Homework #2

Projections, Shading, Ray Tracing, and Texture Mapping

Assigned: Thursday, May 6th

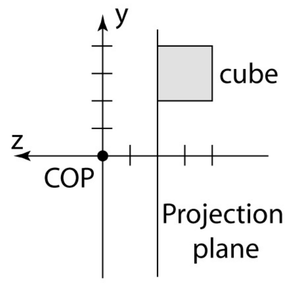
Due: Wednesday, May 19th
at the beginning of class

Directions: Please provide short written answers to the following questions. Be sure to write your name on the assignment. Feel free to discuss the problems with classmates, but please ***answer the questions on your own*.**

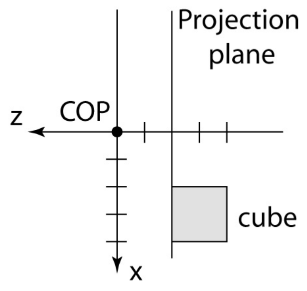
Name:____

Problem 1: Projections (16 points)

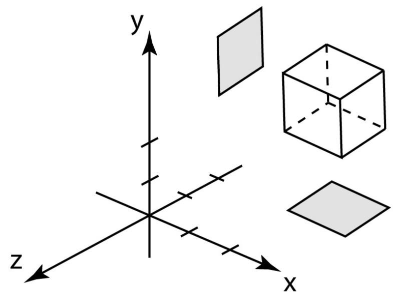
Shown below on the left are two parallel projections of a cube, a perspective projection (image) plane, and a center of projection (COP). The “side view” is the parallel projection looking down the -x direction and the “top view” is the parallel projection looking down the -y direction. On the far right is a visualization of just the cube, with projections onto the x-z and y-z planes shown as gray quadrilaterals. As can be determined from the drawings, the corner of the cube closest to the origin is at (2, 2,-2), the cube has side length of 2, and the projection plane is at z = -2.



Side view



Top view



Positioning of cube in 3D

For the questions below, you will start by considering the perspective projection that arises from the geometry illustrated in the figure.

As part of this problem, you will be asked about the vanishing points of the cube. In particular, each line segment of the cube lies on a line that has a vanishing point. We will refer to this as one of the vanishing points of the cube. However, some vanishing points may be at infinity; *we will not consider or count vanishing points at infinity in this problem*.

(a) (4 points) Form the projection matrix and compute the (x ,y) coordinates of the eight corners of the cube after projection.

(b) (4 points) Sketch the cube as it would be seen in the perspective projection given the center of projection and projection plane. Place marks at unit spacing on the axes (analogous to what has been done in the figure) so that the locations of the cube corners can be read from the drawing. Draw hidden lines as dashed, and identify any vanishing points (labeled “VP” on your drawing).

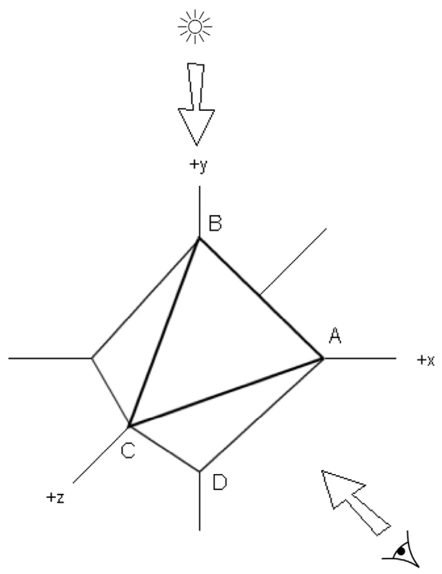
(c) (2 points) What happens to the vanishing point(s) as we translate the cube in the –y direction? Explain your reasoning.

(d) (3 points) If you can freely rotate and translate the cube, what are the minimum and maximum number of vanishing points you will see in its projection? Explain your reasoning.

(e) (3 points) In general, if we rotate and translate the projection plane without moving the center of projection and without moving the object, how do the occlusion relationships change? That is, do surfaces that were occluded in one image become unoccluded in another and/or vice versa? Explain your reasoning.

Problem 2. Interpolated shading (20 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?

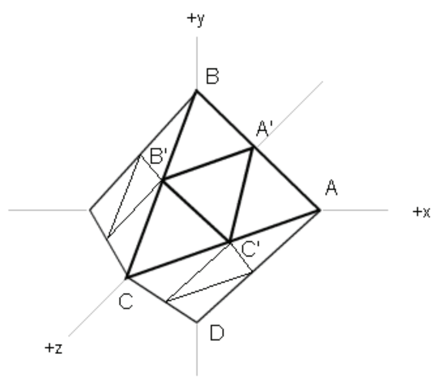
(b) (3 points) Assume that this object is really just a crude approximation of a sphere (e.g., perhaps you are using the octahedron to represent the sphere because your graphics card is slow). If you want to shade the octahedron so that it approximates the shading of a sphere, what would you specify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

(c) (6 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?



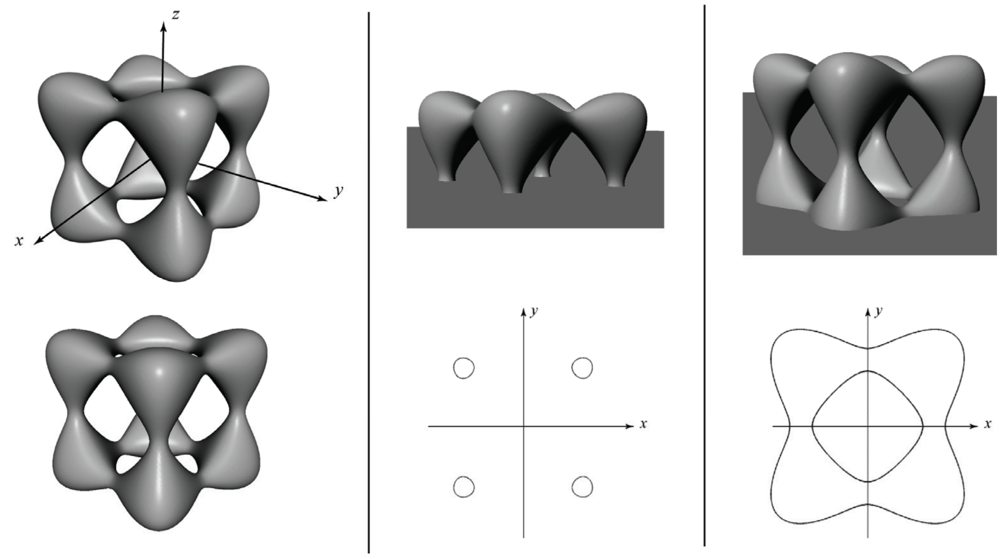
(g) (2 points) If you continued this subdivision process – repeatedly performing 4-to-1 subdivision, repositioning the inserted vertices, and computing their ideal normals – would the Gouraud-interpolated and Phong-interpolated renderings of the refined shape converge toward the same answer, i.e., the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

Problem 3. Ray intersection with implicit surfaces (24 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 "tangle cube," described by the equation:

$$x^4+y^4+z^4-5x^2-5y^2-5z^2+12=0$$

In the figure below, the left column shows two renderings of the tangle cube, the middle column illustrates taking a slice through the *x-y* plane (at *z* = 0), and the right column shows a slice parallel to the *x-y* plane taken toward the bottom of the tangle cube (plane at z ≈ -1.5):



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:

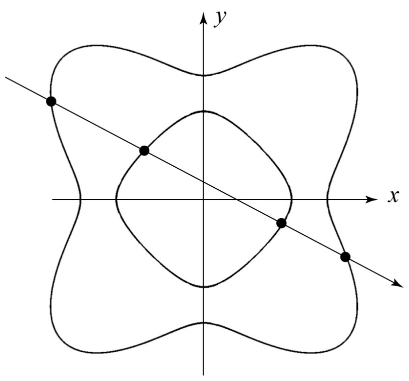
- You will find as many roots as the order (largest exponent) of the polynomial.
- You may find a mixture of real and complex roots. When we say complex here, we mean a number that has a non-zero imaginary component.
- All complex roots occur in complex conjugate pairs. If *A* + i*B* is a root, then 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.

(a) (9 points) Consider the ray $P+t\mathbf{d}$, where $P=\begin{pmatrix}0 & 0 & 0\end{pmatrix}$ and $\mathbf{d}=\begin{pmatrix}1 & 1 & 0\end{pmatrix}$. Typically, we normalize **d,** but for simplicity (and without loss of generality) you can work with the un-normalized **d** as given here.

- Solve for all values of *t* where the ray intersects the tangle cube (**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 3 (cont'd)

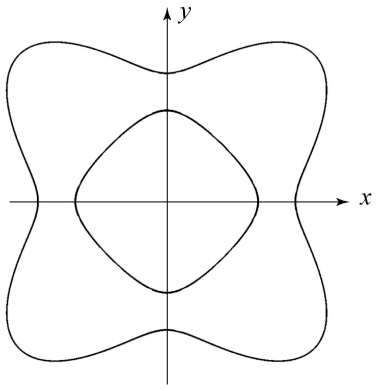
b) (15 points) What are all the possible combinations of roots when ray tracing this surface, not counting the one in part (a)? For each combination, describe the 4 roots as in part (a), draw a ray in the *x*-*y* plane that gives rise to that combination, and place a dot at each intersection point. Assume the origin of the ray is outside of the bounding box of the object. There are five diagrams below that have not been filled in. You may not need all five; on the other hand, if you can actually think of more distinct cases than spaces provided, then we might just give extra credit. The first one has already been filled in. (Note: not all conceivable combinations can be achieved on this particular implicit surface. 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 solution. You do not need to justify your answers.***



# of distinct real roots: **4**

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

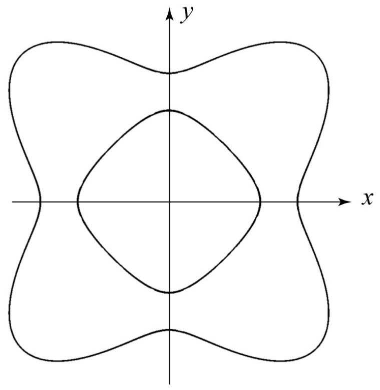
# of complex roots: **0**



# of distinct real roots:

# of real roots w/ multiplicity > 1:

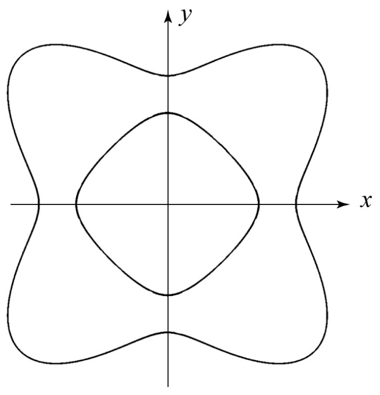
# of complex roots:



# of distinct real roots:

# of roots w/ multiplicity > 1:

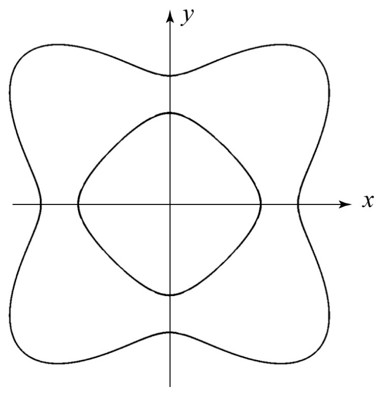
# of complex roots:



# of distinct real roots:

# of real roots w/ multiplicity > 1:

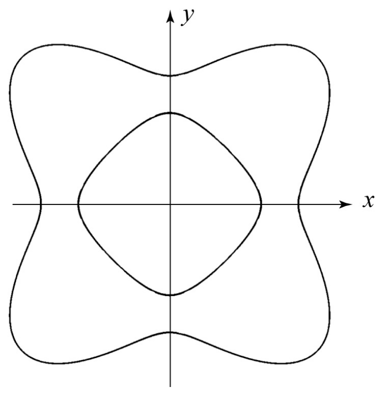
# of complex roots:



# of distinct real roots:

# of real roots w/ multiplicity > 1:

# of complex roots:



# of distinct real roots:

# of real roots w/ multiplicity > 1:

# of complex roots:

Problem 4. Counting rays (25 points)

In this problem, we study the number of rays traced for using different ray tracing algorithms. Consider the following setup:

m x m pixels
k x k supersampling
n geometric primitives
ℓ light sources
d bounces (reflections and/or refractions)

For each of the algorithms and scenarios discussed in parts (a)-(e) below, assume the following:

- You are counting rays cast, including primary rays, shadow (light) rays, reflected rays, and (when asked for in the problem) refracted rays.
- *No* acceleration techniques are used.
- *Every* recursively traced (reflected or refracted) ray hits an object, including the primary rays.
- You will always cast a ray to the light source after intersecting an object, and this does not count as a recursive "bounce" (but certainly counts as a cast ray).
- Each ray cast to a light source counts as a single ray-cast, even when accounting for transparent shadows. (The transparent shadow case can be handled by keeping track of all intersections encountered – not just the closest – when casting a ray to a light, so this is a reasonable assumption.)

You do not need to justify your answers, though doing so may help you to earn partial credit. For each sub-problem, write out a summation (with the Σ symbol for the summation) and then, if possible, convert it to closed form.

a) (5 points) For Whitted ray tracing, assuming reflection (but *no* refraction) at every surface, how many rays are cast?

b) (5 points) For Whitted ray tracing, assuming reflection *and* refraction at every surface, how many rays are cast?

c) (5 points) Suppose now, in order to get glossy reflections, you recursively cast k x k rays around the reflection direction at each bounce. Assuming glossy reflection (but *no* refracted/transmitted rays) at every surface, how many rays are cast?

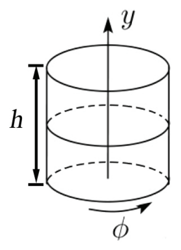
d) (5 points) In addition, in order to get translucent (blurry) refraction effects, you recursively cast k x k rays around the refraction direction at each bounce. Assuming glossy reflection and translucent refraction at every surface, how many rays are cast?

e) (5 points) Suppose now you switch to using distribution ray tracing. Assuming glossy reflection and translucent refraction at every surface, how many rays are cast?

Problem 5. Texture mapping (15 points)

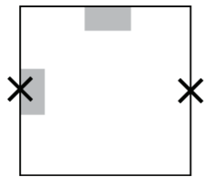
When a texture map is applied to a surface, points that are distinct in the rectangular texture map may be mapped to the same place on the object. For example, when a texture map is applied to a cylinder, the left and right edges of the texture map are mapped to the same place. We call a mapping "valid" if it does not map two points of different colors to the same point on the object. For each of the 16 cases below, indicate whether the mapping is valid (write "Yes" or "No" above the texture map). If it is not valid, mark with an **X** two points on the texture map that map to the same point on the object, but have different colors. Use the texture mapping formulas specified below for each primitive, where the u, v parameters range from 0 to 1. The (u, v) origin of each texture map is in the lower left corner of the texture. The first of the 16 cases below is done for you. ***Please write on this page and include it with your homework solution. You do not need to justify your answers.***

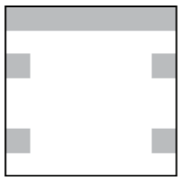
Uncapped cylinder (top and bottom are open)

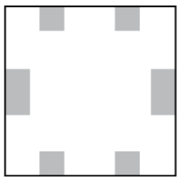


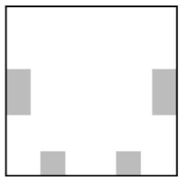
$$\begin{cases} u = \phi/2\pi \\ v = y/h \end{cases}$$

No

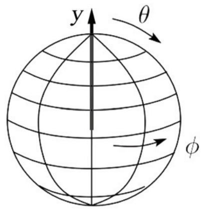




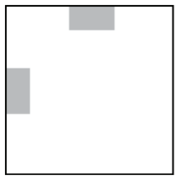


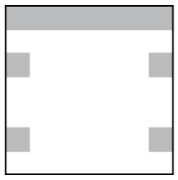


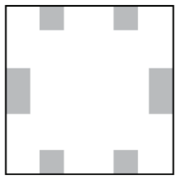
Sphere

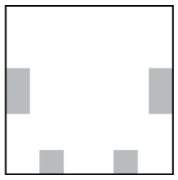


$$\begin{cases} u = \phi/2\pi \\ v = \theta/\pi \end{cases}$$

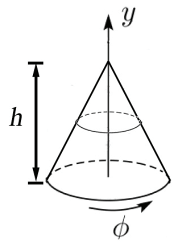




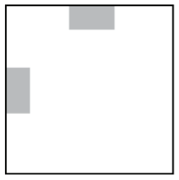


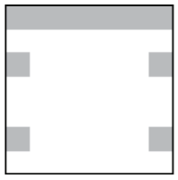


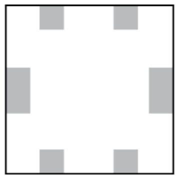
Uncapped cone (bottom is open)

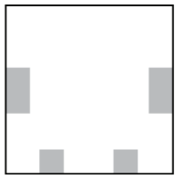


$$\begin{cases} u = \phi/2\pi \\ v = y/h \end{cases}$$

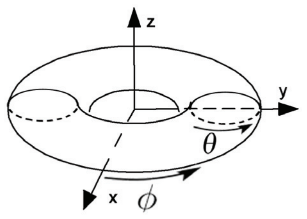








Torus



$$\begin{cases} u = \phi/2\pi \\ v = \theta/2\pi \end{cases}$$