Homework #2

Hidden Surfaces, Shading, Ray Tracing, Texture Mapping, and Parametric Curves

Assigned: Friday, May 22nd

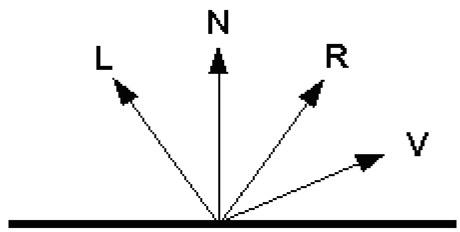
Due: Wednesday, June 10th
at the beginning of class

Directions: Please provide short written answers to the following questions. Be sure to write your name on the assignment. Please answer the questions on your own and show your work. If you have questions about the assignment, please contact the instructor directly.

Problem 1. Halfway vector specular shading (10 points)

Blinn and Newell have suggested that if **V** and **L** are each assumed to be constants the computation of $\mathbf{V} \cdot \mathbf{R}$ in the Phong shading model can be simplified by associating with each light source a fictitious light source that will generate specular reflections. This second light source is located in a direction **H** halfway between **V** and **L**. The specular component is then computed from $(\mathbf{N} \cdot \mathbf{H})^{n_s}$, instead of from $(\mathbf{V} \cdot \mathbf{R})^{n_s}$. This is in fact the specular shading model used by OpenGL. ***Please write on this page and include it with your homework solution.***

a) (2 points) On the diagram below, assume that **V** and **L** are the new constant viewing direction and lighting direction vectors. Draw the new direction **H** on the diagram.



b) (4 points) Under what circumstances are **L** and **V** *exactly* constant for every point in the scene as seen through every pixel on the image plane? Under what circumstances are they *approximately* constant? (Answer both questions.)

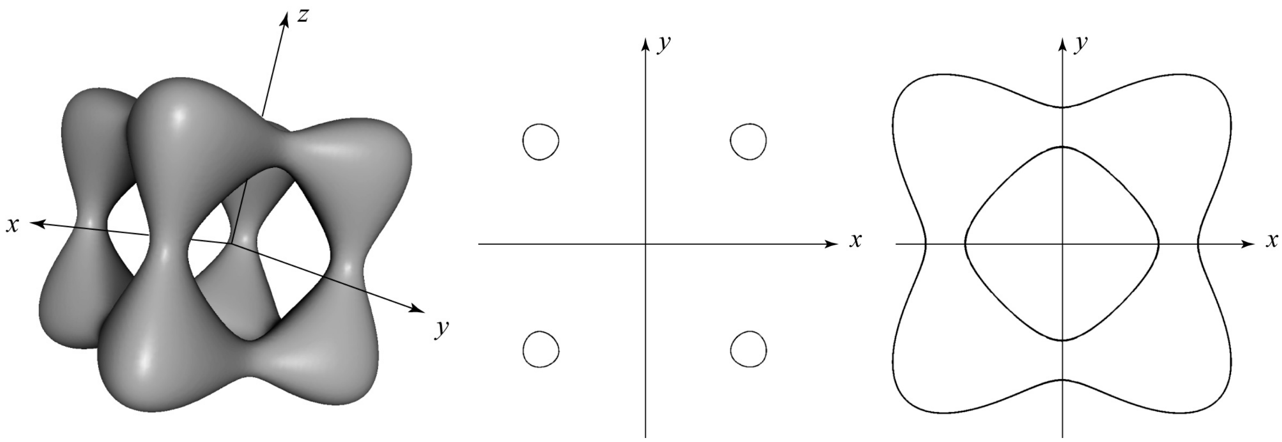
c) (4 points) Let's take the approach of simply assuming **L** and **V** are constant for every point in the scene as seen through every pixel in the image plane – regardless of whether this assumption is satisfied under the actual lighting and viewing conditions – and use the halfway vector for specular shading. What is an advantage of this approach? For general lighting and viewing conditions, what is a drawback of this approach? (Answer both questions.)

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 "tangle cube," described by the equation:

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

On the left is a picture of a tangle cube, in the middle is the slice through the *x-y* plane (at *z* = 0), and on the right is a slice parallel to the *x-y* plane taken toward the bottom of the tangle cube (plane at z ≈ -1.5):



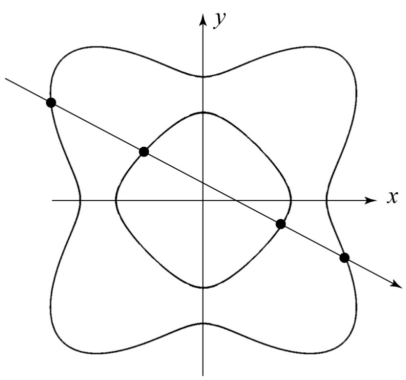
In the next problem steps, you will be asked to solve for and/or discuss ray intersections with this primitive. Performing the ray intersections will amount to solving for the roots of a polynomial, much as it did for sphere intersection. For your answers, you need to keep a few things in mind:

- 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) (10 points) Consider the ray $P + t\mathbf{d}$, where $P = (0 \quad 0 \quad 0)$ and $\mathbf{d} = (1 \quad 1 \quad 0)$. 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.
   - 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 2 (cont'd)

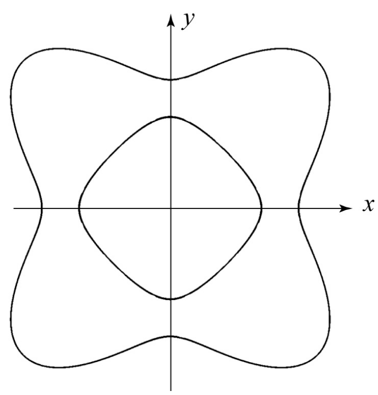
b) (15 points) What are all the possible combinations of roots, not counting the one in part (a)? For each combination, describe the 4 roots as in part (a), draw a ray in the x-y plane that gives rise to that combination, and place a dot at each intersection point. Assume the origin of the ray is outside 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 roots w/ multiplicity > 1: **0**

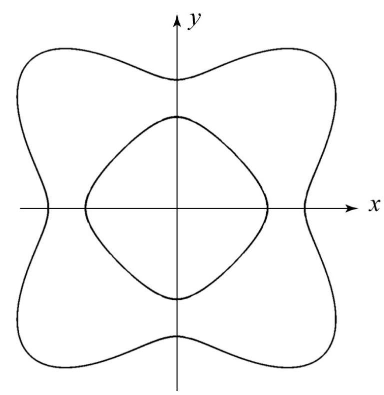
# of complex roots: **0**



# of distinct real roots:

# of roots w/ multiplicity > 1:

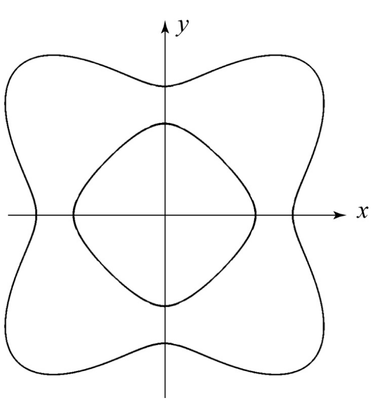
# of complex roots:



# of distinct real roots:

# of roots w/ multiplicity > 1:

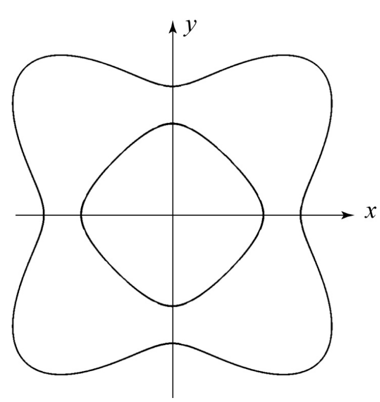
# of complex roots:



# of distinct real roots:

# of roots w/ multiplicity > 1:

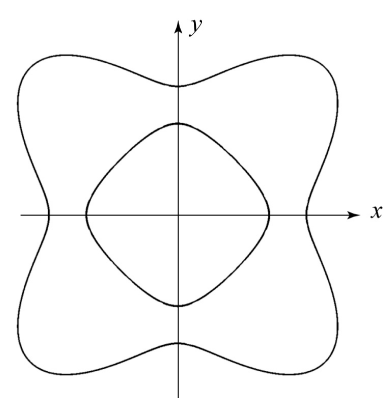
# of complex roots:



# 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. Counting rays (30 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) (6 points) For Whitted ray tracing, assuming reflection (but *no* refraction) at every surface, how many rays are cast?

b) (6 points) For Whitted ray tracing, assuming reflection *and* refraction at every surface, how many rays are cast?

c) (6 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* refraction) at every surface, how many rays are cast?

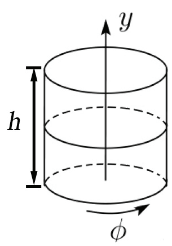
d) (6 points) In addition, in order to get translucent (blurry) refraction effects, you recursively cast k x k rays around the refraction direction at each bounce. Assuming glossy reflection and translucent refraction at every surface, how many rays are cast?

e) (6 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 4. Texture mapping (15 points)

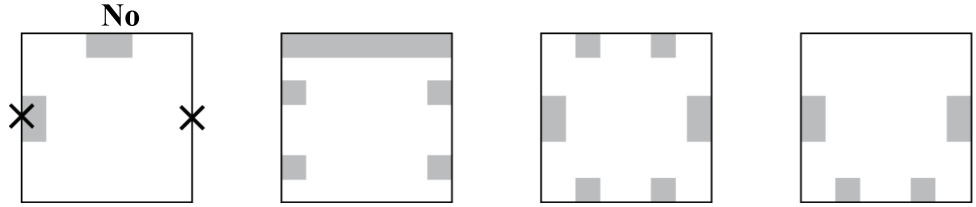
When a texture map is applied to a surface, points that are distinct in the rectangular texture map may be mapped to the same place on the object. For example, when a texture map is applied to a cylinder, the left and right edges of the texture map are mapped to the same place. We call a mapping "valid" 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 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)

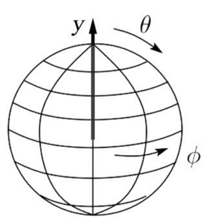


$$\begin{cases} u = \phi/2\pi \\ v = y/h \end{cases}$$

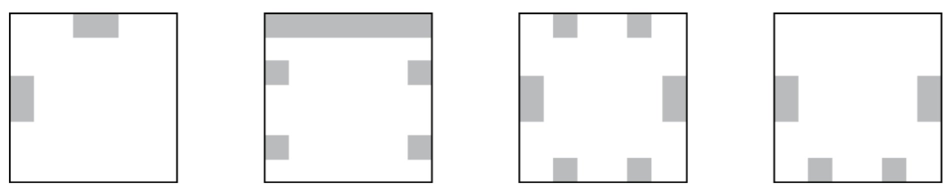
No



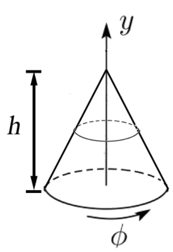
Sphere



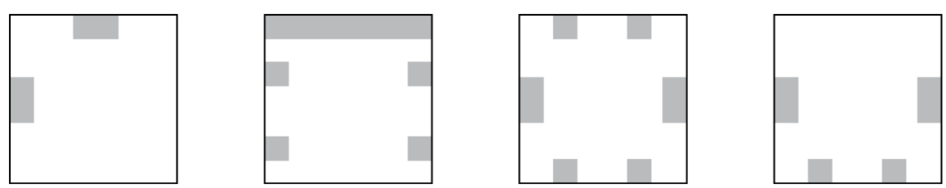
$$\begin{cases} u = \phi/2\pi \\ v = \theta/\pi \end{cases}$$



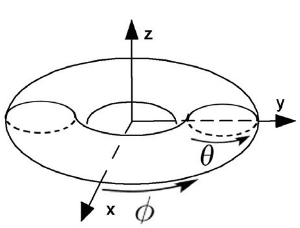
Uncapped cone (bottom is open)



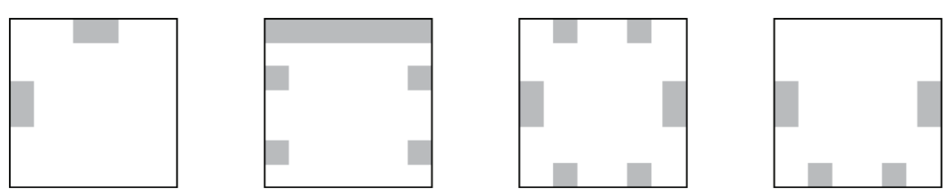
$$\begin{cases} u = \phi/2\pi \\ v = y/h \end{cases}$$



Torus



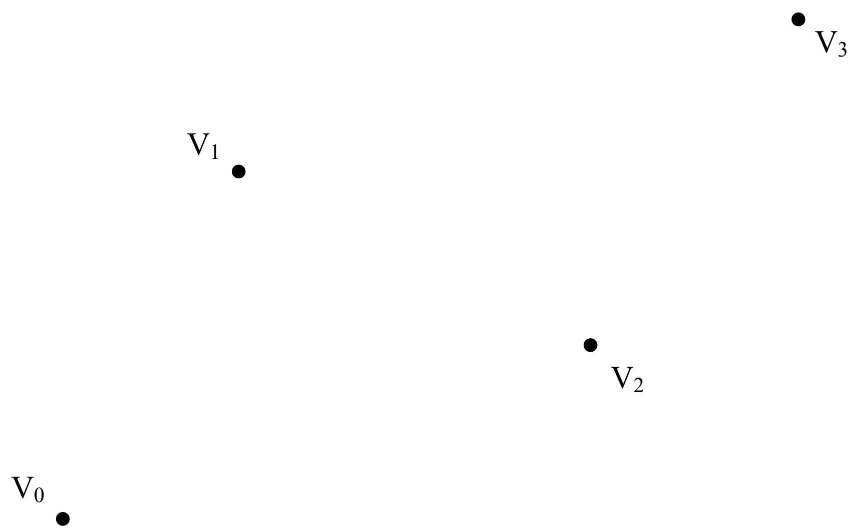
$$\begin{cases} u = \phi/2\pi \\ v = \theta/2\pi \end{cases}$$



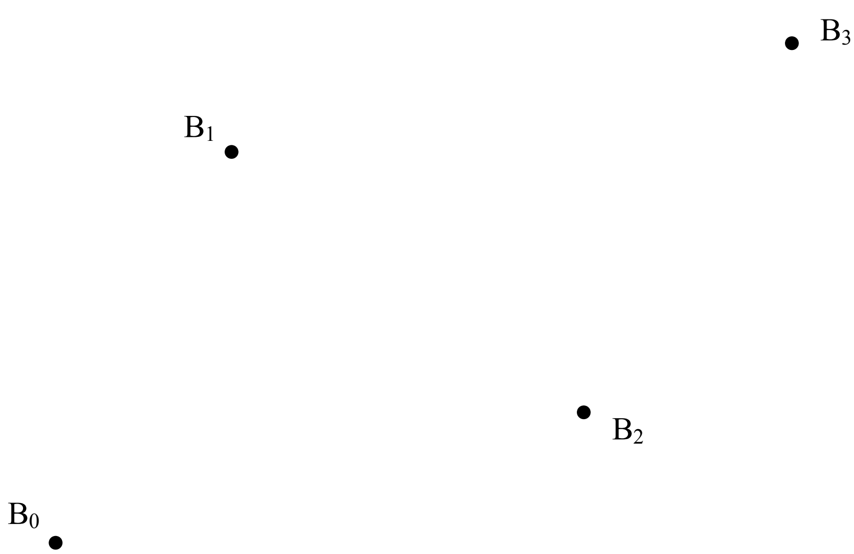
Problem 5. Parametric curves (20 points)

In this problem, we will explore the construction of parametric. ***Please write on the pages for this problem and include them with your homework solution.***

a) (4 points) Given the following Bezier control points, construct all of the de Casteljau lines and points needed to evaluate the curve at $u=1/3$. Mark this point on your diagram and then sketch the path the Bezier curve will take. The curve does not need to be exact, but it should conform to some of the geometric properties of Bezier curves (convex hull condition, tangency at endpoints).



b) (8 points) Given the following de Boor points, construct all of the lines and points needed to generate the Bezier control points for the B-spline. Assume that the first and last points are each repeated three times, so that the spline is endpoint interpolating. You must mark each Bezier point (*including any that coincide with a de Boor point*) with an X, but you do not need to label it (i.e., no need to give each Bezier point a name). Sketch the resulting spline curve, respecting the properties of Bezier curves noted above.



Problem 4 (cont'd)

c) (8 points) Given the following Catmull-Rom control points, construct all of the lines and points needed to generate the Bezier control points for the Catmull-Rom curve. Use a tension value of τ=1. Assume that the first and last points are each repeated two times. You must mark each Bezier point (*including any that coincide with a Catmull-Rom control point*) with an X, but you do not need to label it (i.e., no need to give each Bezier point a name). Sketch the resulting spline curve, respecting the properties of Bezier curves noted above.