Computer Graphics Comprehensive Exam

Computer Science Department
Stanford University
Fall 2003

NAME:

Note: This is exam is *closed-book.*

The exam consists of 5 questions. Each question is worth 20 points. Please answer all the questions in the space provided, overflowing on to the back of the page if necessary.

You have 60 minutes to complete the exam.

1. [20 points] Computer graphics definitions.

Define in a few sentences each of the following computer graphics terms. Some of these terms may be used in other fields, so be sure to give the computer graphics meaning.

1A [5 points] Alpha channel.

The alpha channel stores transparency or opacity. An alpha channel makes it possible to composite one image over another image.

1B [5 points] Phong reflection model.

The Phong reflection is $(R \, . \, L)^s$.

In this equation, L is a vector to the light source, and R is the reflection of the vector to the E across the normal N.

1C [5 points] Spline.

A spline is a smooth curve constrained by a small number of control points. Typical splines are polynomial curves of low-degree and the curve passes near or through the control points.

1D [5 points] Gouraud shading.

In Gouraud shading, lighting values or colors at vertices are interpolated across the polygon.

2. [20 points] Transformations.

Computer graphics relies heavily on transformations. Most common are linear transformations such as rotations and translations. In the following, T(dx,dy,dz) refers to a translation by (dx,dy,dz), Rx(a) refers to a rotation about the x-axis by a degrees, Ry(a) and Rz(a) refer to rotations about the y- and z-axis, respectively.

The order of transformations may matter. Also, sometimes the order may be rearranged, but the arguments will change. Describe whether the following statements are true or false.

T(1,0,0) Rx(360) = Rx(720) T(1,0,0)? TRUE

T(1,0,0) T(0,2,0) = T(0,1,0) T(1,1,0)? TRUE

Rx(45) Ry(30) = Ry(45) Rx(30)? FALSE

Rx(45) Rx(30) = Rx(15) Rx(60)? TRUE

T(1,0,0) Rz(180) = T(1,0,0) Rz(-180) TRUE

Transformations have inverses. State whether the following formulas for the inverse transformations are true or false.

$Rz(45)^{-1} = Rz(-45)$? TRUE

$Rz(180)^{-1} = Rz(180)$? TRUE

$[T(1,0,0)T(0,2,0)]^{-1} = T(-1,0,0)T(0,-2,0)$? TRUE

$[Rz(45)\ T(1,0,0)]^{-1} = T(-1,0,0)\ Rz(45)$? FALSE

$[Rx(45)\ Ry(30)]^{-1} = Rx(-45)\ Ry(-30)$? FALSE

3. [20 points] Ray tracing.

One of the most general methods for rendering is ray tracing. At the core of a ray tracer is a procedure to find ray-surface intersections. The inputs to the procedure are a ray and a description of a surface; the output is the point of intersection.

Assume a ray is given by the following parametric equations:

$$x = x0 + t * x1$$
$$y = y0 + t * y1$$
$$z = z0 + t * z1;$$

(x0,y0,z0) is the origin of the ray, and (x1,y1,z1) is the direction of the ray. As t increases, the points on the ray move from the origin along the direction.

The simplest surface is a plane. The following equation defines a plane:

$$Ax + By + Cz + D = 0.$$

A plane also defines a half-space: $Ax + By + Cz + d < 0$.

A set of planes may be used to define convex polyhedra. Each face i of the convex polyhedra is associated with a plane (Ai, Bi, Ci, Di). A convex polyhedra is defined to be the intersection of the half-spaces created by the planes making up its faces.

Work out a procedure for computing the point of intersection of a ray with a convex polyhedra defined as the intersection of n half-spaces. In general, a ray intersects a convex shape in two points. Your procedure should return the closest point of intersection in the direction of the ray.

There are two major operations that need to be done. Compute the point of intersection with the ray and the plane. Test whether the point is on the surface of the poyhedra. Finally, the closest point with $t > 0$ must be returned.

The intersection of a point and a plane is $t = (A\, x0 + B\, y0 + C\, z0 + D)/(A\, x1 + B\, y1 + d\, z1)$. Ignore points when the denominator equals 0.

For each of these intersection points, test whether it is inside all the other planes.

Finally, set the intersection point to be the closest point if $t > 0$ and t is less than the closest point found so far.

This is an O(n^2) algorithm. There is also an O(n) algorithm!

4. [20 points] Hidden-surface elimination.

Hidden-surface elimination is one the classic algorithms in computer graphics. The goal of hidden-surface elimination is to draw a picture where each point in the image shows the surface visible at that point; hidden points and surfaces are not shown.

4A [5 points] Describe succinctly the z-buffer algorithm for hidden surface elimination.

A z-buffer is an additional image buffer that stores the depth of the current nearest point that has been drawn. As new fragments are drawn, their depth value is compared to value in the z-buffer. If the new value is closer than the existing value, the new fragment is drawn and the z-buffer is updated with the new z-value

4B [5 points] What is the complexity of the z-buffer algorithm?

The z-buffer algorithm requires N polygons to be drawn. If each polygon covers on average a pixels, then Na fragments are drawn.

4C [5 points] Is the z-buffer algorithm optimal? What would be the complexity of an optimal algorithm?

No. The optimal algorithm would only draw visible polygons and visible pictures. The depth complexity d is the average number of pixels drawn per image pixel. d is 1 for an optimal algorithm, and $d > 1$ for the z-buffer algorithm.

4D [5 points] Suppose you are given a scene consisting of a set of polygons. You can draw polygons as lines (i.e. as an outlined polygon) or as a filled polygon. How would you create a line drawing of the scene with hidden lines removed? Hint: Consider enhancing the basic z-buffer drawing mode slightly.

Add a drawn if z-equal. That is, draw if the fragment's z-value is less than or equal to the existing value in the z-buffer.

First, draw all the polygons in the scene, updating the z-buffer but not the color buffer.

Second, draw all the lines in z-equal mode.

Hardware must be provided to implement pow(L, 1/gamma) – the mapping from L to V. This is easiest to do with a lookup table.