INFO6014 – Graphics 1

Final Exam – Fall 2010

Instructor: Michael Feeney

The exam format:

- You may use any resources you feel are necessary to complete the exam, but you are to answer the questions on your own.

- The questions are ***not*** of equal weight. There are three (3) pages with eight (8) questions with a total of 100 marks (if my math is correct)

- The answers may be one or a combination of the following:

    - Short answer (in your own words)
    - Snippets of code
    - Complete running solutions

- CLEARLY indicate which answer goes to which question. My suggestion is that you place each answer in its own folder, named "Question_01", "Question_02" and so on (or something equally clear). Another option is to create a Visual Studio solution and add a number of projects – one per question – to it.

- Place any written answers into a Word, RTF, or text file. Again, *clearly* indicate which question you are answering.

- If you are combining answers (which is likely), please indicate this with a "readme" file or some note (not buried in the source code somewhere) indicating this.

- For applications: If it doesn't build and run; it's like you didn't answer it. I'll correct trivial, obvious problems, but you need to be sure that it compiles and/or runs.

- You have until Friday, December 17th at 11:59 pm to submit all your files to the appropriate drop box on Fanshawe Online; this is because of the grade deadline of December 20th. Note: The intention is that you have the exam for 24 hours.

- You can reach me:
    - In my "cubicle of destiny" most afternoons this week (I'm invigilating in the mornings)
    - On the office phone: (519) 452-4430 x4798
    - My cell 519-494-7569
    - Or at home 519-438-3300 (but if you call me in the middle of the night, I'll be *very* annoyed, and if one of my kids answer, you'll get a verbal smack down, possibly in French or German. Really.)
    - Or through e-mail (mfeeney@fanshawec.ca), of course.

Questions:

1. (25 marks) *Demonstrate* the difference between Gourad and Phong shading.

   Hint:
   - One is mainly implemented in the vertex shader (while it could be implemented in the fragment shader, there is no point)
   - The other can only be done in the fragment shader.

2. (5 marks) If you were worried about performance over quality of lighting, would you implement your lighting in the vertex or fragment shader? Why?

3. (10 marks) Your instructor tells you it's pointless to implement flat shading in the fragment shader? Is he correct, or has be been drinking too much beer and is talking "crazy talk," and why?

4. (10 marks) What client side (i.e. C++) code would you need to set the uniform variable:

   ```
   uniform float offsetValue;
   ```

   located in the vertex shader (assume that the shaders OpenGL 'name' is 17)?

5. (10 marks) Why would you run into problems setting this same variable in the fragment shader?

6. (5 marks) What happens to your model if you remove the specular component from the shader listed in the appendix (i.e. if you commented it out or something)?

7. (5 marks) What happens when you try to declare variables starting with "gl_" or try to assign values to these variables in a GLSL shader?

8. (30 marks) *Demonstrate* the difference between a spot light and another type of light (directional or point) by showing both lights illuminating the same scene.

   You may:

   - place the lights in such a way as to show the difference, or
   - you may have the lights turn on and off independently, or
   - you may have a single light that can be changed by the user

Appendix:

```
/* Vertex shader for vertex shading */

uniform float timeTick;
varying vec4 colour_out;

void main()
{
  /* Transforming the vertex */
  vec4 pos = gl_Vertex;
  pos.x += sin(timeTick) * 10.0;

  gl_Position = gl_ModelViewProjectionMatrix * pos;

  vec4 ambient;
  vec4 diffuse;
  vec4 specular;

  float f = 1.0;

  vec4 eyePosition = gl_ModelViewMatrix * gl_Vertex;
  vec4 eyeLightPos = gl_LightSource[0].position;

  vec3 N = normalize( gl_NormalMatrix * gl_Normal );
  vec3 L = normalize( eyeLightPos.xyz - eyePosition.xyz );
  vec3 E = -normalize( eyePosition.xyz );
  vec3 H = normalize( L + E );

  float Kd = max( dot( L, N ), 0.0 );

  float Ks = pow( max( dot(N, H), 0.0), gl_FrontMaterial.shininess );

  if ( dot( L, N ) < 0.0 ) f = 0.0;

  ambient = gl_FrontLightProduct[0].ambient;
  diffuse = Kd * gl_FrontLightProduct[0].diffuse;
  specular = f * Ks * gl_FrontLightProduct[0].specular;

  colour_out = ambient + diffuse + specular;
}
```

That’s it.