Homework #2

Shading, Projections, Texture Mapping,

Ray Tracing, and Bezier Curves

Assigned: Friday, May 8th

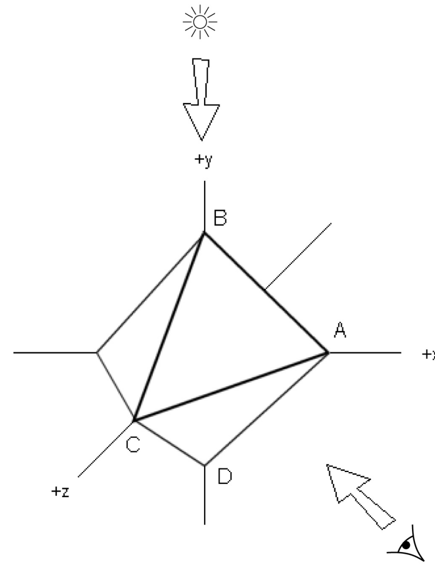
Due: Thursday, May 21st
at the beginning of class

Directions: Please provide short written answers to the following questions on your own paper. Feel free to discuss the problems with classmates, but please ***answer the questions on your own and show your work.***

Please write your name on your assignment!

Problem 1. Interpolated shading (19 points)

The faceted polyhedron shown in the figure at right is an octahedron and consists of two pyramids connected at the base comprised of a total of 8 equilateral triangular faces with vertices at (1,0,0), (0,1,0), (0,0,1), (-1,0,0), (0,-1,0), and (0,0,-1). The viewer is at infinity (i.e., views the scene under parallel projection) looking in the (-1,0,-1) direction, and the scene is lit by directional light shining down from above parallel to the y-axis with intensity $I_L = (1,1,1)$. The octahedron's materials have both diffuse and specular components, but no ambient or emissive components. The Blinn-Phong shading equation thus reduces to:



$$I = I_L B\left[k_d\left(\mathbf{N}\cdot\mathbf{L}\right) + k_s\left(\mathbf{N}\cdot\mathbf{H}\right)_+^{n_s}\right]$$

where

$$B = \begin{cases} 1 & \text{if } \mathbf{N}\cdot\mathbf{L} > 0 \\ 0 & \text{if } \mathbf{N}\cdot\mathbf{L} \le 0 \end{cases}$$

For this problem, $k_d = k_s = (0.5, 0.5, 0.5)$ and n_s=40.

(a) (2 points) In order to draw the faces as flat-shaded triangles, we must shade them using only their face normals. In OpenGL, this could be accomplished by specifying the vertex normals as equal to the face normals. (The same vertex would get specified multiple times, once per triangle with the same coordinates but different normal each time.) What is the unit normal for triangle ABC?

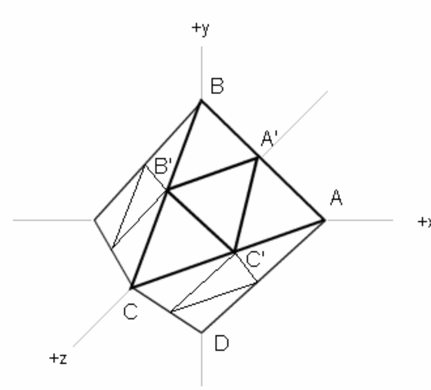
(b) (3 points) Assume that this object is really just a crude approximation of a sphere (e.g., perhaps you are using the octahedron to represent the sphere because your graphics card is slow). If you want to shade the octahedron so that it approximates the shading of a sphere, what would you specify as the unit normal at each vertex of triangle ABC?

(c) (5 points) Given the normals in (b), compute the rendered colors of vertices A, B, and C. Show your work.

(d) (2 points) Again given the normals in (b), describe the appearance of triangle ABC as seen by the viewer using Gouraud interpolation.

(e) (2 points) Now switch from Gouraud-interpolated shading to Phong-interpolated shading. How will the appearance of triangle ABC change (given the normals in (b)?

(f) (3 points) Remember that this object is being used to simulate a sphere. One simple improvement to the geometry of the model is to subdivide each triangular face into four new equilateral triangle (sometimes called 4-to-1 triangular subdivision), and then move the newly inserted vertices to better approximate the sphere's shape. If you subdivided triangle ABC this way, as shown in the figure to the right, what would be the best choices for the new coordinates and unit normals of the three added vertices A', B', and C' in order to more closely approximate a unit sphere?

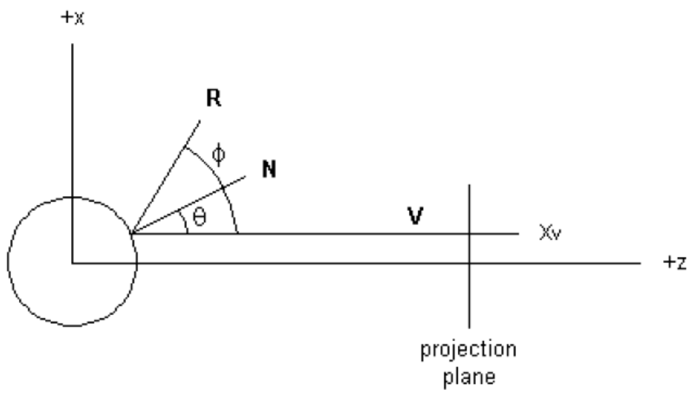


(g) (2 points) If you continued this subdivision process – repeatedly performing 4-to-1 subdivision, repositioning the inserted vertices, and computing their ideal normals – would the Gouraud-interpolated and Phong-interpolated renderings of the refined shape converge toward the same answer, i.e., the appearance of a ray traced sphere? Explain.

Problem 2. Environment mapping (20 points)

One method of environment mapping (reflection mapping) involves using a "gazing ball" to capture an image of the surroundings. The idea is to place a chrome sphere in a real environment, take a photograph of the sphere, and use the resulting image as an environment map. Each pixel that "sees" a point on the chrome sphere maps to a ray with direction determined by the reflection through the chrome sphere; the pixel records the color of a point in the surroundings along that reflected direction. You can turn this around and construct a lookup table that maps each reflection direction to a color. This table is the environment map, sometimes called a reflection map.

Let's examine this in two dimensions, using a "gazing circle" to capture the environment around a point. Below is a diagram of the setup. In order to keep the intersection and angle calculations simple, we will assume that each viewing ray **V** that is cast through the projection plane to the gazing circle is parallel to the z-axis. The circle is of radius 1, centered at the origin.



a) (5 points) If the x-coordinate of the view ray is x_V, what are the (x,z) coordinates of the point at which the ray intersects the circle? What is the unit normal vector at this point?

b) (3 points) What is the angle between the view ray **V** and the normal **N** as a function of x_V? Note that we will treat this as a "signed angle." In the figure above, the angle θ between **V** and **N** is positive. If the viewing ray hit the lower half of the circle (x_V is negative), then θ would be negative.

c) (5 points) Note that the (signed) angle φ between the view ray **V** and the reflection direction **R** is equal to 2θ, where θ is the angle between **V** and the normal **N**. Plot φ versus x_V. In what regions of the image do small changes in the x_V coordinate result in large changes in the reflection direction?

d) (4 points) We can now use the photograph of the chrome circle to build an environment map (for a 2D world); we store an array of colors drawn from the photograph, regularly sampled across reflection angles. When ray tracing a new chrome object, we compute the mirror reflection angle when a ray intersects the object, and then just look up the color from the environment map. (If the computed reflection angle lands between angles stored in the environment map, then you can use linear interpolation to get the desired color.) Would we expect to get exactly the same rendering as if we had placed the object into the original environment we photographed? Why or why not? In answering the question, you can neglect viewing rays that do not hit the object, assume that the new object is not itself a chrome circle, and assume that the original environment is some finite distance from the chrome circle that was originally photographed.

e) (3 points) Suppose you lightly sanded the chrome circle before photographing it, so that the surface was just a little rough.

- What would the photograph of the circle look like now, compared to how it looked before roughening its surface?
- If you used this image as an environment map around an object, what kind of material would the object seem to made of?
- If you did not want to actually roughen the object, what kind of image filter might you apply to the image of the original chrome circle to approximate this effect?

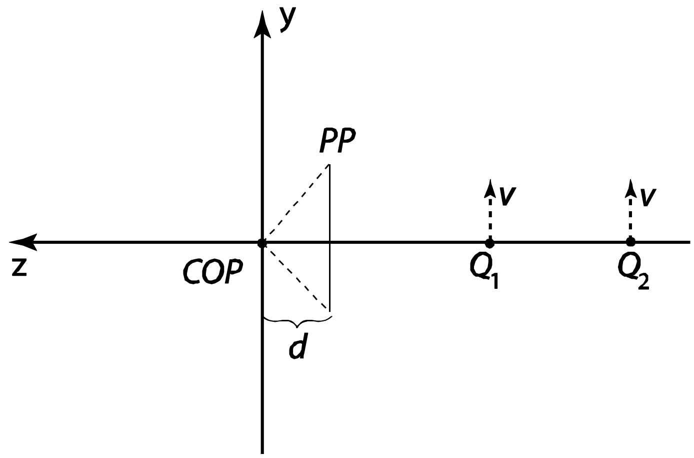
Problem 3: Projections (18 points)

The apparent motion of objects in a scene can be a strong cue for determining how far away they are. In this problem, we will consider the projected motion of points and line segments and their apparent velocities as a function of initial depths.

a) (6 points) Consider the projections of two points, Q_1 and Q_2, on the projection plane *PP*, shown below. Q_1 and Q_2 are described in the equations below. They are moving parallel to the projection plane, in the positive *y*-direction with speed *v*.

$$Q_1(t) = \begin{bmatrix} 0 \\ vt \\ z_1 \\ 1 \end{bmatrix} \qquad Q_2(t) = \begin{bmatrix} 0 \\ vt \\ z_2 \\ 1 \end{bmatrix}$$

$$0 > z_1 > z_2$$



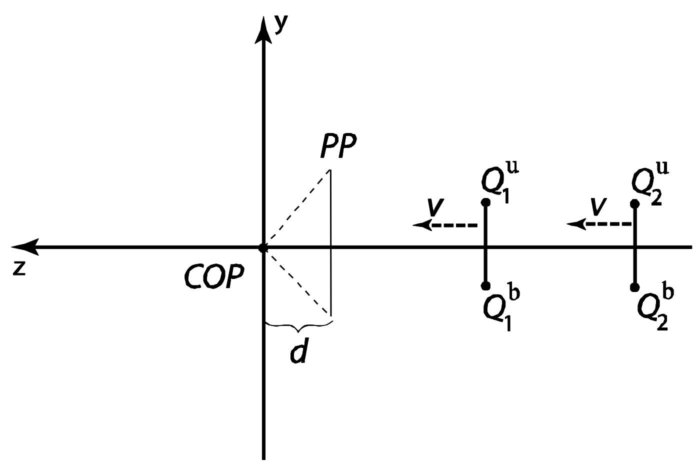
Compute the projections q_1 and q_2 of points Q_1 and Q_2, respectively. Then, compute the velocities, dq_1/dt and dq_2/dt, of each projected point in the image plane. Which appears to move faster? Show your work.

b) (8 points) Consider the projections of two vertical line segments, S_1 and S_2, on the projection plane *PP*, shown below. S_1 has endpoints, Q_1^u and Q_1^b. S_2 has endpoints, Q_2^u and Q_2^b. The line segments are moving perpendicular to the projection plane in the positive *z*-direction with speed *v*.

$$Q_1^u(t) = \begin{bmatrix} 0 \\ 1 \\ z_1 + vt \\ 1 \end{bmatrix} \qquad Q_2^u(t) = \begin{bmatrix} 0 \\ 1 \\ z_2 + vt \\ 1 \end{bmatrix}$$

$$Q_1^b(t) = \begin{bmatrix} 0 \\ -1 \\ z_1 + vt \\ 1 \end{bmatrix} \qquad Q_2^b(t) = \begin{bmatrix} 0 \\ -1 \\ z_2 + vt \\ 1 \end{bmatrix}$$

$$0 > z_1 > z_2$$



Compute the projected lengths, l_1 and l_2, of the line segments. Then, compute the rates of change, dl_1/dt and dl_2/dt, of these projected lengths. Are they growing or shrinking? Which projected line segment is changing length faster? Show your work.

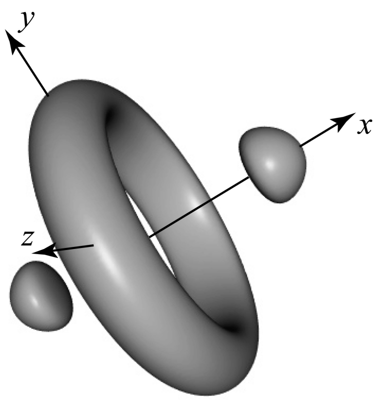
f) (4 points) Suppose now we replace the perspective camera in (a) and (b) with an orthographic camera. Which point, if any, in (a) would appear to move faster? Will the line segments in (b) appear to grow or shrink, and if so, which would change faster? Justify your answers in words or with equations.

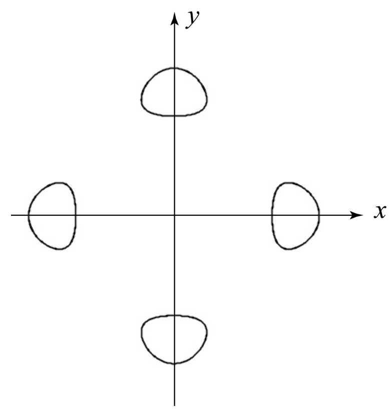
Problem 4. Ray intersection with implicit surfaces (18 points)

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 so-called "gumdrop torus," described by the equation:

$$4x^4 + 4y^4 + 4z^4 + 17x^2y^2 + 17x^2z^2 + 8y^2z^2 - 20x^2 - 20y^2 - 20z^2 + 17 = 0$$

On the left is a picture of a gumdrop torus, and on the right is a slice through the *x-y* plane.





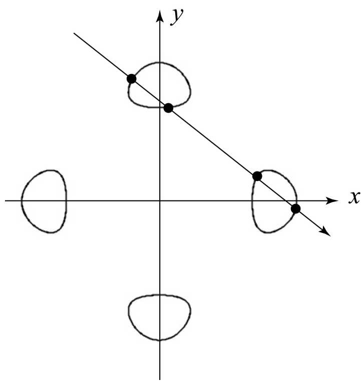
In the next problem steps, you will be asked to solve for and/or discuss ray intersections with this primitive. Performing the ray intersections will amount to solving for the roots of a polynomial, much as it did for sphere intersection. For your answers, you need to keep a few things in mind:

- 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 $A + iB$ is a root, then so is $A - iB$.
- 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.

a) (8 points) Consider the ray $P + t\mathbf{d}$, where $P = (0 \quad 0 \quad 0)$ and $\mathbf{d} = (0 \quad 1 \quad 0)$.
   - Solve for all values of *t* where the ray intersects the gumdrop torus (**including** any negative values of *t*). Show your work.
   - In the process of solving for t, you should have computed the roots of a polynomial. How many distinct real roots did you find? How many of them have multiplicity > 1? How many complex roots did you find?
   - Which value of *t* represents the intersection we care about for ray tracing?

Problem 4 (cont'd)

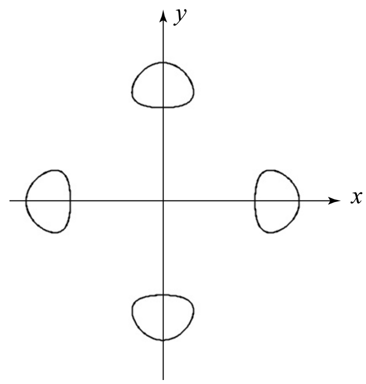
b) (10 points) What are all the possible combinations of roots, not counting the one in part (a)? For each combination, describe the 4 roots as in part (a), draw a ray in the *x-y* plane that gives rise to that combination, and place a dot at each intersection point. Assume the origin of the ray is outside of the bounding box of the object. The first diagram below has already been filled in. There are five diagrams that have not been filled in. You may not need all five. (Note: not all conceivable combinations can be achieved on this particular gumdrop torus. For example, there is no ray that will give a root with multiplicity 4.) ***Please write on this page and include it with your homework solutions. You do not need to justify your answers.***



# of distinct real roots: **4**

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

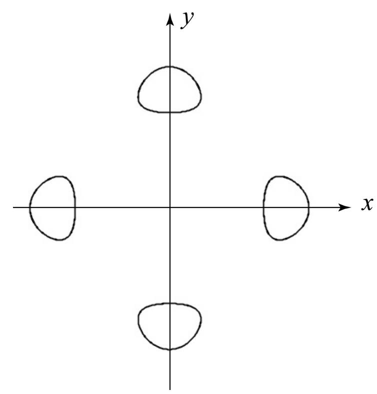
# of complex roots: **0**



# of distinct real roots:

# of roots w/ multiplicity > 1:

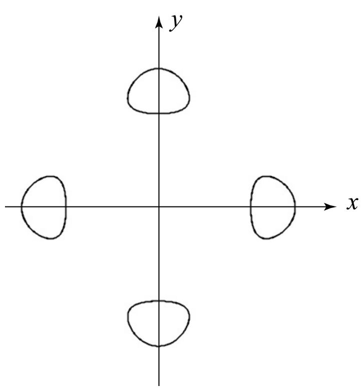
# of complex roots:



# of distinct real roots:

# of roots w/ multiplicity > 1:

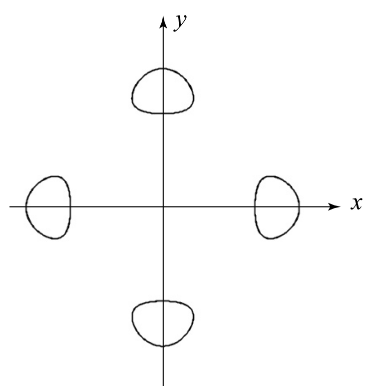
# of complex roots:



# of distinct real roots:

# of roots w/ multiplicity > 1:

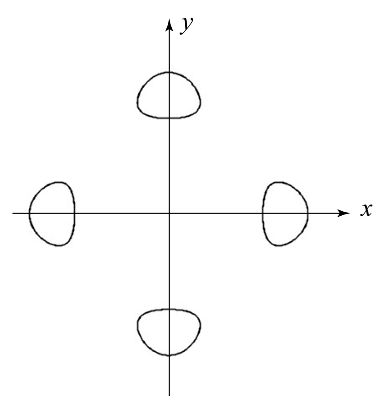
# of complex roots:



# 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. Bezier splines (25 points)

Consider a Bezier curve segment defined by three control points V_0, V_1, and V_2.

a) (3 points) What is the polynomial form of this curve, when written out in the form $Q(u) = A_n u^n + A_{n-1} u^{n-1} + \ldots + A_0$, where n is determined by the number of control points. The coefficients $A_0, \ldots, A_n$ should be substituted in the polynomial equation with expressions that depend on the control points V_0, V_1, and V_2. You may start with recursive subdivision or with the summation over Bernstein polynomials provided in lecture. Either way, show your work.

b) (3 points) What is the first derivative of $Q(u)$ evaluated at $u = 0$ and at $u = 1$ (i.e., what are $Q'(0)$ and $Q'(1)$)? Show your work.

c) (3 points) What is the second derivative of $Q(u)$ evaluated at $u = 0$ and at $u = 1$ (i.e., what are $Q''(0)$ and $Q''(1)$)? Show your work.

d) (5 points) To create a spline curve, we can stitch together consecutive Bezier curves. In this problem, we can add control points W_0, W_1, and W_2. What constraints must be placed on W_0, W_1, and/or W_2 so that, when combined with V_0, V_1, and V_2, the resulting spline curve is C^1 continuous at the joint between the Bezier segments? Write out equations for W_0, W_1, and/or W_2 in terms of V_0, V_1, and/or V_2. (It may be that not all of the W control points are constrained, in which case you would have fewer than three equations.) Show your work. Draw a copy of the control polygon below (shown at the bottom of the page) and place all constrained vertices exactly, and unconstrained vertices wherever you like, and then sketch the spline curve.

e) (5 points) Suppose we wanted to make the spline curve C^2 continuous at the joint between the Bezier segments. Now what constraints must be placed on W_0, W_1, and W_2? Write out equations for W_0, W_1, and/or W_2 in terms of V_0, V_1, and/or V_2. (It may be that not all of the W control points are constrained, in which case you would have fewer than three equations.) Show your work. Draw a copy of the control polygon below (shown at the bottom of the page) and place all constrained vertices exactly, and unconstrained vertices wherever you like, and then sketch the spline curve.

f) (3 points) Is it possible to achieve C^3 continuity with this spline? Explain.

g) (3 points) Suppose again that the control points are in two dimensions, but now $V_1 = V_2 = W_0 = W_1$. Think of this as sliding V_2 over on top of V_1 in the figure below, then placing W_0 and W_1 on top of those points, and then adding W_2 at some arbitrary position, somewhere to the right but not collinear with V_0 and V_1. Sketch the resulting curve. Will this curve be C^1? Justify your answer.