CPSC 314
Quiz 2

March 17, 2006

Answer the questions in the spaces provided on the question sheets. If you run out of room for an answer, continue on the back of the page.

Name: _____

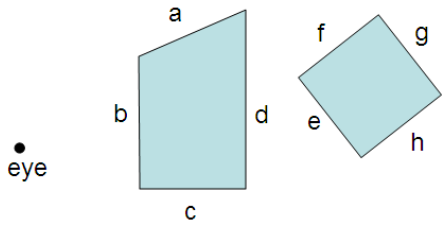
Student Number: _____

Question 1	/ 3
Question 2	/ 4
Question 3	/ 10
Question 4	/ 4
Question 5	/ 6
TOTAL	/ 27

This exam has 5 questions, for a total of 27 points.

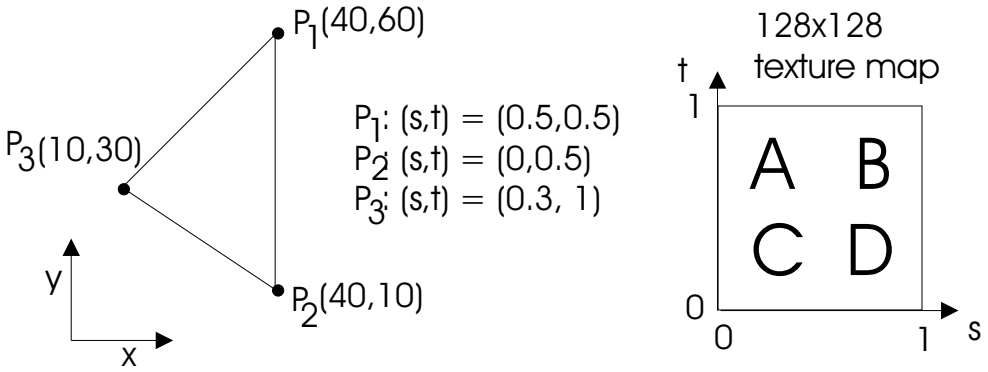
1. Visibility

   (a) (2 points) For the following scene, the polygons forming a closed solid object are represented by edges. Circle all the faces that would be removed by back-face culling. Show your work for borderline cases.



   (b) (1 point) Suppose you have a 10000 polygon Buddha statue that you need rendered. You would like to quickly determine if you can skip rendering the statue because it is not within the current field of view. Describe in some detail how you would do this. English or pseudocode is fine.

2. Texture Mapping



   (a) (2 points) A texture map with the letters "ABCD" is mapped onto the given triangle. Draw the texture map (i.e., the letters) as they would appear on the texure-mapped triangle.

   (b) (1 point) What are MIP-maps and why are they useful ?

   (c) (1 point) What are billboard textures useful for ?

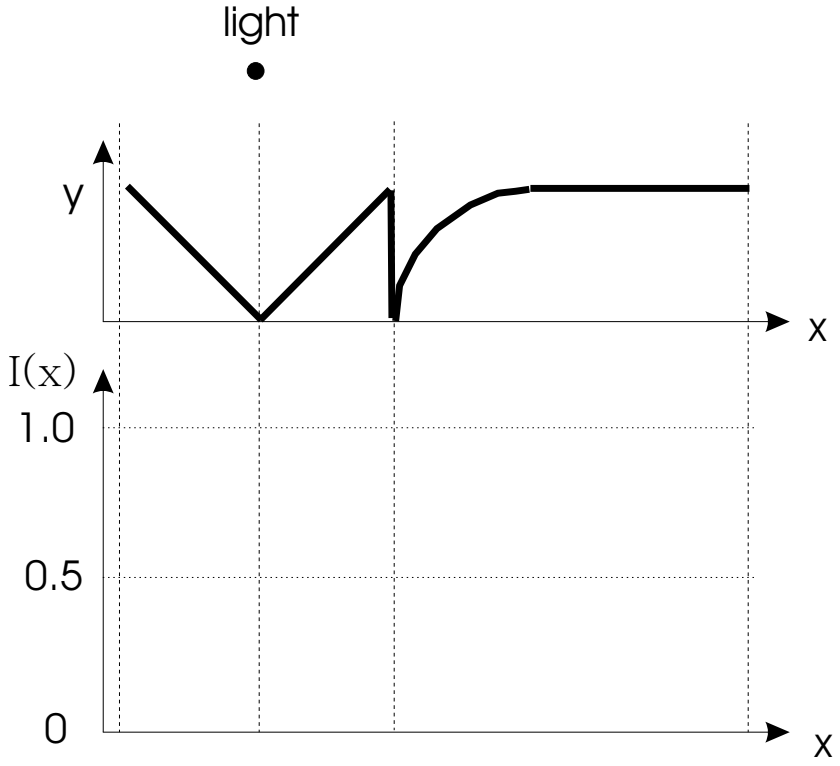
3. Implicit, Explicit, and Parametric Equations

   (a) (6 points) Points P_1, P_2, and P_3 define a triangle in 3D space. Show how to compute an implicit plane equation $F(x, y, z) = 0$, an explicit plane equation $z = f(x, y)$, and a parametric plane equation $P(s, t)$ for the plane defined by the triangle.

   (b) (2 points) Give the parametric equation for a line $P(t)$ in 3D space that passes through P_A at $t = 0$ and P_B at $t = 1$.

   (c) (2 points) Show how to compute the intersection point, $P_{intersect}$, between a line and a plane by substituting your answer from part (b) into an implicit plane equation. Assume a general implicit plane equation, i.e., your answer need not be in terms of points P_1, P_2, P_3.

4. (4 points) Sketch the ambient, diffuse, specular, and total illumination for the following scene as a function of x. Assume the Phong illumination model,
i.e., $I = k_a I_a + k_d I_d (N \cdot L) + k_s I_s (R \cdot V)^n$,
where $k_a = 0.2, k_d = 0.6, k_s = 0.6, I_a = I_d = I_s = 1.0, n = 300$.



5. (6 points) Barycentric Coordinates and Scan Conversion

   The following page gives pseudocode for the function
   `scanConvert(x1, y1, z1, x2, y2, z2, x3, y3, z3)`
   which sets all the pixels covered by the given triangle. The parameters give the three vertices in device coordinates, i.e., pixel coordinates.

   You need to do three things:

   - add the code to set the pixels when appropriate, including a z-buffer check;
   - add the code to do clipping
   - indicate how alpha, beta, and gamma could be computed more efficiently.

   You can add and change code wherever you like. You can assume that the z-buffer has been properly initialized. Below are the definitions of some of the functions and variables that are used. The barycentric coordinates are defines in terms of $P = \alpha P_1 + \beta P_2 + \gamma P_3$.

```
setPixel(x,y)      // sets the pixel
min(a,b, ... )     // returns the minimum value of all its arguments
max(a,b, ... )     // returns the maximum value of all its arguments
zBuf[x][y]         // 2D array storing z-buffer values
w                  // screen width, in pixels
h                  // screen height, in pixels
alpha(x,y)         // computes and returns alpha
beta(x,y)          // computes and returns beta
gamma(x,y)         // computes and returns gamma
```

```
scanConvert(x1,y1,z1, x2,y2,z2, x3,y3,z3)
{
  xmin = min(x1,x2,x3);
  xmax = max(x1,x2,x3);
  ymin = min(y1,y2,y3);
  ymax = max(y1,y2,y3);









  for (y=ymin; y<=ymax; y++) {
    for (x=xmin; x<=xmin; x++) {
      a = alpha(x,y);
      b = beta(x,y)
      c = gamma(x,y);









    }  // end for y
  } // end for x
}  // end of scanConvert()

alpha(x,y) {
  return ( A_alpha*x + B_alpha*y + C_alpha);
}

beta(x,y) {
  return ( A_beta*x + B_beta*y + C_beta);
}

gamma(x,y) {
  return ( A_gamma*x + B_gamma*y + C_gamma);
}
```