CPSC 314, Midterm Exam

8 March 2013

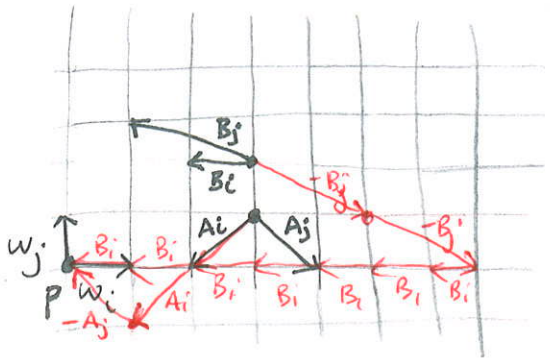
Closed book, no electronic devices besides simple calculators. Cell phones must be turned off. Place your photo ID face up on your desk. One single-sided sheet of handwritten notes is allowed, keep it so that you can reuse it for the final if you want.
Do not open the exam until told to do so. Answer the questions in the space provided. If you run out of room for an answer, continue on the back. There are 50 points, you have 50 minutes.
Good luck!

Name: Solutions

Student Number: _____

Question	Points Earned	Points Possible
1		4
2		6
3		16
4		~~17~~ 16
5		3
6		4
Total		~~50~~ 49

1. (4 pts) The point **p** can be specified as $\mathbf{p}_A = (0,0)^T$ in the coordinate frame W, with orthonormal basis vectors **i** and **j**. Specify the coordinates of point **p** in frames A and B.



A : (2, -1)

B : (7, -2)

1/2 sign error

1/2 flipped coords

1 pt per number

2. (6 pts) True/false

F (a) The projection matrix requires an eye point, lookat point, and up vector.

F (b) Normals can be safely transformed by the modelling matrix if it contains only translations and scales.

F (c) The homogeneous points (10,25,5,10) and (10,12,14,9) map to the same Cartesian point after homogenization.

F (d) Using display lists is incompatible with normalized device coordinates.

T (e) A translation transformation leaves the w coordinate of a homogeneous point unchanged.

F (f) The transformation from an orthographic view volume to the display coordinate system changes the relative ordering of objects in depth.

T (g) An object with normalized devices coordinates (.5, .5, .5) is visible in the final framebuffer image given a viewport of width 100 and height 100. (with no other object in the scene)

F (h) The transformation from an orthographic view volume to the display coordinate system may change the relative horizontal and vertical spacing between objects.

T (i) The transformation from an perspective view volume to the display coordinate system may change the relative horizontal and vertical positions of objects.

T (j) In the OpenGL rendering pipeline, graphics state persists until explicitly changed.

T (k) Any point inside the viewing frustum will fall within the range -1 to 1 in x, y, and z in normalized device coordinates.

T (l) One use case for an asymmetric viewing frustum is stereo viewing.

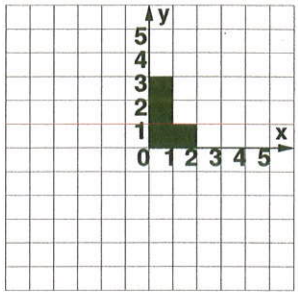
3. (16 pts) Draw shapes 2, 3, 4, and 5 transformed by the appropriate OpenGL commands in the left column below. The `drawShape()` code is shown in the middle column, and the result of the first call is shown in the right column.

```
 glIdentity();
drawShape(); // shape 1
glRotate(90, 0, 0, 1);
glTranslate(0, 2, 0);
drawShape(); // shape 2
glScale(2, 1, 1);
glTranslate(1, 0, 0);
drawShape(); // shape 3
glPushMatrix();
glTranslate(-2, -6, 0);
drawShape(); // shape 4
glPushMatrix();
glScale(2, 2, 1);
glRotate(90, 0, 0, 1);
glTranslate(0, 2, 0);
glPopMatrix();
```

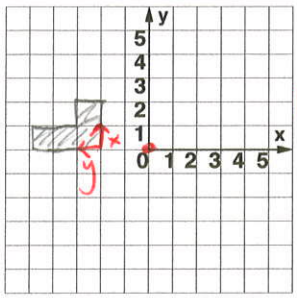
ignore

```
glRotate(90, 0, 0, 1);
glTranslate(2, 0, 0);
drawShape(); // shape 5


drawShape() {
  glBegin(GL_POLYGON);
  glVertex(0,0,0,1);
  glVertex(2,0,0,1);
  glVertex(2,1,0,1);
  glVertex(1,1,0,1);
  glVertex(1,3,0,1);
  glVertex(0,3,0,1);
  glEnd(GL_POLYGON);
}
```

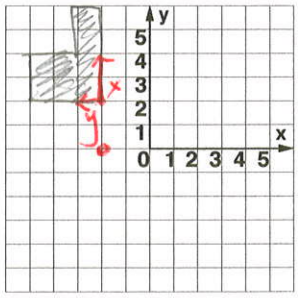


a) shape 2



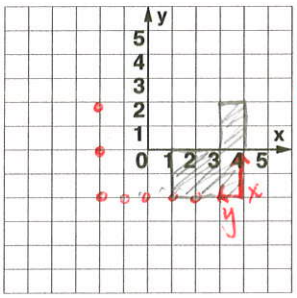
Origin : (-2, 0)

b) shape 3



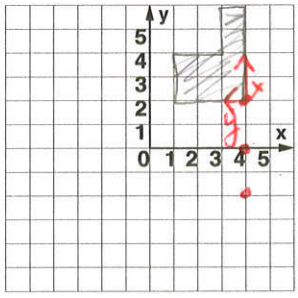
Origin : (-2, 2)

c) shape 4



Origin : (4, -2)

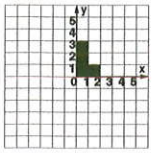
d) shape 5



Origin : (4, 2)

16

4. (17 pts) For each equation below, sketch the new location L' of the L shape on the grid, whose original position is shown below. For each equation, on the first row of grids show the incremental computations reading from from right to left (moving object). Then on the next line, show the computations considered left to right (moving coordinate frame). You should get different intermediate answers, but check your work to ensure that the final position of the L is be the same in both.



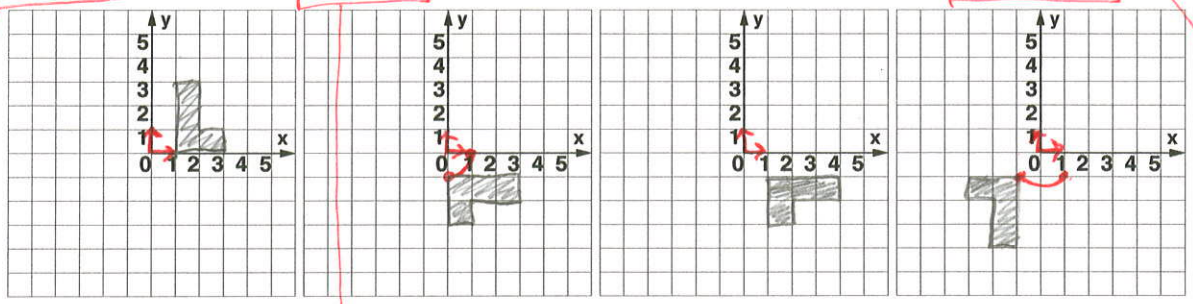
trans y rot 90° cw trans x scale x

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

a) L' = BCBC L.

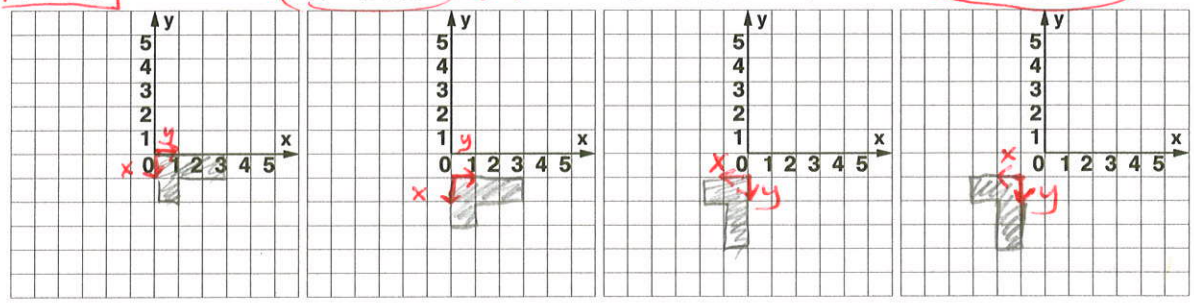
First row: right to left, moving object.

.5 L' = C L .5 L' = BC L 1 L' = CBC L .5 L' = BCBC L



Second row: left to right, moving coordinate frame.

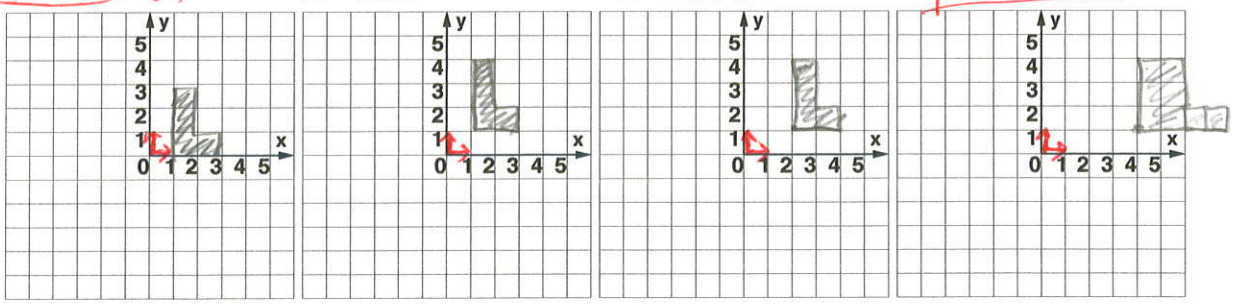
.5 L' = B L .5 L' = BC L 1 L' = BCB L .5 L' = BCBC L



b) L' = DCAC L

First row: right to left, moving object.

.5 L' = C L 1 L' = AC L 1 L' = CAC L .5 L' = DCAC L



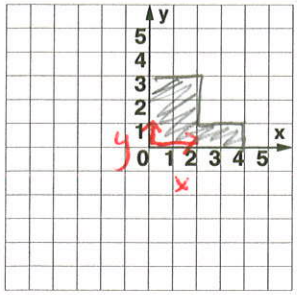
-1 for incorrect interpretation of matrix

duplicates .5 each, singletons 1 each

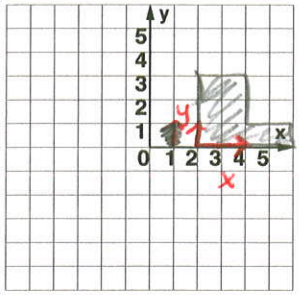
8 pts this page

Second row: left to right, moving coordinate frame.

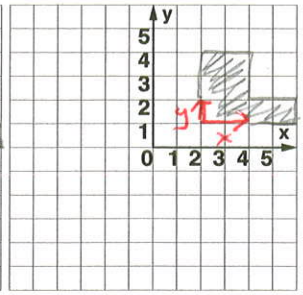
L' = D L



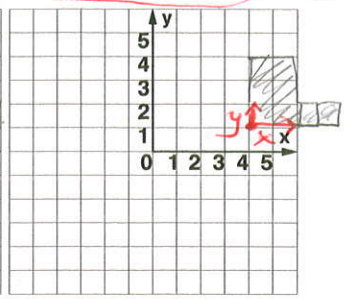
L' = DC L



L' = DCA L



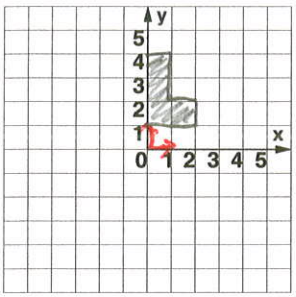
L' = DCAC L



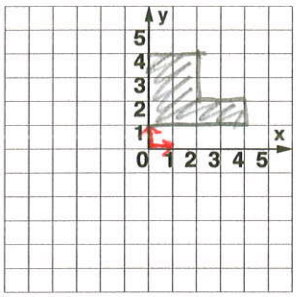
c) L' = BDA L

First row: right to left, moving object.

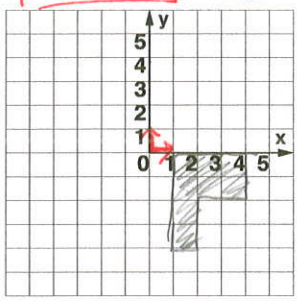
L' = A L



L' = DA L

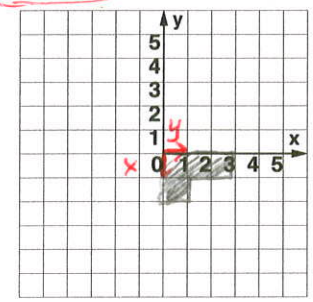


L' = BDA L

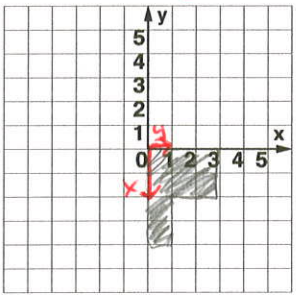


Second row: left to right, moving coordinate frame.

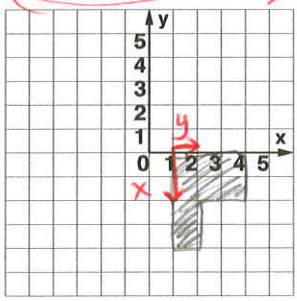
L' = B L



L' = BD L



L' = BDA L



8 pts this pag

5. (3 pts) Consider the following code.

```
1 glMatrixMode(GL_PERSPECTIVE);
2 glLoadIdentity();
3 glFrustum(-10, 10, -10, 10, 10, 100);
4 glMatrixMode(GL_MODELVIEW);
5 glLoadIdentity();
6 gluLookAt(100, 0, 0, 0, 0, -20, 0, 1, 0);
7 drawRabbit();
```

5, 2.5, 5

For reference, the OpenGL command syntax is

- `glFrustum(left, right, bottom, top, near, far)`
- `gluLookAt(eyex, eyey, eyez, centerx, centery, centerz, upx, upy, upz)`

The `drawRabbit()` function draws an rabbit in the current coordinate system with extent ranging from 0 to 10 in x, 0 to 5 in y, and 0 to 10 in z. When you run your program, you are sad that you cannot see the rabbit at all.

Fix this problem so that the whole rabbit is visible and centered within the image plane by changing only the lookat point. State which line you are changing, and give new parameters for the OpenGL command.

To center animal, calculate middle of range: (10-0)/2 = 5 in x, (5-0)/2 = 2.5 in y, (10-0)/2 = 5 in z. Many other answers fully correct, along ray between eye and center. For animal visible but not centered, +1. For centered, +2 more.

6. (4 pts) A point p is at location (0,0) in DCS (display coordinates), with a viewport of width 1000 and height 1000. Give its x and y location in NDCS (normalized device coordinates).

(-1, 1)

(-.999, .999) also fine

sign error: -1

short way: realize that upper left corner is just (-1,1)

if derived in full: 2/4 for right big idea but problem with details