CPSC 314, Written Homework 1: Transformations

Out: Wed 20 Jan 2016
Due: Wed 27 Jan 2016 2pm (hand in at start of lecture)
Value: 4% of final grade
Total Points: 100

Name: solution

Student Number: ____________________

$Q1$		$/15 + 2extra$
$Q2$		$/3$
$Q3$		$/4$
$Q4$		$/16$
$Q5$		$/8$
$Q6$		$/54$
$Total$		$/100 + 2extra$

Please check one of the following:

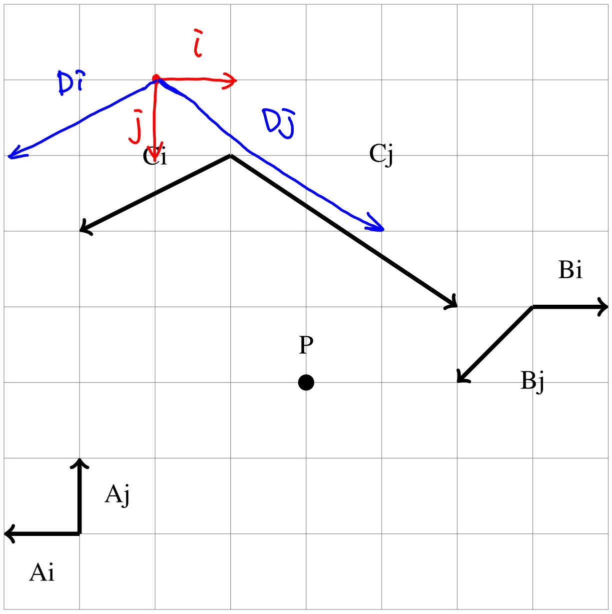
☐ I did not collaborate with anyone in the completion of this homework.
☐ I collaborated with people named below in the completion of this problem set:

Name: ____________________ Student Number: ____________________

Name: ____________________ Student Number: ____________________

Name: ____________________ Student Number: ____________________

1. (15 pts + 2 pts extra) The point coordinate P can be expressed as (4,3): that is, P = 4*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(-3, 2) [5 pts, 3 pts if only one tuple is correct]

B: p(-2, 1) [5 pts, 3 pts if only one tuple is correct]

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

D: p(8/7, 10/7) [2 pts extra credit if 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 $\mathbf{D}_i$ and $\mathbf{D}_j$, 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 $\mathbf{D}_i$ and $\mathbf{D}_j$'s. Then we can represent vector $\mathbf{D}_i$, $\mathbf{D}_j$ and **p** in frame **i** and **j**. In this case, they are

$$\begin{aligned} \mathbf{D}_i &= -2\mathbf{i} + \mathbf{j} \\ \mathbf{D}_j &= 3\mathbf{i} + 2\mathbf{j} \\ \mathbf{p} &= 2\mathbf{i} + 4\mathbf{j} \end{aligned} \tag{1}$$

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

$$\begin{gathered} \begin{pmatrix} \mathbf{D}_i & \mathbf{D}_j \end{pmatrix} \mathbf{p}' = \mathbf{p} \\ \mathbf{D}_i = \begin{pmatrix} -2 \\ 1 \end{pmatrix}, \mathbf{D}_j = \begin{pmatrix} 3 \\ 2 \end{pmatrix}, \mathbf{p} = \begin{pmatrix} 2 \\ 4 \end{pmatrix} \\ \begin{pmatrix} -2 & 3 \\ 1 & 2 \end{pmatrix} \mathbf{p}' = \begin{pmatrix} 2 \\ 4 \end{pmatrix} \end{gathered} \tag{2}$$

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 scale an object by 1 in y, 3 in x, and 2 in z.

Answer:

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

3. (4 pts) Homogenize the point (6,3,0,3).

Answer:

(2, 1, 0, 1) [4 pts, ~~no partial mark~~][3 pts if the result is not a homogenous point]

4. (16 pts) Give the 4x4 modelview matrix at the four lines A, B, C, and D in the pseudocode below. Assume the matrix stack has been initialized with LoadIdentity(). The transformation direction goes left to right.

```
LoadIdentity(); = I
translation(1,0,0); = T(1,0,0)
A
rotation(90,0,0,1); = R(90°,0,0,1)
B
scale(2,1,3); = S(2,1,3)
C
translation(0,1,2); = T(0,1,2)
D
```

Answer:

$$A = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad A = I * T(1,0,0) = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$B = \begin{pmatrix} 0 & -1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad B = A * R(90°,0,0,1) = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$C = \begin{pmatrix} 0 & -1 & 0 & 1 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad C = B * S(2,1,3) = B * \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$D = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 6 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad D = C * T(0,1,2) = C * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

5. (8 pts) Give the pseudocode required to encode M with left-to-right direction. You may assume the matrix stack has been initialized with `LoadIdentity()`.

$$\mathbf{M} = \begin{bmatrix} 2 & 0 & 0 & 3 \\ 0 & 2 & 0 & 1 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Answer:

translation(3,1,2);
scale(2, 2, 1):

or

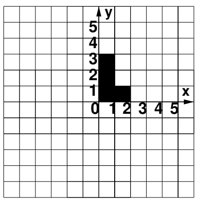
scale(2,2,1)
translation(1.5, 0.5, 2)

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

6. (54 pts) For each equation below, sketch the new location L' of the L shape on the grid and provide the pseudocode sequence needed to carry out those operations. You may assume the matrix mode is `mvMatrix` and that the stack has been initialized with `LoadIdentity()`.

For reference, the pseudocode transformation is `scale(x,y,z)`, `rotation(theta,x,y,z)`, `translation(x,y,z)`. Show your partial work, with the position that the L would be drawn after each transformation.

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 don't need to rewrite the pseudocode from right to left, once is enough.

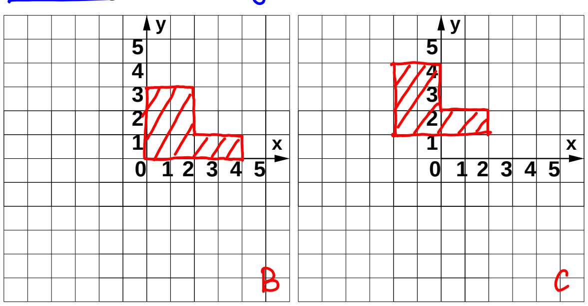


translate(1,0,0) scale(2,1,1) translate(-1,1,0) rotate(-90°,0,0,1)

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{D} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

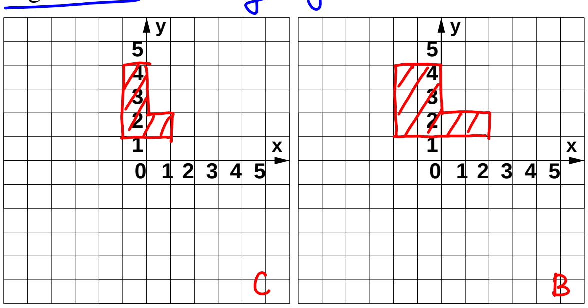
a) L' = BC L

Left to Right: moving coord



scale(2,1,1)
translate(-1,1,0)

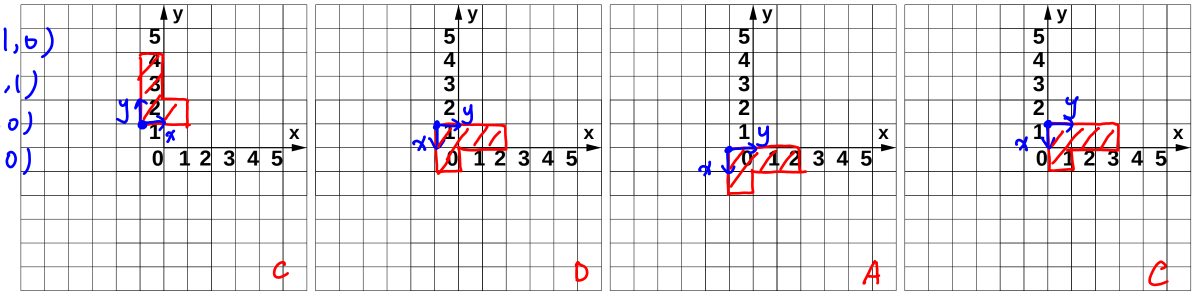
Right to Left: moving object



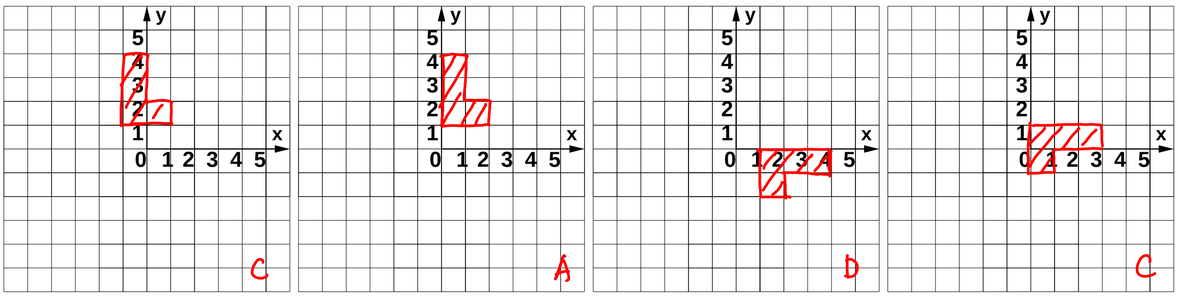
b) L' = CDAC L

Left to Right:

translate(-1,1,0)
rotate(-90°,0,0,1)
translate(1,0,0)
translate(-1,1,0)



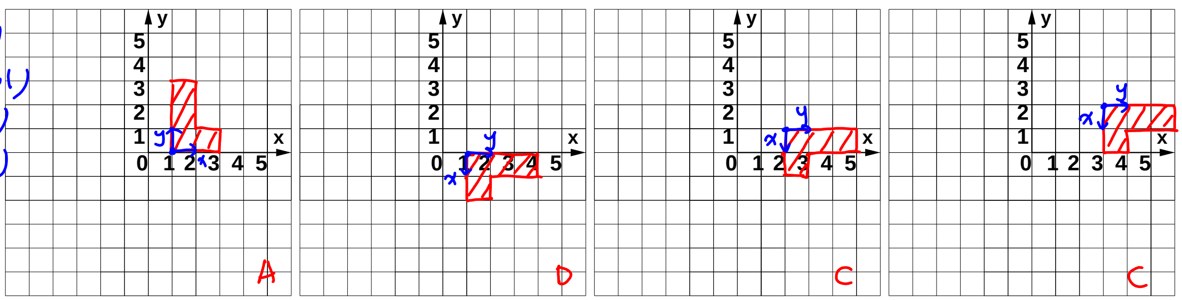
Right to Left:



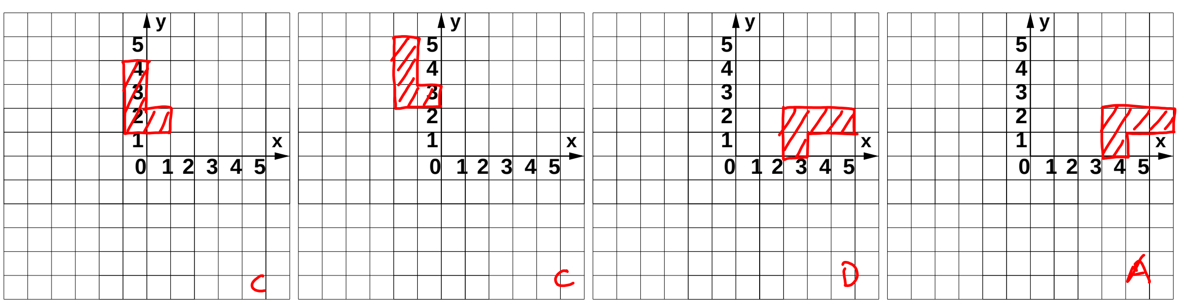
c) L' = ADCC L

Left to Right:

translate(1,0,0)
rotate(-90°,0,0,1)
translate(-1,1,0)
translate(-1,1,0)



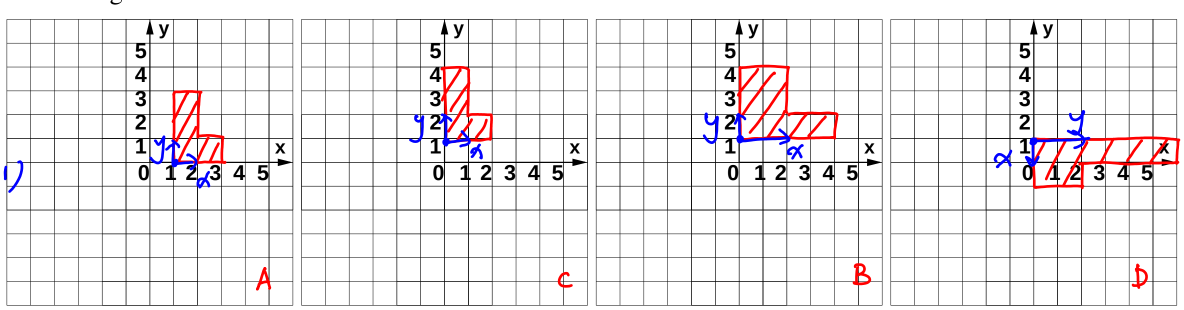
Right to Left:



d) L' = ACBD L

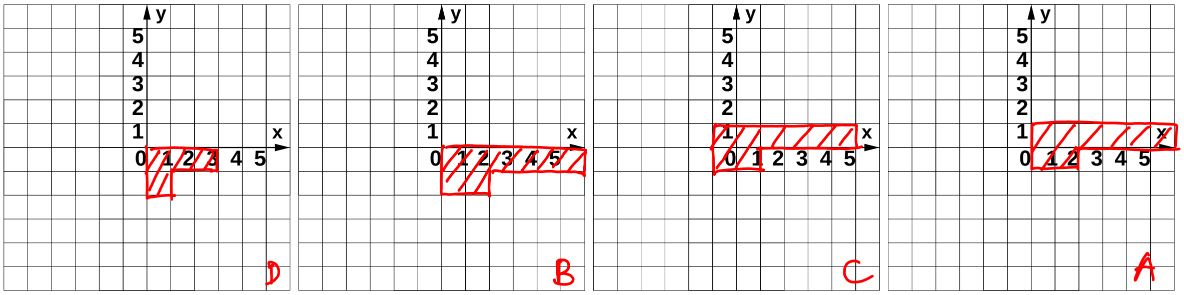
Left to Right:

translate(1,0,0)
translate(-1,1,0)
scale(2,1,1)
rotate(-90°,0,0,1)



Right to Left:

asymmetric scaling → rotation: shape changes when you rotate the local coord

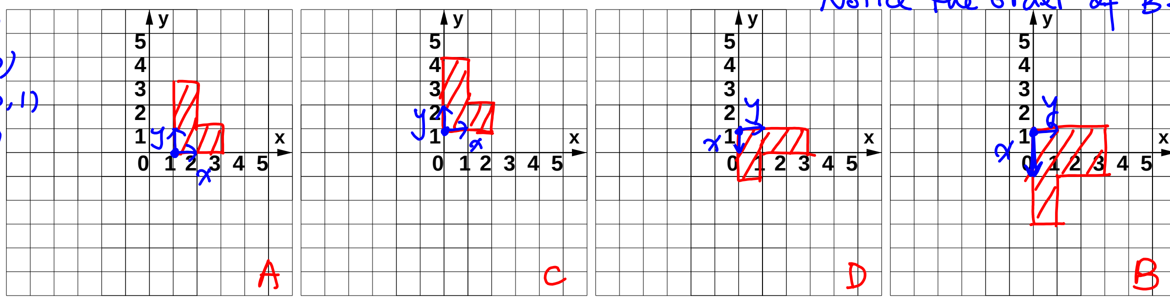


e) L’ = ACDB L

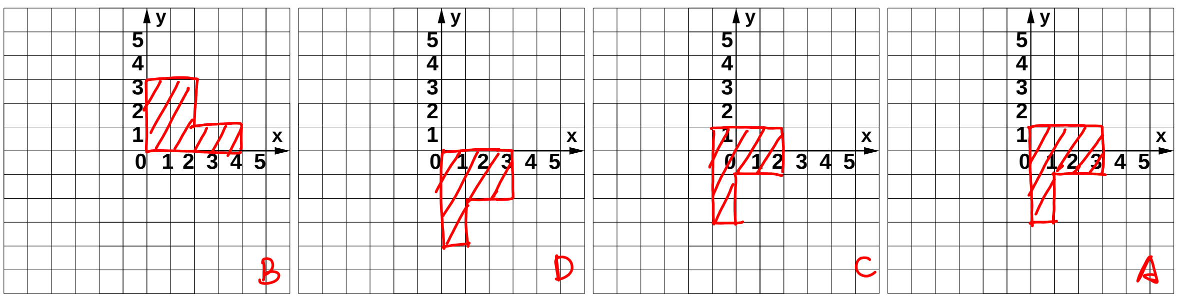
compare this with d) L' = ACBD L . Shape doesn't change when you rotate.
Notice the order of B & D.

Left to Right:

translate(1,0,0)
translate(-1,1,0)
rotate(-90°,0,0,1)
scale (2,1,1)



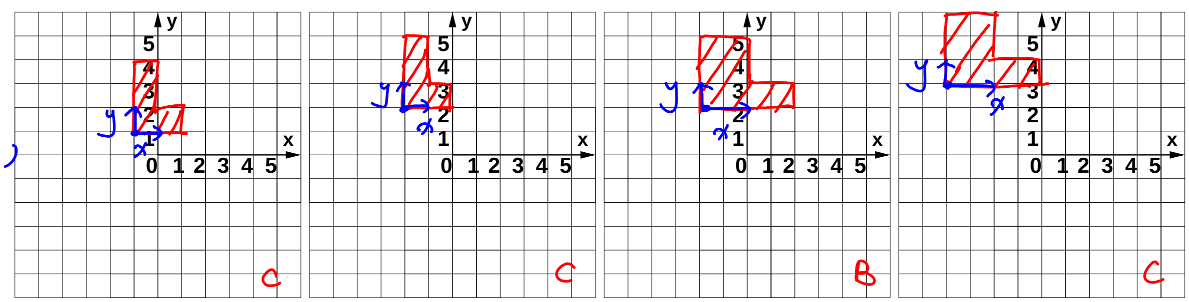
Right to Left:



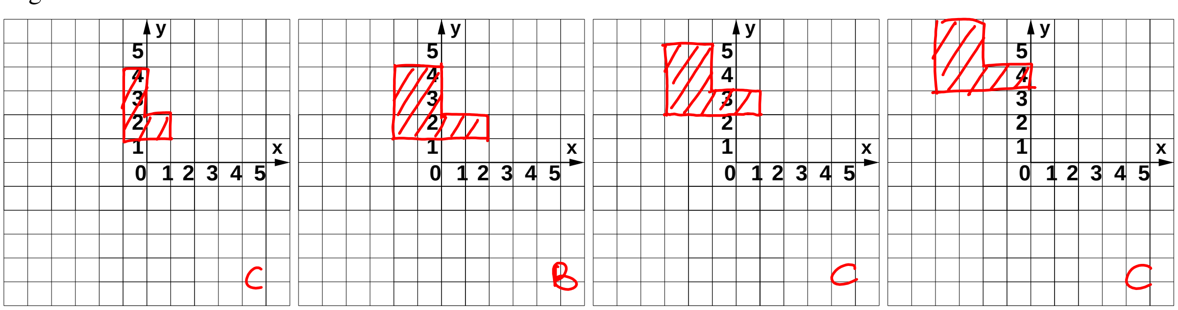
f) L’ = CCBC L

Left to Right:

translate(-1,1,0)
translate(-1,1,0)
scale(2,1,1)
translate(-1,1,0)



Right to Left:



[1 pt for each graph and 0.5pt for each command]