CS354: Computer Graphics
Final

Please write your answers ON THESE SHEETS. Attach extra sheets if necessary. The size of the space indicates roughly the size of the answer we are expecting.

SHOW YOUR WORK; especially for calculations, you might make an arithmetic mistake but get partial credit for setting up the problem correctly.

Feel free to give a one or two sentence EXPLANATION for any question or any item, to explain your reasoning. Especially if you are not sure what the question means or if you feel it is ambiguous, an explanation of your assumptions and your decision might help us assign partial credit.

There is a total of 100 points.

1. (3 pts) **TRUE** or **FALSE**: The code fragment

   ```
   glRotatef(20, 1, 0, 0);
   glRotatef(30, 0, 0, 1);
   ```

   could always be replaced by

   ```
   glRotatef(30, 0, 0, 1);
   glRotatef(20, 1, 0, 0);
   ```

   without changing the image in any way.

2. (3 points) **TRUE** or **FALSE**: Under perspective projection, a rectangle in three-dimensional space will always be projected so that parallel faces appear parallel in the image.

3. (4 pts) Describe, in at most three sentences, a situation in which you are drawing three-dimensional graphics but you might want to disable z-buffering.

4. (4 pts) Here is an OpenGL material specification:
   glFloat amb [] ={0.0,0.5,0.5,1.0}
   glMaterialf(GL_FRONT,GL_AMBIENT,amb);
   glFloat diff [] ={0.0,0.0,0.0,1.0}
   glMaterialf(GL_FRONT,GL_DIFFUSE,diff);
   glFloat spec [] ={1.0,1.0,1.0,1.0}
   glMaterialf(GL_FRONT,GL_SPECULAR,spec);
   Circle the best statement; just pick one.
   a) This would be a good material specification for a blueish-gray stone.
   b) This would be a good material specification for an calm sea on a sunny day.
   c) This would be a good material specification for a deep black night sky.
   d) This would be a good material specification for a brown ceramic vase.

5. (4 points) Consider the command
glLightf(GL_LIGHT_0, GL_AMBIENT, 0.2, 0.2, 0.2, 1.0)
The ambient intensity of the light source (pick the single best answer)
a) is attenuated quadratically with distance
b) is attenuated linearly with distance
c) originates at the point $(0.2, 0.2, 0.2)$
d) is the same wherever the light is positioned

6. (4 pts) Recall that by default OpenGL expects unit length normal vectors. Assume we are using this default, and we define a material with only diffuse lighting. A vertex assigned a normal with the command
glNormalf(0.2,0.5,0.3);
will appear to be
a) too dark.
b) too bright.
c) too dark when the object is in some positions and too bright in other positions
d) just fine; this is a unit normal vector.

7. (4 points) Loading the following matrix into the OpenGL *PROJECTION* matrix

$$\begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \end{bmatrix}$$

(check all that apply)
a) defines a world-coordinate view volume which is a 2D rectangle, not a frustum or a box.
b) would not be a good idea if we intend to use the depth buffer.
c) produces an orthographic projection.
d) might have been accomplished by a call to *glFrustum*.

8. (4 pts) Which of the following sets of homogeneous coordinates represent the same point? Group together the names of the vectors which are the same, ie. $(a, b, c), (d, e)$.

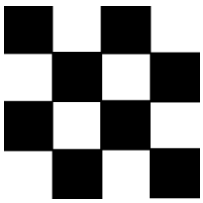
$$a = \begin{bmatrix} 1 \\ 3 \\ 2 \\ 1 \end{bmatrix} b = \begin{bmatrix} 1 \\ 3 \\ 2 \\ 0 \end{bmatrix} c = \begin{bmatrix} 2 \\ 6 \\ 4 \\ 0 \end{bmatrix} d = \begin{bmatrix} 2 \\ 6 \\ 4 \\ 1 \end{bmatrix} e = \begin{bmatrix} 2 \\ 6 \\ 4 \\ 2 \end{bmatrix}$$

9. (4 points) Increasing the value for the *GL_SHININESS* parameter of a material specification will (circle all that appy):
   a) Increase the ratio of specular to diffuse reflectivity
   b) Decrease the ratio of specular to diffuse reflectivity
   c) Increase the brightness of diffuse hilights
   d) Decrease the brightness of diffuse hilights
   e) Increase the size of specular hilights
   f) Decrease the size of specular hilights

10. (6 points) Which of the following values are computed at the vertices and then interpolated across a polygon during rasterization? Circle all that apply:
   a) world coordinate system x values
   b) window coordinate system y values
   c) colors from *glColor3f*
   d) colors determined by lighting calculations
   e) colors from texture-map look-ups
   f) texture coordinates
   g) normals

11. (8 points) What is the outward-facing normal to the triangle with vertices (in counter-clockwise order) $(0.0, 1.0, 2.0)$, $(3.0, 1.0, 2.0)$, and $(3.0, 3.0, 0.0)$?

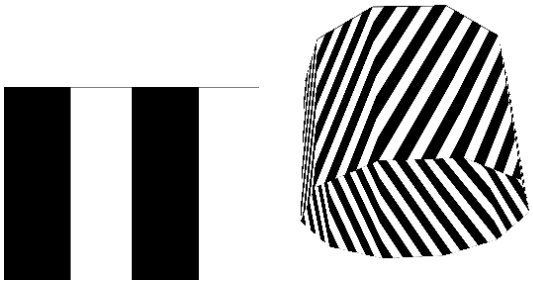
12. (6 points)



A tiny 4x4 texel black and white image is used as a texture; a blown-up version is shown above. The magnification filter is set to *GL_LINEAR* (bilinear filtering), and the texture is defined to repeat in both the s and t directions, with *glTexParameter*. A vertex is assigned texture coordinates with the line:
glTexCoord2f(2.35,0.35);
What color is assigned to the vertex by this command?

13. (8 points) Give a parametric definition for the sphere of radius one centered at the origin, as three functions $X(u,v), Y(u,v), Z(u,v)$. You may choose the parameters u and v in any way you find convenient.

14. (10 points)



On top is a texture consisting of two stripes. below is a cylinder, made up of ten quadrialterals, with the texture applied to it. There are 30 vertical stripes on the cylinder, and a vertical line on the cylinder crosses four stripes. Below is a (rather inefficient) code fragment which draws the cylinder, without the commands producing the texture coordinates. Give the *glTexCoords* commands, and indicate where they go in the code. Recall the syntax:
glTexCoord2f(Glfloat scoord, GLfloat tcoord);
You can assume the texture is set up to repeat.

```
for (i=0; i<10; i++) {

angle1 = i*2.0*M_PI/10.0;

angle2 = (i+1)*2*M_PI/10.0;

glVertex3f(cos(angle1), sin(angle1), 1.5);

glVertex3f(cos(angle2), sin(angle2), 1.5);

glVertex3f(cos(angle2), sin(angle2), 0.0);

glVertex3f(cos(angle1), sin(angle1), 0.0);

    }
```

15. (12 points) In the new "Finding Nemo" game we are trying to write, a fish is swimming in a tank. Moving the mouse with any mouse button depressed moves the fish around in the tank, in the direction of the mouse motion (so the fish always moves in a direction of the form $(x, y, 0, 0)$). The camera is fixed at a location outside the tank, looking in at the fish, but as the fish moves the camera rotates to track the fish and keep it exactly in the center of the picture.

    Below are some functions which might be used in an OpenGL/glut version of this operator interface, and the *display* function which draws each frame. The parts of the functions that actually perform the motions are missing. You are to fill them in. You may add any global variables that you need in the region indicated. Here are some OpenGL calls which you might want to use; if you need others and you are not sure of the syntax just make an assumption about the syntax, write it down, and go ahead and use it; syntax errors will not be counted against you if the assumed syntax is reasonable.

    Do not worry about turning the fish so that she faces forward as she moves; just get the camera pointed at her.
    glRotatef(GLfloat angle_in_degrees, GLfloat x, GLfloat y, GLfloat z)
    glTranslatef(GLfloat x, GLfloat y, GLfloat z)
    glMultMatrixf(Glfloat c) - c points to an array of 16 floats, eg, GLfloat c[16]*
    glGetFloatv(GL_MODELVIEW_MATRIX, GLfloat c)*
    glPushMatrix()
    glPopMatrix()
    glLoadIdentity()
    glMatrixMode(GL_MODELVIEW)

    Begin with a few sentences describing the idea of your user interface program, just in case it is not obvious to the grader.

```
/* Put any global variable definitions you need here*/
```

```
/* Called when mouse buttons are pressed */
void motion(int button, int up_or_down, int cursorX, int cursorY) {
     switch(up_or_down} {
          case GLUT_UP: // mouse button released
          // Add code here if necessay
```

```
            break;
            }
           case GLUT_DOWN: // mouse button pressed
            // Add code here if necessay



            break;
            }
    glutPostRedisplay();
}


/* Called when mouse is moved with any button down */
void motion(int cursorX, int cursorY) {
     // Add code here if necessay







     glutPostRedisplay();
}



/* Called to draw frame */
void display () {
     // Add code here if necessary



     drawTank(); // Draws fish tank, background, and stationary contents
     // Add code here if necessary



     drawFish(); // Draws moving fish
}
```