Computer Graphics Final Exam (2016)

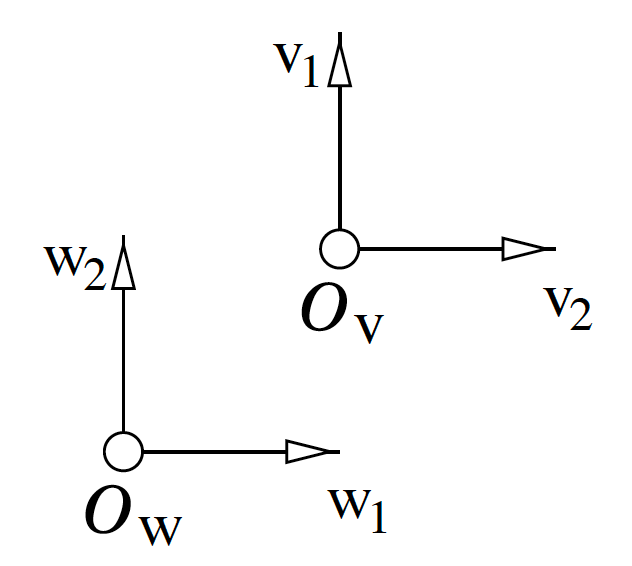
1. Fill in the following blanks (10 pts)
2. When representing vectors in 3D using homogenized coordinates, the fourth coordinate (i.e. “w”) will be zero.
3. The radiosity method operates from the assumption that all surfaces in a scene act like diffuse reflectors.
4. Quaternions encode 3D rotations as point in 4D space.
5. Parallel projection is a special case of perspective projection where the viewer is infinitely far away.
6. When two curve segments join at a point and both curves approach that point with the same derivative, the joining is said to be parametric continuous.
7. (a) Do parallel lines map to parallel lines under affine transformations? If so, then prove it. If not, give a counter example. (5 pts)

A: yes. The reason is that the affine transformation will keep infinity points still in infinity. That means, the intersection point of two parallel lines will be in the infinity after the two lines are applied affine transformation, then the two transformed lines are parallel.

(b) Do perpendicular lines map to perpendicular lines under affine transformations? If so, then prove it. If not, give a counter example. (5 pts)

A: no. For example, two adjacent edges of a rectangle, when apply the rectangle shear transformation, the rectangle will be changed to parallelogram, the two adjacent edges are not perpendicular anymore.

1. There are two coordinate frames, FW = {w1, w2, OW} and FV = {v1, v2, OV } with v1 = w2, v2 = w1, and OV = OW +v1 +v2. Both coordinate frames are orthonormal.
2. Sketch out the two coordinate frames Fw and Fv. (4 pts)



1. Give the change of basis matrix mapping coordinates relative to FW to coordinates relative to FV. (6 pts)

( 0 1 0)

(1 0 0)

(1 1 1)

1. Give the transformation matrix for the geometric transformation that maps OW onto OV, w1 to v1 and w2 to v2. Assume that the coordinate representation of both the domain and range are relative to FW. (8 pts)

A: translate first, then rotate counter clockwise 90 degree, then mirror with y-axis.

The Sutherland-Hodgman polygon-clipping algorithm is described in the text in terms of a single clipping boundary. In practice we usually want to clip against some viewport. This involves passing the polygon through a series of clippers. For example:

POLYGON->TopClipper->BottomClipper->LeftClipper->RightClipper->RESULT

If we need to clip an edge at a boundary, an expensive intersection calculation must be performed. Prove, or give a counter-example to the statement: In general there is a BEST ordering of the clippers." (10 pts)

A: In general, there is not a BEST ordering of the clippers.

For example, if the vertices of a polygon are clustered in the outside of one clipper, it is best to do clipping using this clipper, to eliminate polygon edges as many as possible. However, we don’t know which clipper it is.

1. Which of the following are valid expressions in affine geometry and **why**, where are points, is a vector, and . (12 pts)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Expression | Valid? | Expression | Valid? | Expression | Valid? |
|  | Yes |  | yes |  | no |
|  | No |  | yes |  | no |
|  | Yes |  | yes |  | yes |
| ( | no |  | no |  | yes |

1. Consider a ray (P，v) that hits a surface S at a point Q, where the normal to S at Q is n, and where S has only diffuse reflection material components (dr, dg, db). Further, assume there is a white point light at position L with RGB intensity values (I, I, I), and a white ambient light with RGB intensity values (A, A, A). Describe how to compute the intensity of light that travels back along this ray and eventually reaches the eye. (10 pts)

A:

color c = trace(point p, vector d, int step)

{

  color local, reflected, transmitted;

  point q;

  normal n;

  if(step > max) return(background\_color);

q = intersect(p, d, status);

if(status==light\_source) return(color(I,I,I));

if(status==no\_intersection)

return(color(A,A,A));

n = normal(q);

r = reflect(q, n);

t = transmit(q,n);

local = phong(q, n, r); //using standard phong illumination model, the

// material components using (dr,dg,db)

reflected = trace(q, r, step+1);

transmitted = trace(q,t, step+1);

return(local+reflected+transmitted);

trace(p, v, 1);

1. Since a texture map is a fixed size image, it is often the case that a rendered pixel on the screen will be of very different resolution from the pixels in the texture map image. For example, a single screen pixel may actually cover multiple texture map pixels, when the object being projected to that pixel is far from the camera, so it is very small. Or, a single screen pixel may be much smaller than a texture map pixel when the object is close to the camera, so it is very large.

Both of these situations lead to problems in the resulting texture mapped image. Explain what the artifacts will look like in both cases, and give descriptions of techniques that could be used to minimize the artifacts. (10 pts)

A: when the first case happens, the projected texture will have aliasing, that is, the texture imaged will be shrink a lot and some continuous texture pattern will be lost or broken. To avoid this, the color of the projected pixel will be the averaged color of its corresponding region pixels.

When the second case happens, the texture will be enlarged heavily, thus there are apparent steps between neighbor pixels, called “stairs”. To avoid this, we should sample a region averaged color, not a separate pixel color in the texture and do some smooth filters on the projected image.

1. Devise a method for testing one planar polygon is fully on one side of another planar polygon; write out the source code in C language. (10 pts)

A:

Suppose two polygons are stored in p1[m] and p2[n], m,n are number of the two polygons’ vertices respectively.

bool oneside (p1, p2)

{

//construct the equation of the plane that p1 stays on.

Vector v=(p1[1]-p1[0])x(p1[m/2]-p1[0]);

//then the plane equation is //(x-p1[0].x)+(y-p1[0].y)+(z-p1[0].z)=0

//substitute all points of p[2] into the equation, if the values are all positive or all negative, then return true, otherwise return false.

float t=(p2[0].x-p1[0].x)+(p2[0].y-p1[0].y)+(p2[0].z-p1[0].z);

float t1=(p2[1].x-p1[0].x)+(p2[1].y-p1[0].y)+(p2[1].z-p1[0].z);

boolean s = t\*t1>0?true:false;

for(int i=0; i<m; i++){

t1= (p2[i].x-p1[0].x)+(p2[i].y-p1[0].y)+(p2[i].z-p1[0].z);

boolean s1 = t\*t1>0?true:false;

if s1&s==false

return false;

}

return true;

//end.

1. Assume that the call drawcube() will draw the vertices of a 1\*1\*1 cube centered at the origin. The camera is located at the position (0, 0, 100), is upright and aimed down the negative z-axis. Using drawcube() as the only drawing primitive, give the sequence of OpenGL calls affecting the modelview matrix needed to view a 3\*4\*1 box, oriented 45**°**to the x-axis and rotated -30**°**with respect to y-axis, and finally located with its center at (5, 12, 3). (10 pts)

A:

glPushMatrix();

gluLookAt(0, 0, 100, 0, 0, 0, 0, 1, 0);

glTranslate(5, 12, 3);

glRotate(45, 1, 0, 0);

glRotate (-30, 0, 1, 0);

glScale (3,4,1);

drawcube();

glPopMatrix();