Computer Graphics
50:198:456/56:198:556 (Spring 2009)

Homework:	**3**	**Professor:**	**Suneeta Ramaswami**
Due Date:	**3/9/09**	**E-mail:**	`rsuneeta@camden.rutgers.edu`
Office:	**321 BSB**	**URL:**	`http://crab.rutgers.edu/~rsuneeta`
		Phone:	**(856)-225-6439**

Written Assignment #1

There are 7 questions in all, each with (approximately) the same weight. Undergraduate students are required to answer questions 1-5, and may answer questions 6 and 7 for extra credit. Graduate students are required to answer all 7 questions. Please show all your work; answers without justification will not receive any credit. Turn in a legible copy of your answers **on or before March 9, 2009.**.

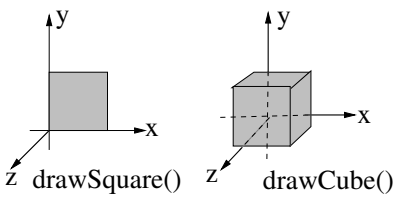
1. Given a point $P = (p_x, p_y, 1)^T$ in the plane and an angle θ, derive a transformation that rotates *the plane* by θ degrees clockwise (not counterclockwise) about the point P. Express your answer as a 3×3 matrix, so that it can be applied to a column matrix in homogenous coordinates. Show how you derived your answer.

2. A very common low-level operation in computer graphics is testing if a point lies on or inside a triangle. One way to do this is by using what we know about affine combinations of points, described below (also discussed in class):

   Given three points P_0, P_1, and P_2 (assume they are non-collinear, *i.e,* they do not all lie on a straight line), an affine combination of these points is the point Q given by $Q = \alpha_0 P_0 + \alpha_1 P_1 + \alpha_2 P_2$, where $\alpha_0 + \alpha_1 + \alpha_2 = 1$. Q lies on or inside the triangle defined by P_0, P_1 and P_2 if and only if each of α_0, α_1 and α_2 is ≥ 0. (If any of them is < 0, then Q lies outside the triangle.)

   Suppose $P_0 = (1, 1), P_1 = (3, 2)$ and $P_2 = (4, 5)$, how would you determine if an arbitrary point $S = (x, y)$ lies inside the triangle $\triangle(P_0, P_1, P_2)$? Derive the necessary equations.

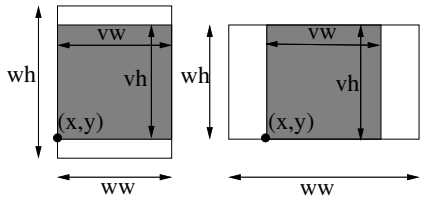
3. Given a unit cube with one corner at $(0, 0, 0)$ and the opposite corner at $(1, 1, 1)$, derive the transformations necessary to rotate the cube by θ degrees about the main diagonal (from $(0, 0, 0)$ to $(1, 1, 1)$) in the counterclockwise direction when we are looking along the diagonal toward the origin. You may express your answer as a concatenation of transformations (as we did in class).



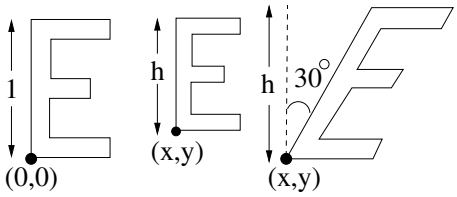
4. Suppose you are given a function `drawSquare()`, which draws a unit square (defined as a `GL_POLYGON`) with its lower left corner at the origin. See figure above. This is drawn on the $z = 0$ plane, with the z axis pointing out of the page. Use the procedure `drawSquare()`

and other OpenGL functions (e.g., `glPushMatrix()`, `glRotate*()`, `glTranslate*()` etc.) to produce a procedure called `drawCube()` that draws a unit cube centered at the origin (shown on the right in figure below).

5. Suppose that you have a square idealized drawing region (height equals width). The user has just resized the window so that it now has width w_w and height w_h. As a function of w_w and w_h, derive the arguments for `glViewPort()` so that the new viewport is the *largest* square that fits within the window and is *centered* within the window. (Refer to the figure below, where the outer rectangle is the graphics window and the shaded rectangle is the viewport.) Recall that the calling sequence is `glViewport(x, y, vw, vh);`, where `(x,y)` are the coordinates of the lower left corner of the viewport (the origin is the lower left corner of the window), and `vw` and `vh` are the width and height, respectively, of the viewport.



6. Assume that $z = 0$ and we are using `gluOrtho2D` for viewing. Suppose that you have an OpenGL procedure `drawE()`, which draws an upper-case letter 'E' of height 1, so that its lower left corner coincides with the origin. Show how to achieve each of the following tasks using OpenGL. (Refer to the figure below.) Assume that the current transformation mode is `GL_MODELVIEW`. You may call the procedure `drawE()`, but you may not modify its contents. On return, the OpenGL transformation stack should be unchanged.

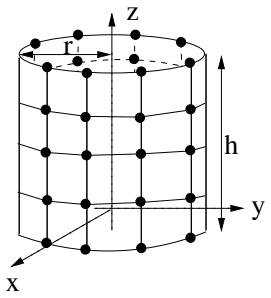


   (a) Give code for a procedure `drawE1(x,y,h)` which draws the letter 'E' so that its lower left corner is at position (x, y) (and $z = 0$) and its height is h. All three arguments are of type `GLfloat` and h is positive. Briefly explain. Make sure there is no distortion in the rendering of 'E'.

   (b) Give code for a procedure `drawE2(x,y,h)` which draws an italic letter '*E*' by slanting the letter by 30 degrees to the right. Again the lower left corner is at (x, y) and the height is h. *Hint:* There is no OpenGL transformation which performs a shear, so you will need to derive the corresponding matrix and use `glMultMatrix` in the code. Recall that $\cos 30^\circ = \sqrt{3}/2$ and $\sin 30^\circ = 1/2$.

7. You are asked to write a procedure to generate a rendering of a cylinder in OpenGL. The cylinder is centered along the z-axis, has a height of h units, and has a radius of r units. Because OpenGL can only display polygons, you are to split the cylinder into v_s vertical

stacks (along the z-axis), and r_s radial slices (around the z-axis). For example, in the figure below, $v_s = 4$ and $r_s = 8$. Draw each face as a `GL_POLYGON`.



Give a procedure (in pseudocode):

```
void cylinder(float h, float r, int vs, int rs);
```

to draw such a cylinder in OpenGL. You may **not** use any GLUT procedures. You do not need to draw the top and bottom of the cylinder.