CPSC 314, Midterm Exam

8 March 2010

Closed book, no electronic devices besides (simple, nongraphing) 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 100 points, you have 50 minutes.

Good luck!

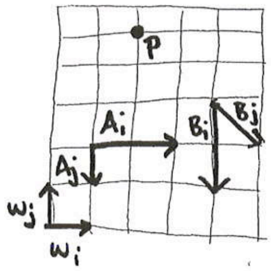
Name: Solutions

Student Number: ____________________

Question	Points Earned	Points Possible
1		6
2		10
3		12
4		16
5		8
6		4
7		20
8		24
Total		100

1. (6 pts) The point **p** can be specified as $\mathbf{p}_A = (2,5)^T$ in the coordinate frame W, with orthonormal basis vectors **i** and **j**. Specify the coordinates of point **p** in frames A and B.

3pts each
-1 sign error each



$$P_A = (0.5, -3)^T$$

$$P_B = (0, -2)^T$$

2. (10 pts) Find the 3x3 homogeneous transformation which transforms a point from Frame B into the Frame W coordinate system. That is, give M where $\mathbf{p}_W = M\mathbf{p}_B$. Verify your solution using one of your answers to question ~~2~~. 1

construct M from basis vectors as first two columns, origin as third

$$M = \begin{bmatrix} 0 & 1 & 4 \\ -2 & -1 & 3 \\ 0 & 0 & 1 \end{bmatrix} \underset{\uparrow P_B}{\begin{bmatrix} 0 \\ -2 \\ 1 \end{bmatrix}} \begin{bmatrix} 2 \\ 5 \\ 1 \end{bmatrix} = p_W = M p_B ? = \begin{bmatrix} -2+4 \\ 4+1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 5 \\ 1 \end{bmatrix} \checkmark$$

9 pts M:
3 pts each column
-6 if flip row/col.

1 pts verify

-1 if w=0 not 1
-1 each sign error

3. (12 pts) True/false

- F • The only way to specify a viewing transformation is by using eye point, lookat point, and up vector.
- F • If a surface is transformed by a nonuniform scale, transforming its normal vector by the same transformation will leave it perpendicular to the surface.
- F • The homogeneous points (1,2,3,4) and (1,2,3,8) map to the same Cartesian point after homogenization.
- T • Using display lists will affect performance differently depending on the graphics card in the machines.
- T • A rotation transformation leaves the w coordinate of a homogeneous point unchanged.
- T • The transformation from an orthographic view volume to the display coordinate system depends on the size of the viewport.
- F • An object with normalized devices coordinates (5, 5, 5) is visible in the final framebuffer image given a viewport of width 500 and height 500.
- F • In OpenGL, vertices are always specified in the world coordinate system.
- T • In OpenGL, light positions can be specified in either the world or the viewing coordinate system.
- F • The transformation from the RGB to the HSV colorspace can be expressed by a 4x4 matrix.
- T • The CIE colorspace incorporates measurements of human perceptual response and encodes all luminance information in one of its three channels.
- F • Intensity in the HSI colorspace correctly encodes luminance information taking into account human perceptual response.

1 pt each

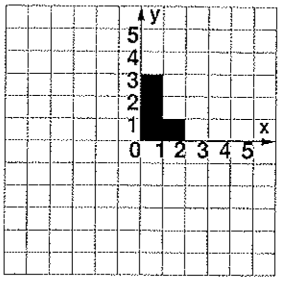
4. (16 pts) Draw shapes 2, 3, 4, and 5 transformed by the appropriate OpenGL commands in the left column below. The `drawShape()` code is shown in the middle column, and the result of the first call is shown in the right column.

```
glIdentity();
drawShape(); // shape 1
glRotate(90, 0, 0, 1);
glTranslate(1, 0, 0);
drawShape(); // shape 2
glRotate(-90, 0, 0, 1);
glPushMatrix();
glTranslate(1, 0, 0);
drawShape(); // shape 3
glScale(1, 2, 1);
glTranslate(1, -2, 0);
drawShape(); // shape 4
glTranslate(-2, 2, 0);
glRotate(90, 0, 0, 1);
glTranslate(-1, -1, 0);
glScale(1, .5, 1);
glRotate(90, 0, 0, 1);
```

no effect!

```
glPopMatrix();
glTranslate(-2, 1, 0);
drawShape(); // shape 5
```

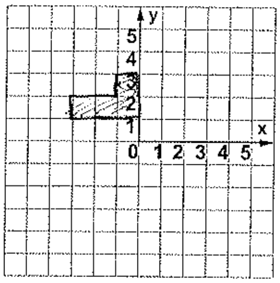
```
drawShape() {
  glBegin(GL_POLYGON);
  glVertex(0,0,0,1);
  glVertex(2,0,0,1);
  glVertex(2,1,0,1);
  glVertex(1,1,0,1);
  glVertex(1,3,0,1);
  glVertex(0,3,0,1);
  glEnd(GL_POLYGON);
}
```



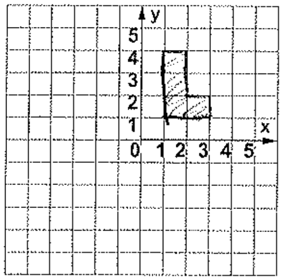
4 pts each

(we checked for partial credit if cascading errors)

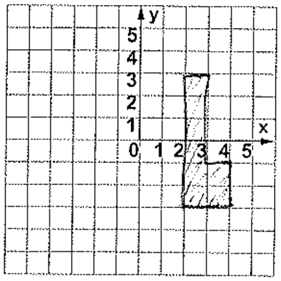
a) shape 2



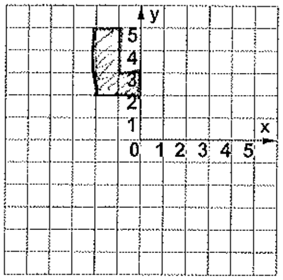
b) shape 3



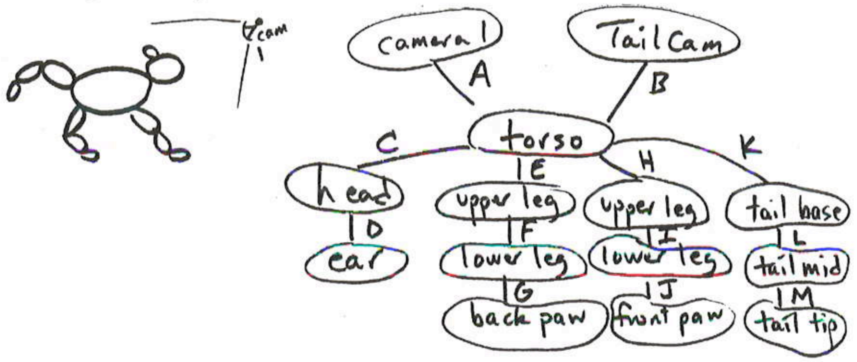
c) shape 4



d) shape 5



5. (8 pts) Scene Graphs. The transformation matrices in the following scene graph define the relative transformations of each body part with respect to its parent. foo.



- (4 pts) Your cat wants an earring, because all the other kitties have one. You should draw it as an offset with respect to the coordinate system of the ear. Give the expression for the composite transformation that should be in the modelview matrix to get from viewing to ear coordinates: that is, the transformation to get from the viewing coordinate system to the ear coordinate system, or equivalently the matrix that takes a point specified in ear coordinates and transforms it to viewing coordinates.

$A\ C\ D$

-1 missing view xform

-2 incorrect inverse

-3 backwards

- (4 pts) You want to add a TailCam, a second camera that lets you see what things would look like from a point of view of the end of your cat's tail. (You will get dizzy when your cat sees a mouse and lashes its tail!) Give the expression for the composite transformation that you should use for B, the viewing matrix for the TailCam. (B is the V2W matrix from the coordinate frame point of view, and the W2V matrix from the point/object point of view.)

$$B = (KLM)^{-1} = M^{-1}L^{-1}K^{-1}$$

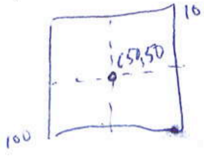
-2 missing inverse

-1 expression includes B

6. (4 pts) A point p is at location (50,50) in DCS (display coordinates), with a viewport of width 100 and height 100. Give its x and y location in NDCS (normalized device coordinates).

the short way: realize it's in center of screen + that is NDC origin!

(0,0)



the long way: multiply it out

$$\begin{bmatrix} 50 \\ 50 \\ z_D \\ 1 \end{bmatrix} = \begin{bmatrix} 50 & 0 & 0 & 50 \\ 0 & -50 & 0 & 50 \\ 0 & 0 & .5 & .5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_N \\ y_N \\ z_N \\ 1 \end{bmatrix}$$

accepted with full credit: x, y in range (-1,1) (depending on whether you do the half-pixel shift, etc...)

+1 pt correct line a-c
-3pts wrong value a-c
+1 pt correct param, wrong value

7. (20 pts) Viewing.

```
1 glMatrixMode(GL_PERSPECTIVE);
2 glLoadIdentity();
3 glFrustum(-5, 5, -5, 5, 15, 150);
4 glMatrixMode(GL_MODELVIEW);
5 glLoadIdentity();
6 gluLookAt(0, 0, 10, 0, 0, 0, 0, 1, 0);
7 drawIguana();
```

The drawIguana() function draws an iguana in the current coordinate system with extent ranging from -5 to 5 in x, -2 to 2 in y, and 0 to 1 in z. When you run your program, you are sad that you cannot see the iguana.

a • (4 pts) Fix this problem so that the iguana is visible and nearly fills the image plane, by changing only the eye point. State which line you are changing, and give new parameters for the OpenGL command.

6 gluLookAt (0, 0, 16, 0,0,0, 0,1,0)   also OK: 15.1 up to 20 or so

b • (4 pts) Fix this problem so that the iguana is visible and nearly fills the image plane, by changing only the near clipping plane. State which line you are changing, and give new parameters for the OpenGL command.

3 glFrustum (-5,5,-5,5, 9, 150)   also OK: up to 9.99 down to 0.01

c • (4 pts) Change the view by rotating the image of the iguana 90 degrees clockwise on the image plane, by changing only the up vector. State which line you are changing, and give new parameters for the OpenGL command.

6 gluLookAt (0,0,16, 0,0,0, -1,0,0)

d • (4 pts) Give the viewing matrix produced by the original code above. That is, the V2W matrix considered from the coordinate frame point of view, or the W2V matrix considered from the transforming object point of view.

the shortway: realize that only non identity element is translation in z

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -10 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

-2 wrong value for z translate, but otherwise right
-1 arithmetic error

the long way: $w = \frac{-g}{\|g\|} = \begin{bmatrix} 0 \\ 0 \\ 10/10 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ $t = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$

$u = \frac{t \times w}{\|t \times w\|} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ $g = \begin{bmatrix} 0 \\ 0 \\ -10 \end{bmatrix}$ $v = w \times u = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -10 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & \cancel{6} \\ 0 & 0 & 1 & -10 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

e • (4 pts) Give the perspective matrix produced by the original code above. That is, the N2V matrix considered from the coordinate frame point of view, or the V2N matrix considered from the transforming object point of view.

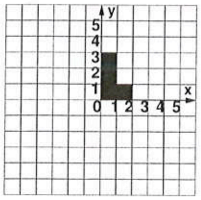
$$\begin{bmatrix} \frac{2n}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2n}{r-l} & 0 & 0 \\ 0 & 0 & \frac{-f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} \cancel{\frac{2\cdot 15}{10}} \to 3 & & & \\ & 3 & & \\ & & -\frac{165}{135} & -\frac{2*15*150}{135} \\ & & -1 & \end{bmatrix} = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & -11/9 & -100/3 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

-1.22   -33.33

n=15, f=150
l=-5, r=5
b=-5, t=5

+1 some attempt to handle perspective matrix
-1 arithmetic errors

8. (24 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)`.



with typo:
glTranslate(0,1,0);
glRotate(-90,0,0,1);

`drawL();`

full credit given whether or not you used intended matrix (just rot) or typo matrix (~~[illegible]~~ trans + rot) for B!

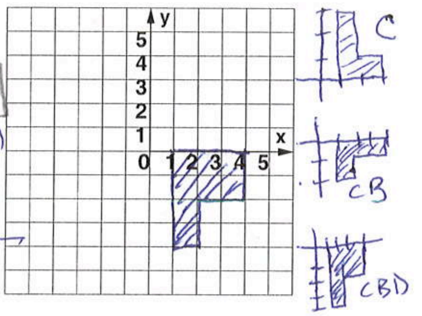
glTranslate(0,1,0) glRotate(-90,0,0,1) glTranslate(1,0,0) glScale(2,1,1)

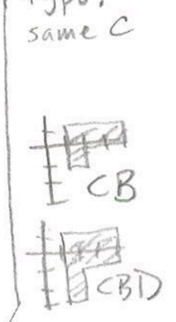
$$\mathbf{A}=\begin{bmatrix}1&0&0&0\\0&1&0&1\\0&0&1&0\\0&0&0&1\end{bmatrix},\mathbf{B}=\begin{bmatrix}0&1&0&0\\-1&0&0&\cancel{1}0\\0&0&1&0\\0&0&0&1\end{bmatrix},\mathbf{C}=\begin{bmatrix}1&0&0&1\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix},\mathbf{D}=\begin{bmatrix}2&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

trans 1 y rot 90° ccw trans 1 y scale 2 x

(with typo) a)
glTrans(1,0,0)
[glTrans(0,1,0)]
glRotate(-90,0,0,1)
glScale(2,1,1)

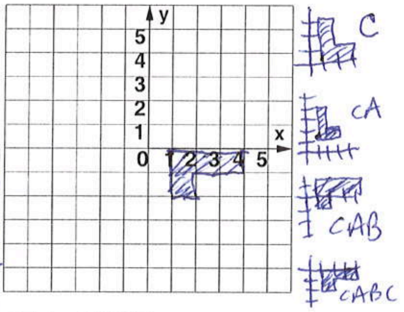
a) L' = CBD L

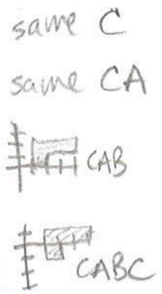




(with typo) b)
glTrans(1,0,0)
glTrans(0,1,0)
~~glRot~~
[glTrans(0,1,0)]
glRot(-90,0,0,1)
glTrans(1,0,0)

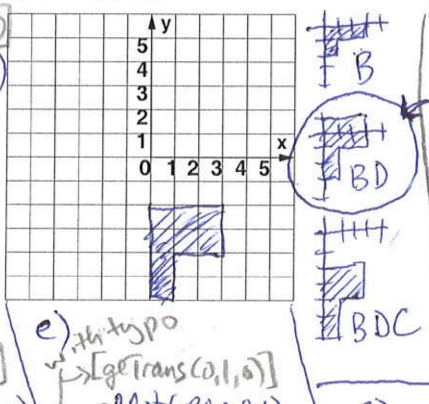
b) L' = CABC L

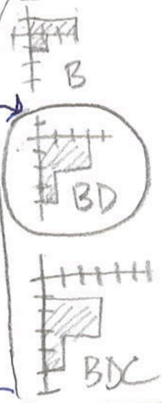




(with typo) c)
[glTrans(0,1,0)]
glRot(-90,0,0,1)
glScale(2,1,1)
glTrans(1,0,0)

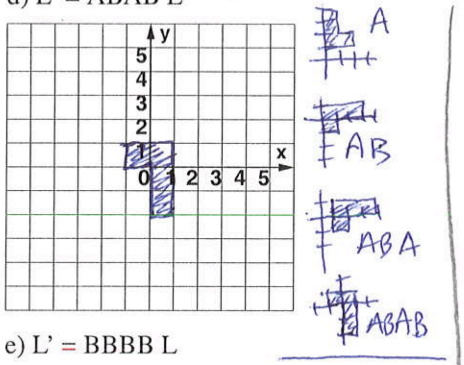
c) L' = BDC L



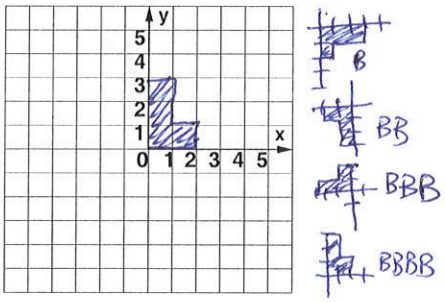


(with typo) d)
glTrans(0,1,0)
[glTrans(0,1,0)]
glRot(-90,0,0,1)
glTrans(0,1,0)
[glTrans(0,1,0)]
glRot(-90,0,0,1)

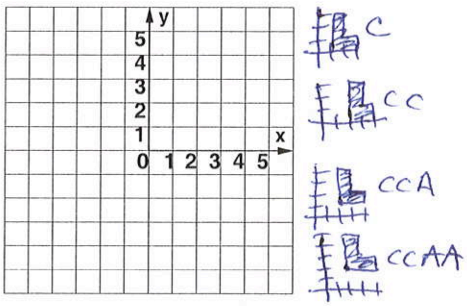
d) L' = ABAB L



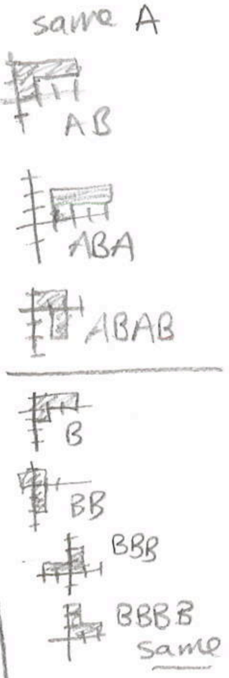
e) L' = BBBB L



f) L' = CCAA L



typo:



same

e) with typo
[glTrans(0,1,0)]
glRot(-90,0,0,1)
[glTrans(0,1,0)]
glRot(-90,0,0,1)
[glTrans(0,1,0)]
glRot(-90,0,0,1)
[glTrans(0,1,0)]
glRot(-90,0,0,1)

f) glTrans(1,0,0);
glTrans(1,0,0);
glTrans(0,1,0)
glTrans(0,1,0)

no penalty: - missing semicolons, drawL call, etc
- interpret B w/ typo backwards as RT vs. TR (checked for cascading errors)

scheme: +1 each for OpenGL
+3 each for L position
-1 once if glScale given with 0 where 1 needed