ViewMatrix \* ModelMatrix instead of MVP
   3. Use ViewMatrix \* vec4(LightPosition,1.0) instead of Light.Position since as per the comment in the file, Light.Position was in eye coordinates, and we pass Light.Position in World Coordinates. Therefore, we need to transform it to eye coordinates.
   4. Use VertexColor for the Material Ambient and Diffuse
   5. Set the vec3 ambient variable to vec3(GlobalAmbient) \* VertexColor
   6. Use LightColor for the Light Specular and Diffuse
   7. Use 0.0 for the Light Ambient (for us, the only ambient will be from GlobalAmbient). Note that this happens automatically when you do e)
   8. If Shininess > 0 (and sDotN > 0.0), use VertexColor for the Material Specular and Shininess for the Material Shininess. Otherwise, use vec3(0.0) for the specular.

For simplicity, you can assume the glsl files are in the same folder as the EXE and just use the file name when loading them. No hard-coded paths allowed. Call the LoadAndCompileShader method ShaderLoader in the Form1 Load event. Be sure to check if it loaded properly and if not, put up a MessageBox with **the** error.