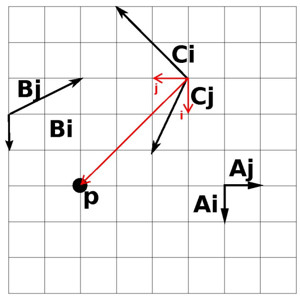
CPSC 314, Written Homework 1: Transformations

solution and grading criteria by Yufeng Zhu
Value: 4% of final grade
Total Points: 101

1. (15 pts) The point coordinate P can be expressed as (2,3): that is, P = 2*i + 3*j, where i and j are basis vectors of unit length along the x and y axes, respectively, with an origin at the lower left of the grid. Describe the point P in terms of the three other coordinate systems given below (A, B, C).



Answer:

A: p(0, -4) [5 pts, 3 pts if only one tuple is correct]

B: p(3, 1) [5 pts, 3 pts if only one tuple is correct]

C: p(0.5, 2) [5 pts, 3 pts if only one tuple is correct]

Solution:

The most intuitive way is to figure out the linear combination of the basis vectors from the graph. Here is one alternative way which might also be useful. As the basis vectors, for instance $\mathbf{C}_i$ and $\mathbf{C}_j$, sometimes are neither orthogonal nor unit vectors, it's hard to get the coefficient directly. Instead, we can set up an orthogonal basis coordinate frame, say **i** and **j** as shown in the graph, whose origin is the same as $\mathbf{C}_i$ and $\mathbf{C}_j$'s. Then we can represent vector $\mathbf{C}_i$, $\mathbf{C}_j$ and **p** in frame **i** and **j**. In this case, they are

$$\begin{aligned} \mathbf{C}_i &= -2\mathbf{i} + 2\mathbf{j} \\ \mathbf{C}_j &= 2\mathbf{i} + \mathbf{j} \\ \mathbf{p} &= 3\mathbf{i} + 3\mathbf{j} \end{aligned} \tag{1}$$

Then we want a linear combination $\mathbf{p}'$ of vectors $\mathbf{C}_i$ and $\mathbf{C}_j$, which gives us **p**. Based on the definition of matrix vector product, we have the following equation

$$\begin{aligned} &\begin{pmatrix} \mathbf{C}_i & \mathbf{C}_j \end{pmatrix} \mathbf{p}' = \mathbf{p} \\ &\mathbf{C}_i = \begin{pmatrix} -2 \\ 2 \end{pmatrix}, \mathbf{C}_j = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \mathbf{p} = \begin{pmatrix} 3 \\ 3 \end{pmatrix} \\ &\begin{pmatrix} -2 & 2 \\ 2 & 1 \end{pmatrix} \mathbf{p}' = \begin{pmatrix} 3 \\ 3 \end{pmatrix} \end{aligned} \tag{2}$$

By solving the last linear equations using software like *Matlab* or *Mathematica*, we can get $\mathbf{p}'$ which is the linear combination coefficient we are looking for.

2. (3 pts) Write down the 4x4 matrix for translating an object by 2 in y, 3 in x, and 4 in z.

Answer:

The transformation matrix is $\begin{pmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 1 \end{pmatrix}$. [3 pts, no partial mark]

3. (8 pts) Give the OpenGL commands required to encode M. You may assume the matrix stack has been initialized with `glIdentity()`.

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

   Answer:

   glTranslatef(1,1,1);

   glRotatef(90, 0, 1, 0):

   [3 pts for each function call with correct function name and parameters, 2 pts for correct order]

4. (4 pts) Homogenize the point (4,4,6,2).

   Answer:

   (2, 2, 3, 1) [4 pts, no partial mark]

5. (16 pts) Give the 4x4 OpenGL modelview matrix at the four lines A, B, C, and D below.

```
glLoadIdentity();
glTranslate(1,1,0);
A
glRotate(90, 1,0,0);
B
glPushMatrix();
glScale(1,2,1);
glTranslate(1,1,0);
C
glPopMatrix();
glTranslate(1,1,0);
D
```

   Answer:

   A = $\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

   B = $\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & -1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

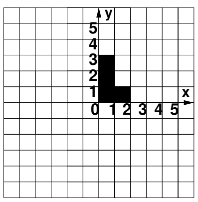
   C = $\begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 0 & -1 & 1 \\ 0 & 2 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

   D = $\begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 0 & -1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

   [4 pts for each matrix. Correctness will base on the transformation between each step instead of exact matching to the correct result.]

6. (55 pts) For each equation below, sketch the new location L' of the L shape on the grid and provide the OpenGL sequence needed to carry out those operations. Use the function `drawL()`, which draws an L shape with the lower left corner at the current origin as shown below. You may assume the matrix mode is `GL_MODELVIEW` and that the stack has been initialized with `glLoadIdentity()`. For reference, the OpenGL command syntax is `glRotatef(angle,x,y,z)`, `glTranslatef(x,y,z)`, `glScalef(x,y,z)`. Show your partial work, with the position that the L would be drawn after each matrix multiplication.

   Update 18 Jan: Do these computations in both directions: from left to right (moving coordinate frame), and also from right to left (moving object). You will get different intermediate answers, but the final position of the L should be the same each way; it's a good way to cross-check your work! You'll want to print out two copies of this writeup. Make sure to indicate which direction you're doing the computation for each set of pages. You don't need to rewrite the OpenGL commands on the second set, once is enough.

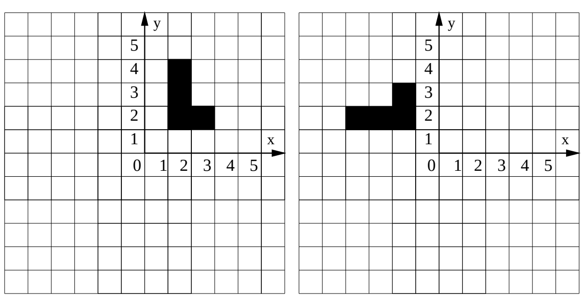


$$\mathbf{A} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{glScalef(1,2,1)}, \mathbf{B} = \underbrace{\begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{glRotatef(90,0,0,1)}, \mathbf{C} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{glTranslatef(1,1,0)}, \mathbf{D} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{glTranslatef(0,-3,0)}$$

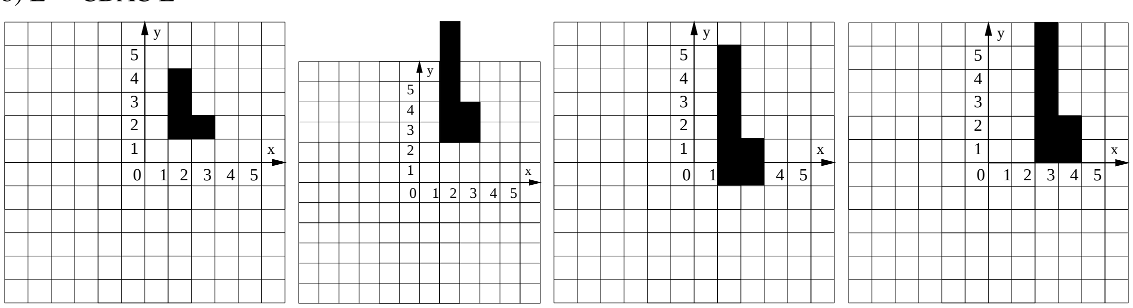
Answer:

Right to Left (moving object):

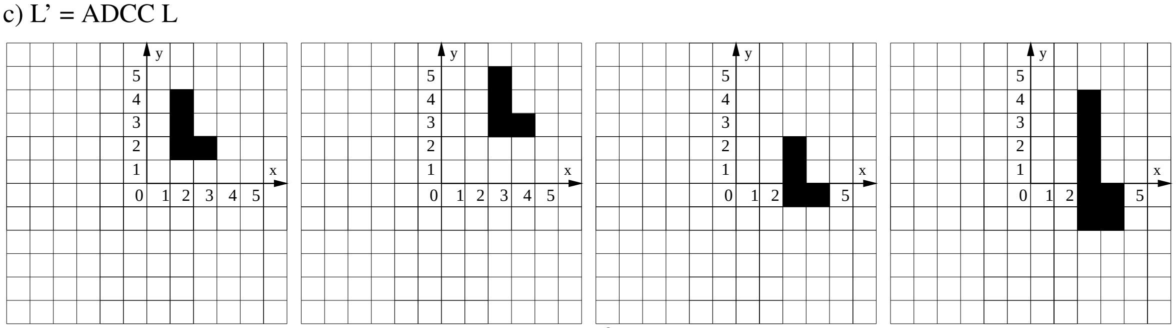
a) L' = BC L



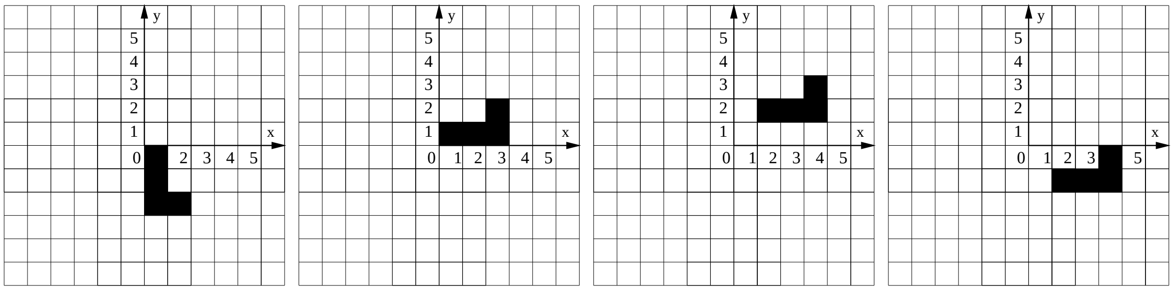
b) L' = CDAC L



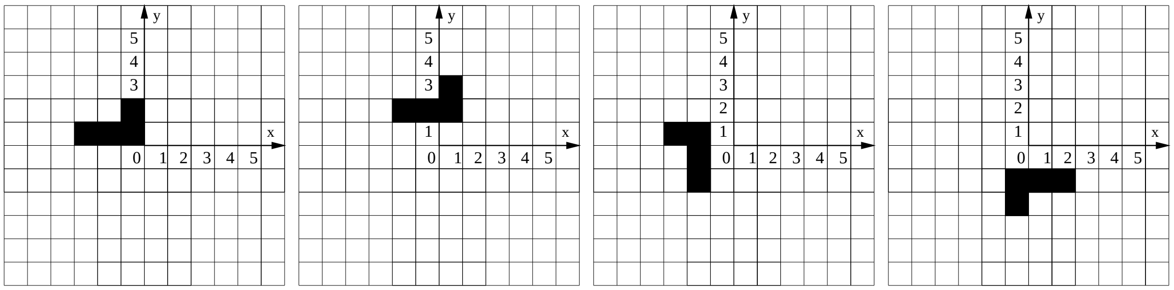
c) L' = ADCC L



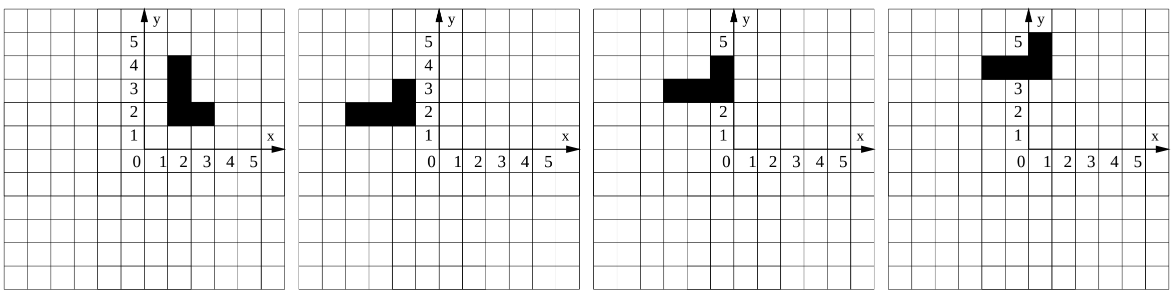
d) L’ = DCBD L



e) L’ = BBCB L



f) L’ = CCBC L

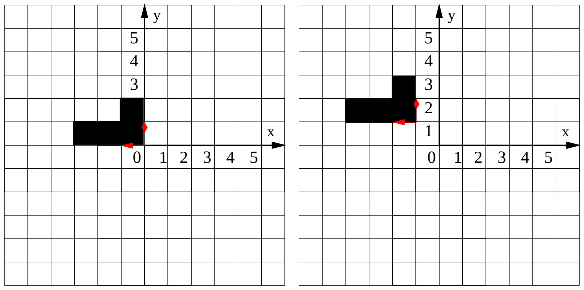


Left to Right (moving frame):

a) L' = BC L

glRotatef(90,0,0,1);

glTranslatef(1,1,0);



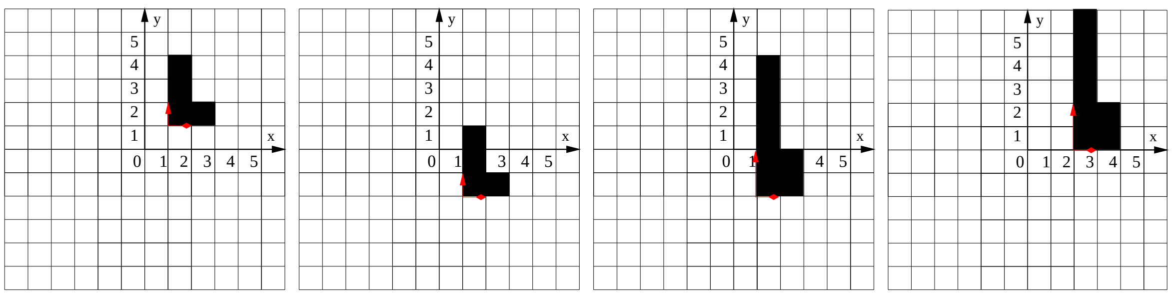
b) L' = CDAC L

glTranslatef(1,1,0);

glTranslatef(0,-3,0);

glScalef(1,2,1);

glTranslatef(1,1,0);



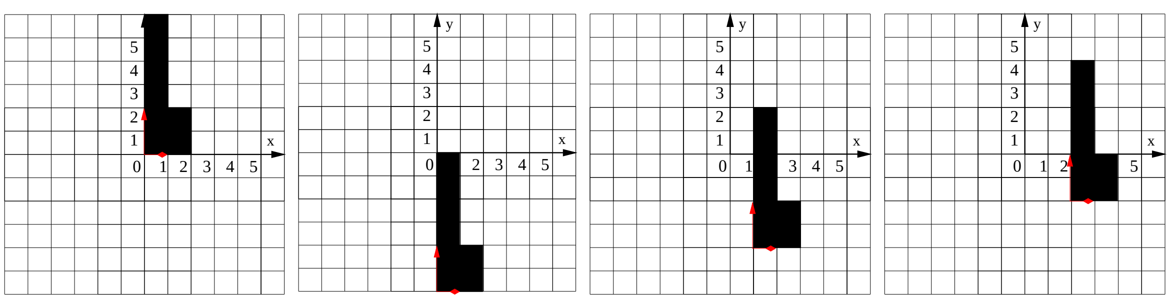
c) L' = ADCC L

glScalef(1,2,1);

glTranslatef(0,-3,0);

glTranslatef(1,1,0);

glTranslatef(1,1,0);



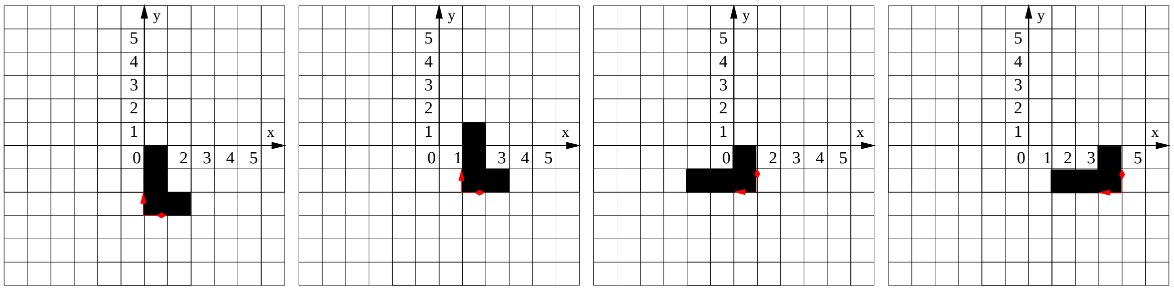
d) L’ = DCBD L

glTranslatef(0,-3,0);

glTranslatef(1,1,0);

glRotatef(90,0,0,1);

glTranslatef(0,-3,0);



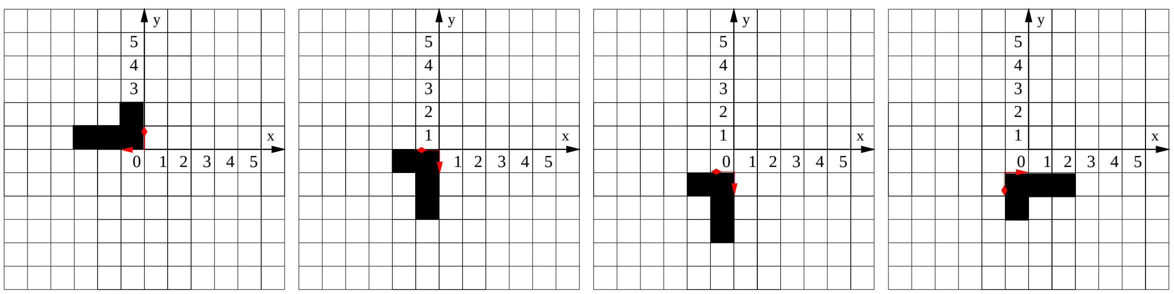
e) L’ = BBCB L

glRotatef(90,0,0,1);

glRotatef(90,0,0,1);

glTranslatef(1,1,0);

glRotatef(90,0,0,1);



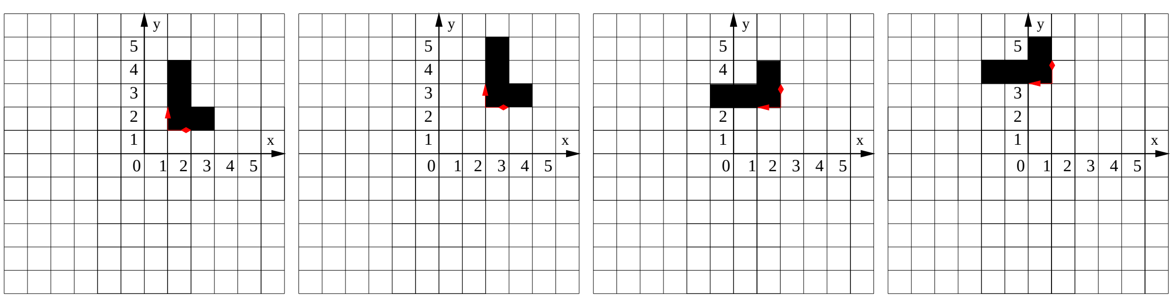
f) L’ = CCBC L

glTranslatef(1,1,0);

glTranslatef(1,1,0);

glRotatef(90,0,0,1);

glTranslatef(1,1,0);



[1 pt for each graph and 0.5 pt for each OpenGL function]