Homework #2

Projections, Hidden Surfaces, Shading,

Ray Tracing, and Texture Mapping

Prepared by: Jeremy Hammer, Austin Foxley, and Brian Curless

Assigned: Thursday, November 10th

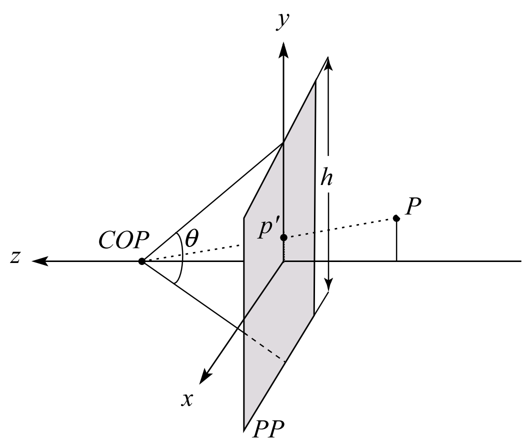
Due: Wednesday, November 23rd
at the beginning of class

Directions: Please provide short written answers to the questions in the space provided. If you require extra space, you may staple additional pages to the back of your assignment. Feel free to discuss the problems with classmates, but please ***answer the questions on your own*****.**

Name:____

Problem 1: Projections (14 Points)

In class, we derived the matrix for perspective projection for a viewer sitting at the origin looking down the $-z$ axis. Suppose now that we assume that the projection plane *PP* passes through the origin, and the viewer is at the *COP* somewhere on the $+z$ axis, still looking in the $-z$ direction, as illustrated in the figure below.



In addition, assume that we parameterize the projection in terms of viewing angle θ that measures the angle from top to bottom subtended by the image plane, which is of height h.

a) (8 Points) What is the projection matrix that we would use to map a point $P = [x\ y\ z\ 1]^T$ to $p' = [x'\ y'\ 1]^T$. (The "^{T}" symbol is the transpose operator, needed so that we continue to think in terms of column vectors. The w coordinate for p' will be 1 after doing the perspective divide.) Show your work.

Problem 1 (cont'd)

(Space for completing part a)

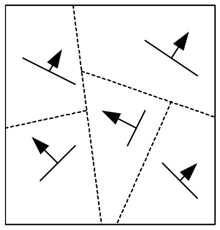
b) (4 Points) Solve for and write out the projection matrix when $\theta = 0$. Also solve for p' when $\theta = 0$. What kind of projection does this case correspond to?

c) (2 Points) What happens to p' as θ goes to 180°? Why?

Problem 2: BSP Trees (14 Points)

BSP trees are widely used in computer graphics. Many variations can be used to increase performance. The following questions deal with some of these variations.

a) (3 Points) For the version of BSP trees that we learned about in class, polygons in the scene (or more precisely, their supporting planes) were used to do the scene splitting. However, it is not necessary to use existing polygons – one can choose arbitrary planes to split the scene:



What is one advantage of being able to pick the plane used to divide the scene at each step? What is one disadvantage of not just using existing polygons?

b) (4 Points) Recall that when using a BSP tree as described in class, we must draw *all* the polygons in the tree. This is very inefficient, since many of these polygons will be completely outside of the view frustum. However, it is possible to store information at the internal nodes in a BSP tree that will allow us to easily determine if any of the polygons below that node will be visible. If none of the polygons in that sub-tree will be visible, we can completely ignore that branch of the tree.

Explain what extra information should be stored at the internal nodes to allow this, and how it would be used to do this "pruning" of the BSP tree.

Problem 2 (cont'd)

c) (4 Points) In class, we talked about doing a "back to front" traversal of a BSP tree. But it is sometimes preferable to do a "front to back" traversal of the tree, in which we draw polygons closer to the viewer before we draw the polygons farther away. (See part (d) for one reason why this is useful) How should the tree traversal order be changed in order to do a front to back traversal?

d) (3 Points) When we traverse a BSP tree in back to front order, we may draw over the same pixel location many times, which is inefficient since we would do a lot of "useless" shading computations. Assume we instead traverse the tree in front to back order. As we scan convert each polygon, we would like to be able to know whether or not each pixel of it will be visible in the final scene (and thus whether we need to compute shading information for that point). This is not a Z-buffer algorithm, and we're not going to keep Z-values around.

What simple information about the screen do we need to maintain in order to know if each pixel in the next polygon we draw will be visible or not?

Problem 3. Phong shading (10 Points)

The Phong shading model for a scene illuminated by global ambient light and a single directional light can be summarized by the following equation:

$$I_{phong} = k_e + k_a I_a + k_d L(\mathbf{N}\cdot\mathbf{L})_+ + k_s L(\mathbf{V}\cdot\mathbf{R})_+^{ns}$$

Imagine a scene with one white sphere illuminated by white global ambient light and a single white directional light. Describe the effect of the following conditions on the shading of the object. At each incremental step, assume that all the preceding steps have been applied first. (2 Points each)

a. (2 Points) The directional light is off. How does the shading vary over the surface of the object?

b. (2 Points) Now turn the directional light on. The specular reflection coefficient k_s of the material is zero, and the diffuse reflection coefficient k_d is non-zero. How does the shading vary over the surface of the object?

c. (2 Points) Now translate the sphere straight towards the viewer. What happens to the shading of the object?

d. (2 Points) Now increase the specular reflection coefficient k_s of the material to be greater than zero. What happens?

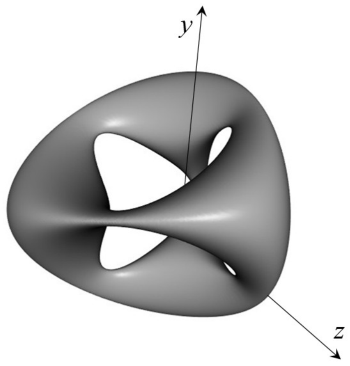
e. (2 Points) Now increase the specular exponent n_s. What happens?

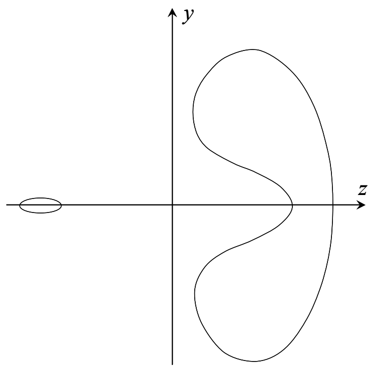
Problem 4. Ray tracing of implicit surfaces (25 points)

There are many ways to represent a surface. One way is to define a function of the form $f(x, y, z) = 0$. Such a function is called an *implicit surface* representation. For example, the equation $f(x, y, z) = x^2 + y^2 + z^2 - r^2 = 0$ defines a sphere of radius *r*. Suppose we wanted to ray trace a "quartic chair," described by the equation:

$$(x^2 + y^2 + z^2 - ak^2)^2 - b\left[(z-k)^2 - 2x^2\right]\left[(z+k)^2 - 2y^2\right] = 0$$

On the left is a picture of a quartic chair, and on the right is a slice through the *y-z* plane.





For this problem, we will assume a = 0.95, b = 0.8, and k = 5.

In the next problem steps, you will be asked to solve for and/or discuss ray intersections with this primitive. Performing the ray intersections will amount to solving for the roots of a polynomial, much as it did for sphere intersection. For your answers, you need to keep a few things in mind:

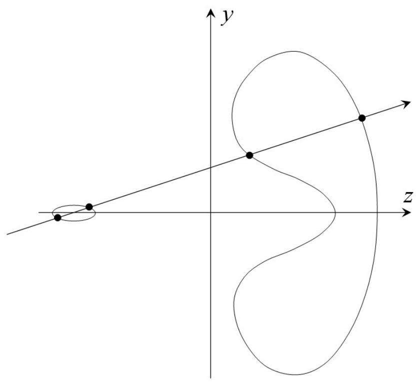
- You will find as many roots as the order (largest exponent) of the polynomial.
- You may find a mixture of real and complex roots. When we say complex here, we mean a number that has a non-zero imaginary component.
- All complex roots occur in complex conjugate pairs. If *A* + i*B* is a root, then so is *A* – i*B*.
- Sometimes a real root will appear more than once, i.e., has multiplicity > 1. Consider the case of sphere intersection, which we solve by computing the roots of a quadratic equation. A ray that intersects the sphere will usually have two distinct roots (each has multiplicity = 1) where the ray enters and leaves the sphere. If we were to take such a ray and translate it away from the center of the sphere, those roots get closer and closer together, until they merge into one root. They merge when the ray is tangent to the sphere. The result is one distinct real root with multiplicity = 2.

Problem 4 (cont'd)

a) (10 points) Consider the ray $P + t\mathbf{d}$, where $P = (0 \quad 0 \quad 0)$ and $\mathbf{d} = (0 \quad 0 \quad 1)$. Solve for all values of t where the ray intersects the quartic chair (including negative values of t). Which value of t represents the intersection we care about for ray tracing? In the process of solving for t, you will be computing the roots of a polynomial. How many distinct real roots do you find? How many of them have multiplicity > 1? How many complex roots do you find?

Problem 4 (cont'd)

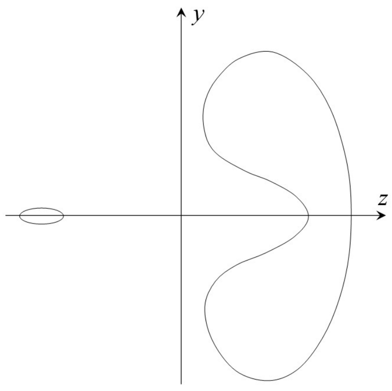
b) (15 points) What are all the possible combinations of roots, not counting the one in part (a)? For each combination, describe the 4 roots as in part (a), draw a ray in the *y*-*z* plane that gives rise to that combination, and place a dot at each intersection point. You may not need all of the given diagrams. The first one has already been filled in. (Note: not all conceivable combinations can be achieved on this particular quartic chair. For example, there is no ray that will give a root with multiplicity 4.)



# of distinct real roots: **4**

# of roots w/ multiplicity > 1: **0**

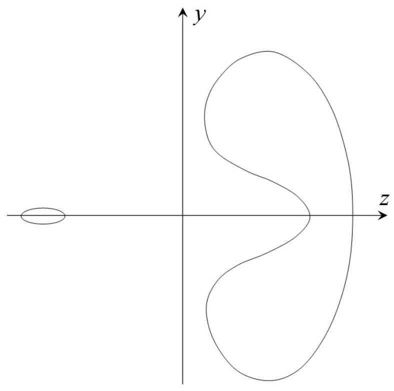
# of complex roots: **0**



# of distinct real roots:

# of roots w/ multiplicity > 1:

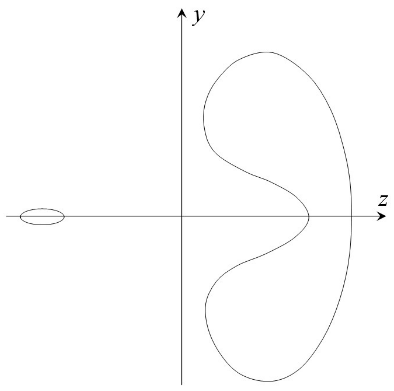
# of complex roots:



# of distinct real roots:

# of roots w/ multiplicity > 1:

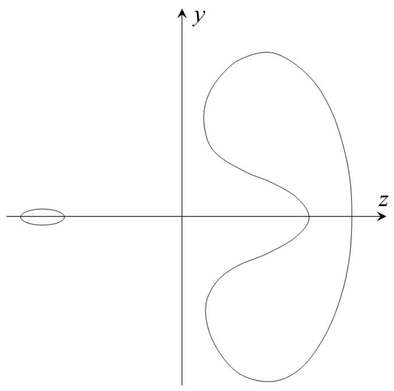
# of complex roots:



# of distinct real roots:

# of roots w/ multiplicity > 1:

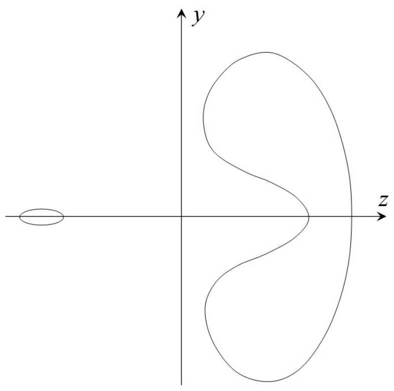
# of complex roots:



# of distinct real roots:

# of roots w/ multiplicity > 1:

# of complex roots:



# of distinct real roots:

# of roots w/ multiplicity > 1:

# of complex roots:

Problem 5: Barycentric and bilinear coordinates (22 Points)

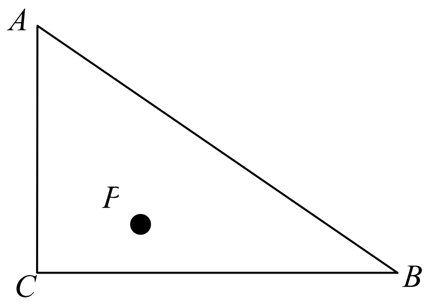
Once a point of intersection has been found on a triangle, it is useful to determine the point's coordinates within the triangle for various interpolating various vertex attributes (e.g., material coefficients, texture coordinates, normals, etc). Interpolation can be thought of as determining the "weight" of each of the vertices which contribute values to the final result at a given point. One method of interpolation is uses *barycentric coordinates*. Given a triangle with vertices *A*, *B*, and *C*, and an intersection point *P* inside the triangle, we can write *P* as a weighted sum of the vertices:

$$P = \alpha A + \beta B + \gamma C$$

where the barycentric coordinates (α, β, γ) are computed as:

$$\alpha = \frac{Area(BPC)}{Area(ABC)}, \beta = \frac{Area(APC)}{Area(ABC)}, \gamma = \frac{Area(APB)}{Area(ABC)}$$

Now consider the following right triangle and point of intersection *P*:

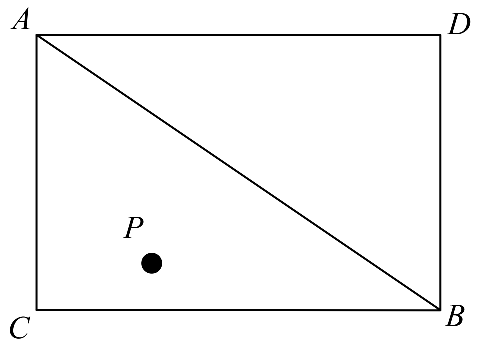


Suppose that the triangle and point of intersection lie in the *x-y* plane, with *A* = (0, 4), *B* = (6, 0), *C* = (0,0) and *P* = (2, 1).

a) (5 Points) Calculate the barycentric coordinates of *P* using cross products as discussed in lecture.

Problem 5 (Cont'd)

Suppose we add another vertex $D = (6, 4)$ to add one more triangle and to form a rectangle:



We can represent P as a weighted sum of all four vertices:

$$P = \alpha A + \beta B + \gamma C + \delta D$$

b) (3 Points) Thinking of P in terms of the barycentric coordinates of these triangles, what weight δ should we assign to D?

c) (4 Points) Suppose now we computed the weights on A, B, C, and D using bilinear interpolation. Solve for these bilinear weights of P.

Problem 5 (Cont'd)

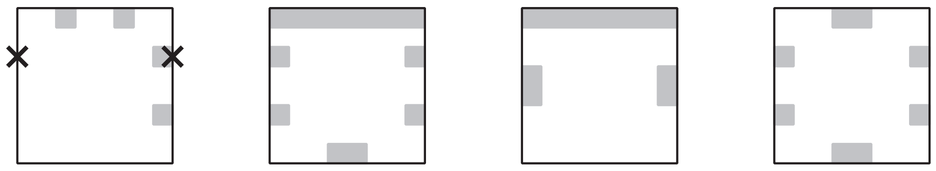
d) (10 Points) Suppose we are interpolating the coefficient k_d across the interior of the rectangle using barycentric and bilinear interpolation. Assume that k_d is a scalar (grayscale, not RGB) with value 1.0 at *C* and 0.0 at *A*, *B*, and *D*. Now, consider the interpolated values you would get when traveling along a straight line from vertex *C* to vertex *D*. Compute the interpolated value k_d you would get as function of the fractional distance, *s*, from *C* to *D* when using barycentric and bilinear interpolation. Plot the k_d functions for both cases. (Note: *s* varies from 0 to 1.)

Problem 6. Texture mapping (15 points)

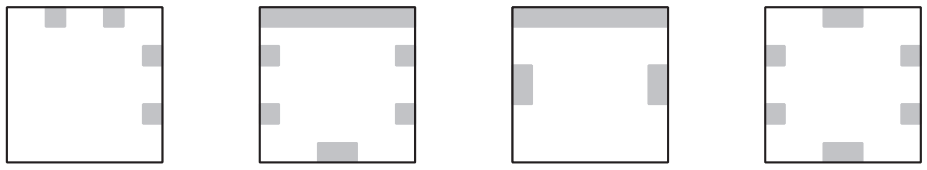
When a texture map is applied to a surface, points that are distinct in the rectangular texture map may be mapped to the same place on the object. For example, when a texture map is applied to a cylinder, the left and right edges of the texture map are mapped to the same place. We will call a mapping "allowed" if it does not map two points of different colors to the same point on the object to be texture mapped. For each of the combinations below, state whether the mapping is allowed (write yes or no). If it is not, mark two points on the texture with an X that are different colors, but map to the same point on the object. The first texture map for a cylinder primitive is done as an example.

Uncapped cylinder (top and bottom are open):

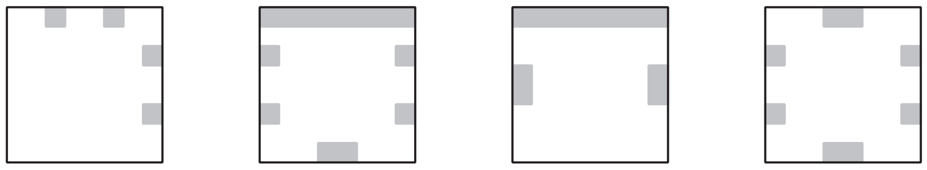
No



Sphere:



Uncapped cone (bottom is open):



Torus: