CSCI 4631/5631 Fall 2016 Final exam (due Tuesday, Dec. 6th)

Provide discussion for the following questions:

1) Given everything we've talked about this semester, why is it so important that polygons you push through the pipeline be "simple" and "convex"? (**10 points**)

2) How are normals assigned to vertices in:

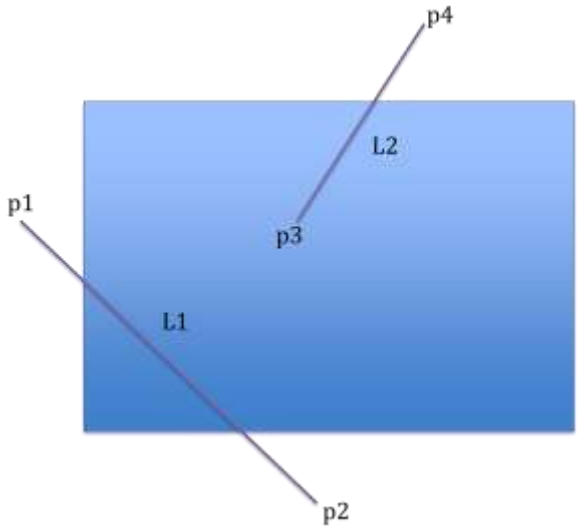
A) Flat shading? **(3 points)**
B) Gouraud shading? **(4 points)**
C) Phong shading? **(3 points)**

3) Line Drawing: Bresenham's algorithm was explicitly derived for lines with slopes between 0 and 1 in the first octant from point P_1 to point P_2, where the x value of P_1 is less than that for P_2. (**15 points total**)

A) In the incremental version of this algorithm, once we've drawn a pixel, we need to choose the next pixel to draw. What are our choices? (assume a line width of 1 pixel) Explain how we go about choosing which pixel to draw next. **(10 points)**

B) Why doesn't this algorithm, as discussed, work for lines with slopes greater than 1? How do we accomplish drawing these lines? **(5 points)**

4) Clipping: In Cohen-Sutherland clipping, we assign outcodes to the points on lines we draw that specify their orientation relative to the clipping window. What are the outcodes that would be assigned to the points to the right? **(2 points each)**



P1:
P2:
P3:
P4:

5) Light
5A) What are the 3 types of light reflection from surfaces in the Phong Lighting Model? (use a diagram if it aids your explanation) **(9 points)**

5B) What vectors are used to assign light intensities to surface points during the rasterization process in the Phong Lighting Model? (Draw a diagram) **(10 Points)**

5C) How does the Modified Phong Lighting Model (also known as Blinn-Phong) differ from the Phong lighting model? **(5 points)**

6) Shading **(15 points)** In what way did we "cheat" when we used smooth shading on our sphere approximation, why does this look better than a Gouraud shaded model, and why is this not necessary *generally applicable to any surface*? Discuss this and Gouraud shading **in detail**, and provide diagrams.

7) Hidden Surface Removal:

7a) How does the Z-buffer algorithm work? (**6 points**)
7b) How does the "painter's algorithm" work? (**6 points**)

8) Mapping : OpenGL likes the texel resolution of textures be some power of 2 in each direction because of mipmapping (this is not a totally stringent condition, but it is preferred) – what is mipmapping and what is it for? **(6 points)**

Bonuses (undergraduates, choose one. grad students, one is mandatory, the other will be for bonus points)

Bonus I (10 Points): Imagine a looking glass (mirror) in the Y-Z plane and facing in the direction of positive Z (in our standard, right-handed coordinate frame). Alice stands in front of it. What transformation matrix would reproduce the transformation her reflection would appear to have? Discuss what "kind" of transformation this is. If Alice actually stepped through the surface of the looking glass, what kind of transformation would this be? Discuss.

Bonus II (10 Points): How does GLSL code get "loaded" and "compiled"? Does the gcc compiler do it? Discuss.