Homework #2

Projections, Hidden Surfaces, Shading, Ray Tracing, and Texture Mapping

Assigned: Thursday, November 9th

Due: Wednesday, November 22nd
at the beginning of class

Directions: Please provide short written answers to the following questions, using this page as a cover sheet. Be sure to justify your answers when requested. Feel free to discuss the problems with classmates, but please ***answer the questions on your own*.**

Please attach this cover sheet to your homework solution.

Name:____

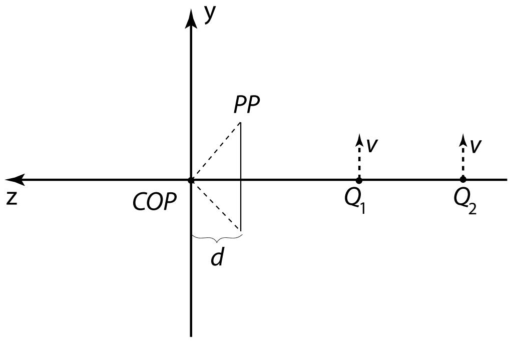
Problem 1: Projections (20 points)

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) (8 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*.

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

$$0 > z_1 > z_2$$



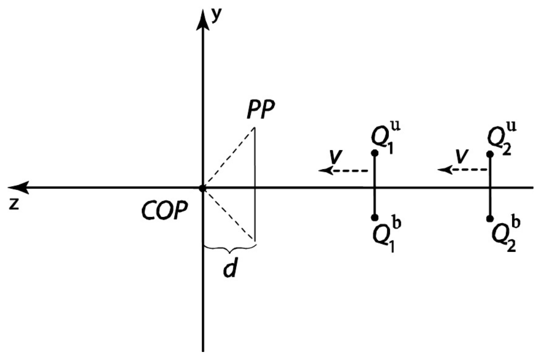
Compute the projections q_1 and q_2 of points Q_1 and Q_2, respectively. Then, compute the velocities, dq_1 / dt and dq_2 / dt, of each projected point in the image plane. Which appears to move faster? Show your work.

b) (12 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*.

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

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

$$0 > z_1 > z_2$$



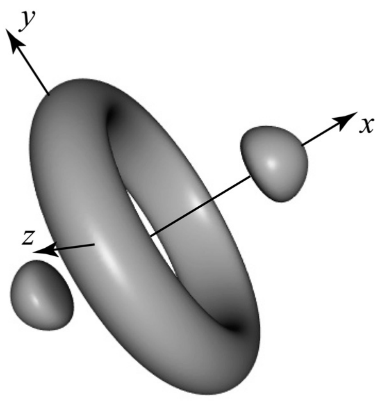
Compute the projected lengths, l_1 and l_2, of the line segments. Then, compute the rates of change, dl_1 / dt and dl_2 / dt, of these projected lengths. Are they growing or shrinking? Which projected line segment is changing length faster? Show your work.

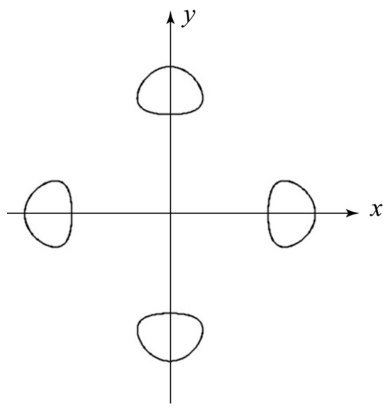
Problem 2. Ray intersection with implicit surfaces (25 points)

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 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.





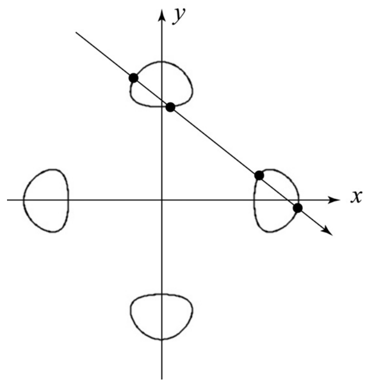
In the next problem steps, you will be asked to solve for and/or discuss ray intersections with this primitive. Performing the ray intersections will amount to solving for the roots of a polynomial, much as it did for sphere intersection. For your answers, you need to keep a few things in mind:

- 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* + 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.

a) (10 points) Consider the ray $P+t\mathbf{d}$, where $P=\begin{pmatrix}0 & 0 & 0\end{pmatrix}$ and $\mathbf{d}=\begin{pmatrix}0 & 1 & 0\end{pmatrix}$. Solve for all values of *t* where the ray intersects the gumdrop torus (**including** any negative values of *t*). Which value of *t* represents the intersection we care about for ray tracing? Show your work. 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 2 (cont'd)

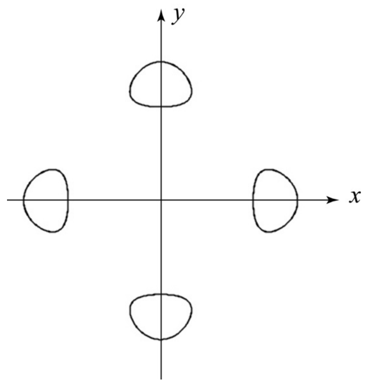
b) (15 points) What are all the possible combinations of roots, not counting the one in part (a)? For each combination, describe the 4 roots as in part (a), draw a ray in the x-y plane that gives rise to that combination, and place a dot at each intersection point. You may not need all of the given diagrams. The first one has already been filled in. (Note: not all conceivable combinations can be achieved on this particular 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**

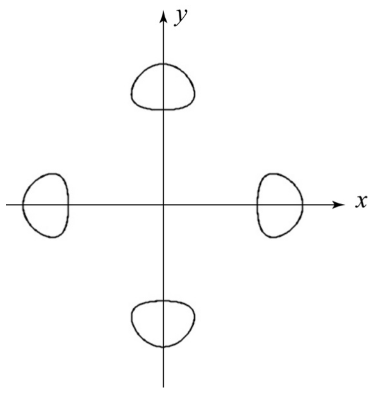
# of complex roots: **0**



# of distinct real roots:

# of roots w/ multiplicity > 1:

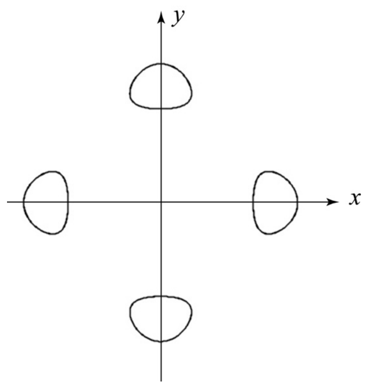
# of complex roots:



# of distinct real roots:

# of roots w/ multiplicity > 1:

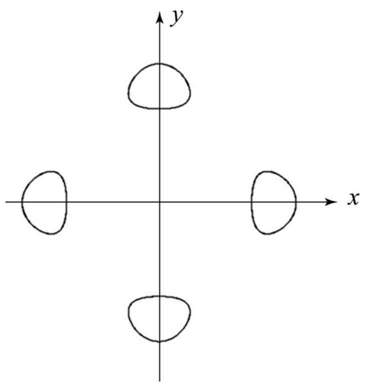
# of complex roots:



# of distinct real roots:

# of roots w/ multiplicity > 1:

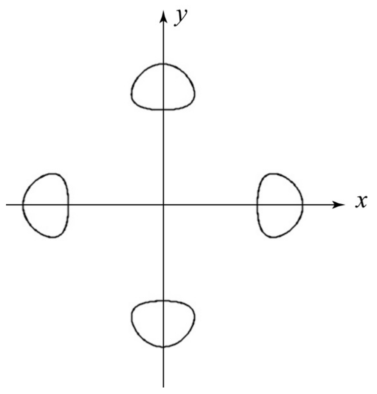
# of complex roots:



# 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 3. Z-buffers and distribution ray tracing (23 points)

Suppose we want to combine the Z-buffer algorithm with ray tracing (assuming that the scene consists only of polygons). We can assign to each polygon a unique emissive color corresponding to an "object ID". Then, we turn off all lighting and render the scene from a given point of view. At each pixel, the Z-buffer now contains the object point (indicated by the pixel *x,y* coords and the *z*-value stored in the buffer) and an object ID which we can look up to figure out the shading parameters. In effect, we have done a ray cast for a set of rays passing through a given point (the center of projection). Answer the following questions and include a short explanation of each answer.

a) (2 points) Consider the figure below. The ray cast from the eye point through an n x n image (projection) plane is actually going to be computed using the Z-buffer algorithm described above. Let's assume that we will effectively increase the resolution to give m x m supersampling per pixel, for antialiasing purposes. In effect, how many rays are fired from the eye to determine the first intersected surfaces?

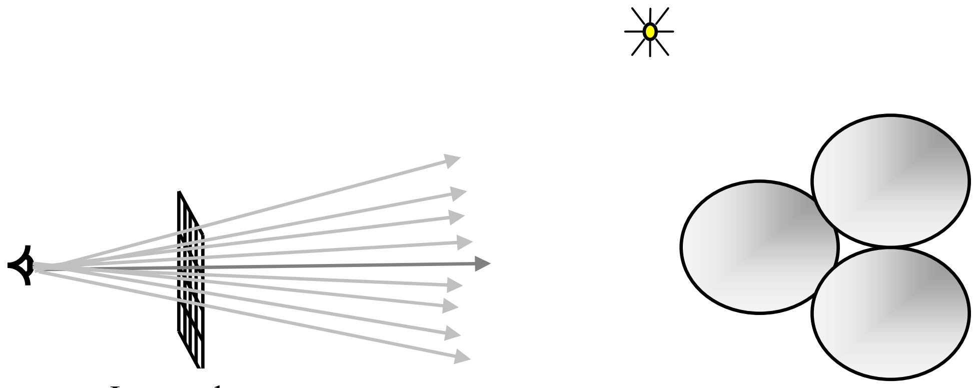
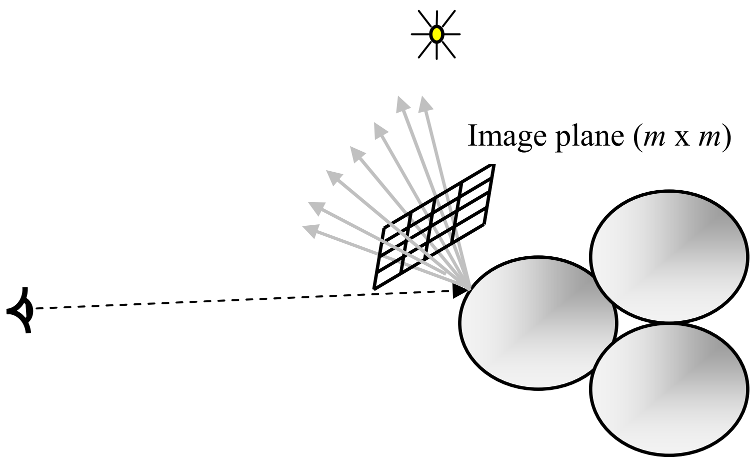


Image plane

b) (4 points) Now let's say we want to capture glossy reflections by spawning many rays at each intersection point roughly in the specular direction. As indicated by the figure below, we can cast these rays by positioning the new viewpoint at a given intersection point, setting up a new image plane of size m x m (the same as the pixel antialiasing supersampling rate), oriented to align with the specular direction, and then run our modified Z-buffer algorithm again. Let's say we follow this procedure for all pixels for k bounces in a scene assuming non-refractive surfaces. Assume that all rays strike surfaces. In effect, how many rays will we end up tracing?



Problem 3. (cont'd)

c) (4 points) Now assume that all the surfaces are also refractive. If we compute a spread of transmitted rays to simulate translucency (blurry refraction) as well, how many rays will we end up tracing?

Now, instead of using a Z-buffer algorithm, we use the distribution ray tracing method. Again, the image plane is n x n, and will trace m x m rays per pixel. Count how many rays do we need to trace for each of these cases:

d) (2 points) First intersections (0 bounces)

e) (4 points) k bounces to simulate gloss only (no refraction/translucency).

f) (4 points) k bounces to simulate both gloss and translucency.

g) (3 points) Given your answers to (a)-(f) when might it be appropriate to use the modified Z-buffer in a ray tracing algorithm?

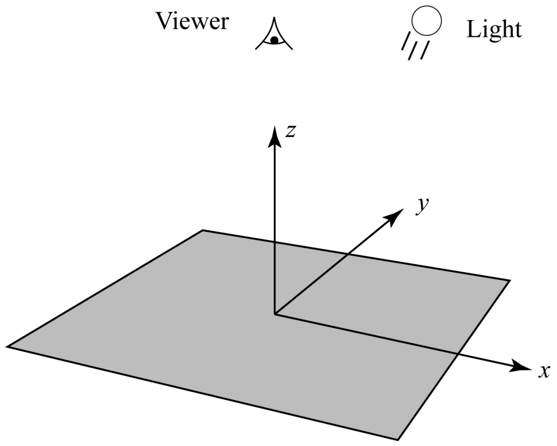
Problem 4. Shading, displacement mapping, and normal mapping (32 points)

In this problem, an opaque surface will be illuminated by one directional light source and will reflect light according to the following Phong shading equation:

$$I = A_{\text{shadow}} I_{\text{light}} \left(k_d \left(\mathbf{N} \cdot \mathbf{L} \right)_+ + k_s \left(\mathbf{V} \cdot \mathbf{R} \right)_+^{n_s} \right)$$

Note the inclusion of a shadowing term, which takes on a value of 0 or 1. For simplicity, we will assume a monochrome world where I, I_{light}, k_d, and k_s are scalar values. You may assume $I_{\text{light}} = 0.5$.

Suppose a viewer is looking down at an infinite plane as illustrated below. The scene is illuminated by a directional light source, also pointing straight down on the scene.



Answer the following questions below, giving a brief justification of each answer. Note that lighting and viewing directions are from the point of view of the light and viewer, respectively, and need to be negated when considering the surface-centric shading equation above.

a) (2 points) Assume: Perspective viewer at (0,0,1) looking in the (0,0,-1) direction, angular field of view of 90 degrees, lighting direction of (0,0,-1), $k_d = 1$, $k_s = 0$. Describe the brightness variation over the image seen by the viewer. Justify your answer.

b) (2 points) Assume: Perspective viewer at (0,0,1) looking in the (0,0,-1) direction, , angular field of view of 90 degrees, lighting direction of (0,0,-1), $k_d = 0$, $k_s = 1$, $n_s = 10$. Describe the brightness variation over the image seen by the viewer. Justify your answer.

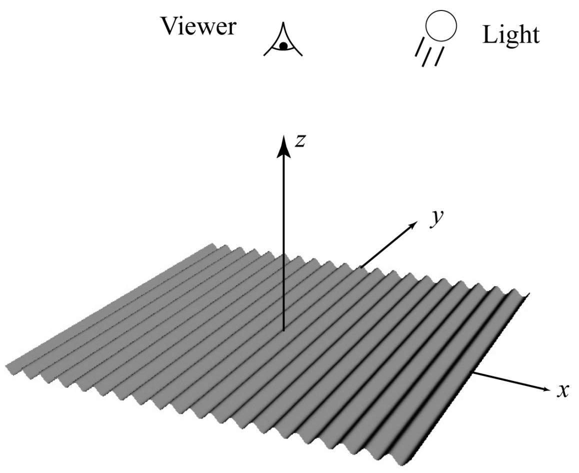
c) (2 points) Assume: Orthographic viewer looking in the (0,0,-1) direction, lighting direction of (0,0,-1), $k_d = 0$, $k_s = 1$, $n_s = 10$. Describe the brightness variation over the image seen by the viewer. Justify your answer.

d) (3 points) Assume: Orthographic viewer looking in the (0,0,-1) direction, $k_d = 1$, $k_s = 0$. The lighting direction starts at (-sqrt(2)/2 ,0, -sqrt(2)/2) and then rotates around the z-axis. Describe the brightness variation over time, as seen by the viewer. Justify your answer.

e) (3 points) Assume: Orthographic viewer looking in the (0,0,-1) direction, $k_d = 0$, $k_s = 1$, $n_s = 10$. The lighting direction starts at (-sqrt(2)/2 ,0, -sqrt(2)/2) and then rotates around the z-axis. Describe the brightness variation over time, as seen by the viewer. Justify your answer.

Problem 4. (cont'd)

Suppose now the infinite plane is covered with a displacement map, d=cos(x):



Since the displacement along the normal is always in the z direction, this is equivalent to creating a surface defined as $z = \cos(x)$.

f) (5 points) Assume: Orthographic viewer looking in the (0,0,-1) direction, lighting direction of (0,0,-1), $k_d = 1$, $k_s = 0$. At what values of x is the surface brightest? At what values of x is it dimmest? Describe the appearance of the surface. Justify your answers.

g) (5 points) Assume: Orthographic viewer looking in the (0,0,-1) direction, lighting direction of (0,0,-1), k_d =0, k_s =1, n_s =10. At what values of x is the surface brightest? Describe the appearance of the surface. How does the appearance change as n_s increases to 100? Justify your answers.

Suppose now that we simply keep the normals computed in (f) and map them over the plane from the first part of the problem. The geometry will be flat, but the shading will be based on the varying normals.

h) (6 points) Assume: Orthographic viewer looking in the (0,0,-1) direction, k_d =1, k_s =0. If we define the lighting to have direction (-sinθ, 0, -cosθ), will the normal mapped rendering look the same as the displacement mapped rendering for each of θ = 0, 10, and 80 degrees? Justify your answer (for each of these angles).

i) (4 points) Assume: Orthographic viewer, lighting direction of (0,0,-1), k_d =1, k_s =0. As we move the viewer around, will the normal mapped rendering look the same as the displacement mapped rendering? Justify your answer.