SBE3130 Computer Graphics
Fall 2021

# Sheet 1
# Transformations

1. Triangle ABC has vertices at (1,1), (2,3), and (3,1). Find the matrix that performs a scale of this triangle by 2 in the x direction while keeping vertex (2,3) fixed. Apply this matrix to the vertices and find the new vertices of ABC.

2. For each sequence of OpenGL like commands, sketch the resulting figure knowing that the function drawSquare() draws a square at origin with a width of 1.

a.

```
drawSquare();
glTranslate(0,1,0);
glScale(0.5,0.5,1);
drawSquare();
glTranslate(0,1,0);
glScale(0.5,0.5,1);
drawSquare();
```

b.

```
drawSquare();
glPushMatrix();
    glTranslate(0,1,0);
    glScale(0.5,0.5,1);
    drawSquare();
glPopMatrix();
glPushMatrix();
    glTranslate(0,2,0);
    glScale(0.5,0.5,1);
    drawSquare();
glPopMatrix();
```

3. Give the 3D homogeneous matrix for translation by (3, -4, 2). Then, give its inverse transformation matrix.

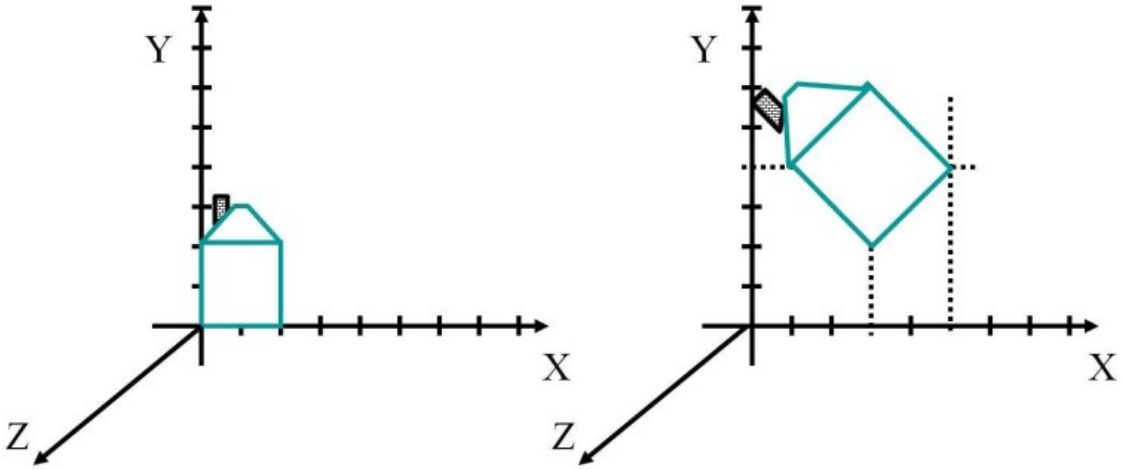
4. Give the 3D homogeneous matrices for reflection about line x = -3.

5. Describe in words what the following matrix does and be specific about the order of operations:

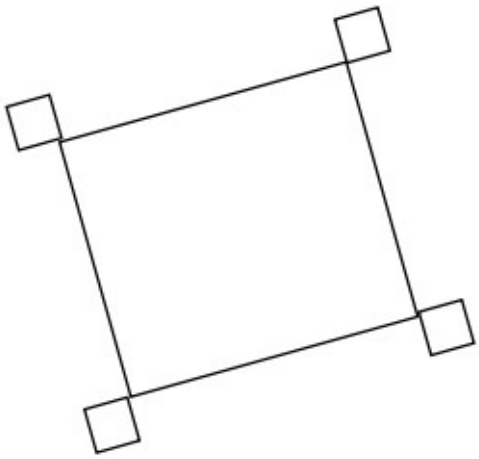
$$\begin{bmatrix} 0.707 & 0 & 0.707 & 0 \\ 0 & 7 & 0 & 0 \\ -0.707 & 0 & 0.707 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

6. Express the homogeneous point P(x,y,z,w) = P(2,1,3,0.5) in Cartesian coordinates

7. Give the series of affine transformations (assuming post-multiplying) needed to create the picture on the right hand side assuming the house started from the position shown on the left hand side. Give the OpenGl commands

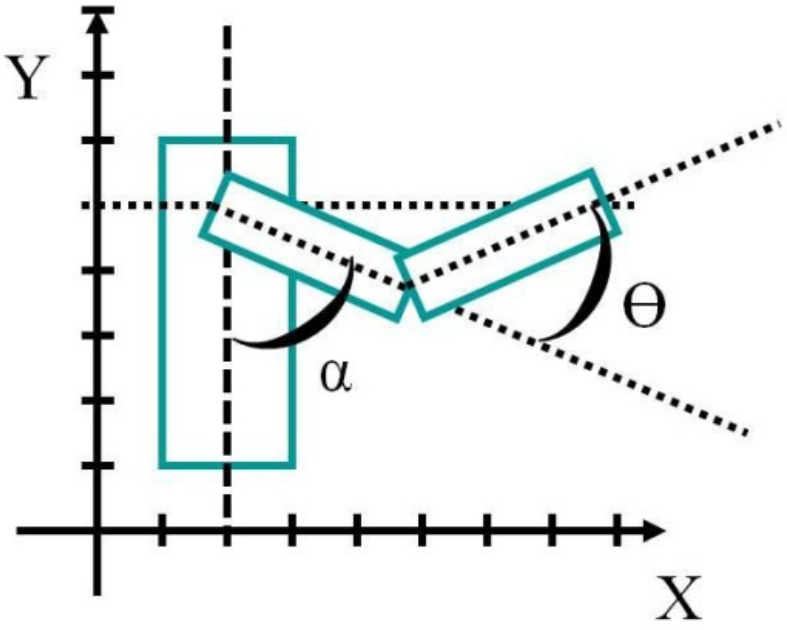



8. Suppose you have a function called drawSquare() that renders a 2D square centered at location (0,0) with a width and height of 1. Using OpenGL commands and the function drawSquare() show how to render the figure below



The figure should be centered at the position (5,5) and the width and height of the small squares is 1 and the large square has width and height of 6. The entire figure is rotated 15 degrees. Here is the syntax for the basic OpenGL transformations
glTranslate(tx,ty,tz)  glRotate(theta,vx,vy,vz)  glScale(sx,sy,sz)

9. Give the code necessary to draw the following robotic arm, using OpenGL function calls and calls to a draw-block routine, DB(). The DB() procedure draws a block of unit size having vertices at (0,0), (1.0), (1,1), and (0,1). You can assume that the model view matrix has been appropriately initialized to reflect the given XY coordinate system. Your code should leave the modelview matrix unaltered upon completion. The blocks that compose the arm are both 3x1 in size.



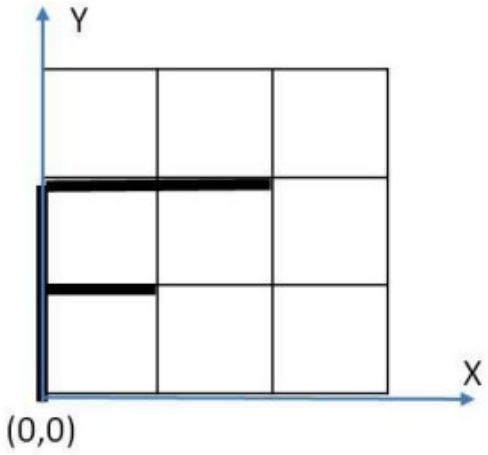
10. Draw shapes 1, 2, 3, 4, and 5 transformed by the below OpenGL commands

```
void drawShape(void)
{
glBegin(GL_POLYGON);
        glVertex3f(0.0, 0.0, 0.0);
        glVertex3f(2.0, 0.0, 0.0);
        glVertex3f(2.0, 1.0, 0.0);
        glVertex3f(1.0, 1.0, 0.0);
        glVertex3f(1.0, 3.0, 0.0);
        glVertex3f(0.0, 3.0, 0.0);
glEnd();
}


glPushMatrix();
        drawShape();                    // Shape 1
        glRotatef(90, 0, 0, 1);
        glTranslatef(1, 0, 0);
        drawShape();                    // Shape 2
        glRotatef(-90, 0, 0, 1);
        glPushMatrix();
                glTranslatef(1, 0, 0);
                drawShape();            // Shape 3
                glScalef(1, 2, 1);
                glTranslatef(1, -2, 0);
                drawShape();            // Shape 4
                glTranslatef(-2, 2, 0);
                glRotatef(90, 0, 0, 1);
                glTranslatef(-1, 1, 0);
                glScalef(1, 0.5, 1);
                glRotatef(90, 0, 0, 1);
        glPopMatrix();
        glTranslatef(-2, 1, 0)
        drawShape();                    // Shape 5
glPopMatrix();
```

11. Draw the output of the given OpenGL code below. You can assume that the modeling matrix is initialized to the identity matrix to begin with. Label each drawn “F” with the corresponding number from 1-6.

**The output of Draw-F()**




```
glTranslatef(2.0, -1.0, 0.0);
Draw-F();                          // 1
glPushMatrix();
glRotatef(-90.0, 0.0, 0.0, 1.0);
Draw-F();                          // 2
glTranslatef(-4.0, -3.0, 0.0);
Draw-F();                          // 3
glPopMatrix();
glPushMatrix();
glTranslatef(-4.0, 0.0, 0.0);
Draw-F();                          // 4
glScalef(-0.5, 0.5, 0.5);
Draw-F();                          // 5
glTranslatef(-2.0, -2.0, 0.0);
Draw-F();                          // 6
glPopMatrix();
```