October 28, 2008 Name _____

# CS445 Midterm Solutions

*Fall 2008*

| | | | |
|---|---|---|---|
| 1. | (max = 15) | 4. | (max = 14) |
| 2. | (max = 12) | 5. | (max = 20) |
| 3. | (max = 14) | 6. | (max = 25) |
| Final Score _____(max=100) | | | |

1) (15 pts) Discuss the difference between immediate mode and display lists in OpenGL. Roughly how does each work? Which one is faster and why?

OpenGL is based on a ***client-server model***.
Generally, the client refers to the cpu and main memory, and the server refers the gpu and display memory. However, these don't have to be on the same machine.

When an OpenGL command is executed (e.g. drawing a vertex with glVertex()), the command must be sent from the client to the server.

**Immediate mode**: One OpenGL command at a time is sent from the client to the server *every time* the command is executed. This results in a lot of overhead.

**Display Lists**: A display consists of a sequence of openGL commands. Initially, these commands are sent all at once over to the server and stored in server memory. This is time consuming but *is done only once*. When the commands are executed (e.g. the vertices are drawn), the openGL commands *are read directly from the server side* rather than the client side, and so *is very fast*.

2) (3 pts each, 12 pts total) True or False (Circle one)

a) T or F : The dot product of two vectors is 0 if they are perpendicular

b) T or F : The dot product of two unit vectors gives the sine of the angle between them.

c) T or F : The cross product of two vectors is 0 if they are perpendicular

d) T or F : In homogeneous coordinates, the expression (x,y,z,w) represents a point if w=1.

3) (7 pts each, 14 pts total) *2D Transforms*: What is the 3x3 matrix transform (in homogeneous coordinates) for the 2D transformation:
   a) Translation by $t_x$, $t_y$:

$$\begin{pmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{pmatrix}$$

   b) Rotation by 30 degrees:

$$\begin{pmatrix} \cos 30 & -\sin 30 & 0 \\ \sin 30 & \cos 30 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

4) (7 pts each, 14 pts total) *2D Transforms*: Give the sequence of matrix transformations (order matters!) that is required to do the following transformations. You do not need to give the actual matrix. Instead, identify each matrix using the syntax $S(s_x, s_y)$ for a scale, $R(\theta)$ for a rotation, $T(t_x, t_y)$ for a translation. For each, also give the sequence of transformations needed to obtain the inverse.

   a) Rotate by 10 degrees about the point at (-3,4)

      T(-3,4) R(10) T(3, -4)

      Inverse:

      T(-3,4) R(-10) T(3, -4)

   b) Scale by 5 along a positive 45 degree angle

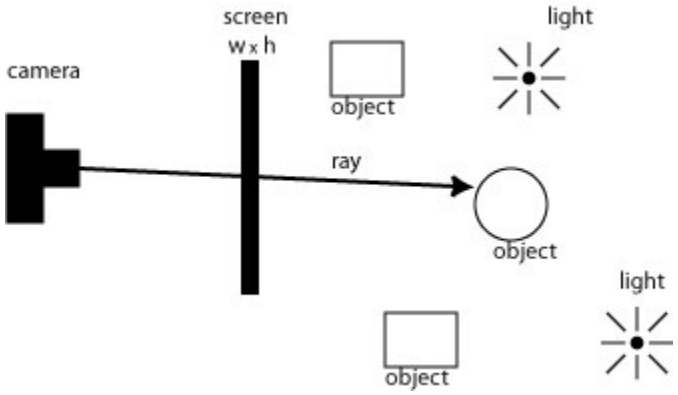
      R(45) S(5,1) R(-45) or R(-45) S(1,5) R(45)

      Inverse:

      R(45) S(1/5,1) R(-45) or R(-45) S(1,1/5) R(45)

5) (20 pts total) Ray Tracing:

a) (5 pts total) Draw a quick sketch showing and labeling the relationship of the different elements of a ray tracer, e.g the camera, etc.




b) (5 pts) What is the parametric equation of a ray that begins at a point $P_0$ and points in the direction of a point $P_1$?

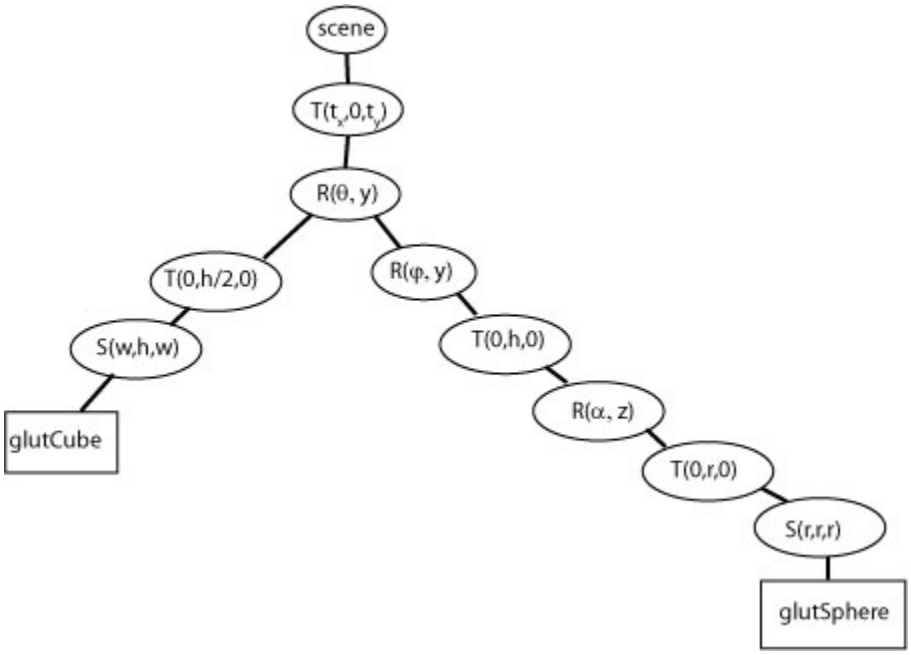
points on a ray are defined by $P = P_0 + t\,(P_1 - P_0)$

c) (10 pts) Give the *pseudocode* for a ray-tracer *without* shadows, reflection or refraction. Be sure it is clear what things mean (e.g. you might want to label them in your picture above).

```
For each pixel in image {
     compute ray from camera to pixel
     pixel color = Trace(ray)
}

color Trace(ray) {
     For each object
          find intersection (if exists)
          if intersection exists, determine if closest to camera
     If a closest object exists
          for each light
               color += color_local (phong model)
     return color
}
```

6) (25 pts total) Bob.
a) (15 pts) Draw a scene graph for Bob. See addendum for details.




Note, R(phi,y) could go in a number of locations on right branch. However, R(alpha,z) must sit between T(0,r,0) and T(0,h,0) in order shown.

b) (10 pts total) How would the Bob's scene graph translate into a sequence of openGL commands (don't worry too much about getting the syntax exactly right – get as close as you can). Assume you are in the display method and the camera has already been set:

```
float r, h, w; // radius, height, width; these are set elsewhere
float theta, phi, alpha, tx, tz; // these are set elsewhere by user
GLUT glut = new GLUT();
public void display(GLAutoDrawable d) {
        GL gl = d.getGL();
        gl.glClear(GL.GL_COLOR_BUFFER_BIT | GL.GL_DEPTH_BUFFER_BIT);
        gl.glMatrixMode(GL.GL_MODELVIEW);
        gl.glLoadIdentity();
        glu.gluLookAt(20.0, 10.0, 20.0, 0.0, 10.0, 0.0, 0.0, 1.0, 0.0);

        // YOUR CODE GOES HERE:
```

```
gl.glTranslate(tx,0, ty);
gl.glRotate(theta, 0,1,0);

// Body
gl.glPushMatrix();
gl.glTranslate(0,h/2.,0);
gl.Scale(w,h,w);
glut.glutCube);
gl.glPopMatrix()

// Head
gl.glPushMatrix();
gl.glRotate(phi, 0,1,0);
gl.glTranslate(0,h,0);
gl.glRotate(alpha, 0,0,1);
gl.glTranslate(0,r.,0);
gl.Scale(r,r,r);
glut.glutSphere();
gl.glPopMatrix()
```