CSE

Computer Graphics
Spring 2015 Midterm

UTA

NAME:

Prob #	1	2	3	4	5	6
Points	11	12	10	22	21	24

Time: 80 Minutes

NOTES:

a. **Credit is only given to the correct numerical values.**
b. **All numerical values must be calculated with three digits of accuracy after the decimal point.**
c. **Do not write on the back side of the papers.**

1. Given two pints A(6,8,3) and B(16,2,11), find the sequence of transformations to bring the point A to the origin and make point B to be on the z axis.

Matrix #1: Translate (3 points)

1	**0**	**0**	**-6**
0	**1**	**0**	**-8**
0	**0**	**1**	**-3**
0	**0**	**0**	**1**

Matrix #2: Rx (4 points)

1	**0**	**0**	**0**
0	**0.8**	**0.6**	**0**
0	**-0.6**	**0.8**	**0**
0	**0**	**0**	**1**

Matrix #3: Ry (4 points)

0.707	**0**	**-0.707**	**0**
0	**1**	**0**	**0**
0.707	**0**	**0.707**	**0**
0	**0**	**0**	**1**

Matrix #4:

2. Consider a parametric quadratic curve in 3-dimensional space.

This curve @t=0 is passing through point $p_0 = (2,4,6)$

The derivative of the curve @ t=0 is $\frac{dp_0}{dt}$=(1,3,7)

and the derivative of this curve @t=1 is $\frac{dp_0}{dt}$=(3, 4, 1)

Find the coordinates of this curve @ t=0.5

The coordinates of the curve @ t=0.5 is $p_{0.5} =$ 2.75,5.625,8.75

$$\begin{cases} x(t) = a_1t^2 + b_1t + c_1 \\ y(t) = a_2t^2 + b_2t + c_2 \\ z(t) = a_3t^2 + b_3t + c_3 \end{cases}$$

@t=0, $\begin{cases} x(0) = 2 \\ y(0) = 4 \\ z(0) = 6 \end{cases}$ => $\begin{cases} c_1 = 2 \\ c_2 = 4 \\ c_3 = 6 \end{cases}$ (2 points)

@t=0, $\begin{cases} x'(0) = 1 \\ y'(0) = 3 \\ z'(0) = 7 \end{cases}$ => $\begin{cases} b_1 = 1 \\ b_2 = 3 \\ b_3 = 7 \end{cases}$ (2 points)

@t=1, $\begin{cases} x'(1) = 2 * a_1 + b_1 = 2 * a_1 + 1 = 3 \\ y'(1) = 2 * a_2 + b_2 = 2 * a_2 + 3 = 4 \\ z'(1) = 2 * a_3 + b_3 = 2 * a_3 + 7 = 1 \end{cases}$ => $\begin{cases} a_1 = 1 \\ a_2 = 0.5 \\ a_3 = -3 \end{cases}$ (4 points)

⇨ @t=0.5, $\begin{cases} x(0.5) = 2.75 \\ y(0.5) = 5.625 \\ z(0.5) = 8.75 \end{cases}$ (3 points)

3. Point A(-10, 5) is given in a two dimensional world coordinate system. Find the coordinates of the point A on the screen after it is mapped from window to viewport.

$x_{wmin} = -15$ $y_{wmin} = 1$ $x_{wmax} = 6$ $y_{wmax} = 9$

Normalized device coordinate of the viewport:

$x_{vmin} = 0.1$ $y_{vmin} = 0.25$ $x_{vmax} = 0.6$ $y_{vmax} = 0.8$

The origin of the screen coordinate system is defined in the **upper left** corner of the screen and the screen resolution is 1920 by 1080.
Use rounding to convert from float to integer.

$$\boldsymbol{S_x = \frac{0.6 - 0.1}{6 - (-15)} = 0.0238}$$

⇨ $\boldsymbol{A_x = [0.1 + 0.0238(-10 - (-15))] \times 1920 = 420.48 \sim 420}$

$$\boldsymbol{S_y = \frac{0.8 - 0.25}{9 - 1} = 0.0687}$$

⇨ $\boldsymbol{A_y = [0.25 + 0.0687(9 - 5)] \times 1080 = 566.784 \sim 566}$

(6 points)

Screen coordinates of point A after mapping are: (420, 566) (4 points)

4. The viewing parameters for a perspective projection are given as
VRP(WC)=(**2,3,5**) VPN(WC)=(**0, 0,4**)
VUP(WC)=(**0,2,0**) PRP (VRC)=(**10,8,5**)

u_{min} (VRC) = **3** u_{max} (VRC) = **5**
v_{min} (VRC) = **38** v_{max} (VRC) = **42**
n_{min} (VRC) = **8** n_{max} (VRC) = **9**

Find the sequence of transformations which will transform this viewing volume into a standard perspective view volume which is bounded by the planes: x=z ; x=-z ; y=z ; y=-z ; z=1 ; z=zmin

a. Find the **Shear matrix** (Matrix #6)
b. Find the **scale matrices** (Matrix #7 and Matrix #8).
c. Find the **zmin** after all transformations are done.

Matrix #2: Rx

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Matrix #1: Translate

1	0	0	-2
0	1	0	-3
0	0	1	-5
0	0	0	1

Matrix #4: Rz

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Matrix #3: Ry

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Matrix #6: Shear (8 points)

1	**0**	**-1.2**	**0**
0	**1**	**6.4**	**0**
0	**0**	**1**	**0**
0	**0**	**0**	**1**

Matrix #5: Translate

1	0	0	-10
0	1	0	-8
0	0	1	-5
0	0	0	1

Scale2 (6 points)

0.25	**0**	**0**	**0**
0	**0.25**	**0**	**0**
0	**0**	**0.25**	**0**
0	**0**	**0**	**1**

Scale1 (4 points)

5	**0**	**0**	**0**
0	**2.5**	**0**	**0**
0	**0**	**1**	**0**
0	**0**	**0**	**1**

Zmin= 0.750 (4 points)

5. Clip line AB **A(10.2 , 4.6 , -7.4)** , **B(-9.8 , -3.4 , 8.6)** against the three planes **x=z** ; **y=z** ; and ; **z=1** in the standard perspective viewing volume with **zmin=0.1**

Note: You do not need to clip against the other three planes in the standard volume. You must specify your reason for accept or rejecting an intersection point.

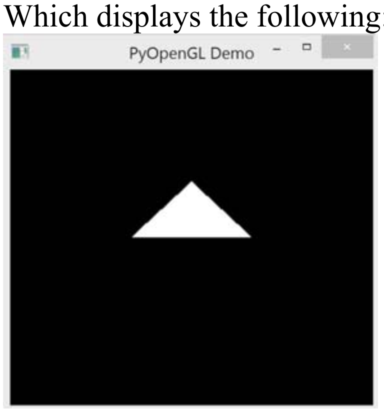
Equation of line AB
$$\begin{cases} x(t) = -20t + 10.2 \\ y(t) = -8t + 4.6 \\ z(t) = 16t - 7.4 \end{cases}$$ **(6 points)**

Plane	t	Intersection point (x,y,z)	Accept or Reject	Reason to accept or reject
x = z	**0.489**	**(0.422, 0.689, 0.422)**	**R**	**|y|>z**
y = z	**0.5**	**(0.2, 0.6, 0.6)**	**A**	**|x|<z**
z=1	**0.525**	**(-0.3, 0.4, 1)**	**A**	**|x|<z** **|y|<z**
(9 points)			**(6 points)**	

6. Consider the following OpenGL program:

```
from OpenGL.GL import *
from OpenGL.GLU import *
from OpenGL.GLUT import *
def display():
        glClear(GL_COLOR_BUFFER_BIT)
        glBegin(GL_TRIANGLES)
        glColor3f(1,1,1)
        glVertex3f(-1,0,0)
        glVertex3f(1,0,0)
        glVertex3f(0,1,0)
        glEnd()
        glFlush()
        glutSwapBuffers()
glutInit(sys.argv)
glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB)
glutCreateWindow(b"PyOpenGL Demo")
glutDisplayFunc(display)
glMatrixMode(GL_PROJECTION)
glLoadIdentity()
glFrustum(-1,1,-1,1,1,30)
gluLookAt(0,0,3,0,0,0,0,1,0)
glMatrixMode(GL_MODELVIEW)
glLoadIdentity()
glutMainLoop()
```

Which displays the following:



a. If the code in line 21 is replaced with: (4 points)

```
gluLookAt(0.0, 0.0, 2.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0)
```

What happens to the image on the screen? (all other lines stay the same).

- ■ The size of the image of the object on the screen get larger
- ☐ The size of the image of the object on the screen get smaller
- ☐ The image of the object on the screen moves to the right
- ☐ The image of the object on the screen moves to the left
- ☐ The image of the object on the screen rotate clockwise
- ☐ The image of the object on the screen rotate counter-clockwise
- ☐ Nothing changes

b. If the code in line 21 is replaced with: (4 points)

```
gluLookAt(0.0, 0.0, 3.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0)
```

What happens to the image on the screen? (all other lines stay the same).

- ☐ The size of the image of the object on the screen get larger
- ☐ The size of the image of the object on the screen get smaller
- ☐ The image of the object on the screen moves to the right
- ■ The image of the object on the screen moves to the left
- ☐ The image of the object on the screen rotate clockwise
- ☐ The image of the object on the screen rotate counter-clockwise
- ☐ Nothing changes

c. If the code in line 21 is replaced with: (4 points)

```
gluLookAt(0.0, 0.0, 3.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0)
```

What happens to the image on the screen? (all other lines stay the same).

- ☐ The size of the image of the object on the screen get larger
- ☐ The size of the image of the object on the screen get smaller
- ☐ The image of the object on the screen moves to the right
- ☐ The image of the object on the screen moves to the left
- ☐ The image of the object on the screen rotate clockwise
- ■ The image of the object on the screen rotate counter-clockwise
- ☐ Nothing changes

d. If the code in line 20 is replaced with: (4 points)

```
glFrustum(-2,0,-1,1,1,30)
```

What happens to the image on the screen? (all other lines stay the same).

- ☐ The size of the image of the object on the screen get larger
- ☐ The size of the image of the object on the screen get smaller
- ■ The image of the object on the screen moves to the right
- ☐ The image of the object on the screen moves to the left
- ☐ The image of the object on the screen rotate clockwise
- ☐ The image of the object on the screen rotate counter-clockwise
- ☐ Nothing changes

e. If the code in line 20 is replaced with: (4 points)

```
glFrustum(-2,2,-2,2,1,30)
```

What happens to the image on the screen? (all other lines stay the same).

- ☐ The size of the image of the object on the screen get larger
- ■ The size of the image of the object on the screen get smaller
- ☐ The image of the object on the screen moves to the right
- ☐ The image of the object on the screen moves to the left
- ☐ The image of the object on the screen rotate clockwise
- ☐ The image of the object on the screen rotate counter-clockwise
- ☐ Nothing changes

f. If the code in line 20 is replaced with: (4 points)

```
glFrustum(-1,1,-1,1,1,10)
```

What happens to the image on the screen? (all other lines stay the same).

- ☐ The size of the image of the object on the screen get larger
- ☐ The size of the image of the object on the screen get smaller
- ☐ The image of the object on the screen moves to the right
- ☐ The image of the object on the screen moves to the left
- ☐ The image of the object on the screen rotate clockwise
- ☐ The image of the object on the screen rotate counter-clockwise
- ■ Nothing changes

$$R_z(\theta)=\begin{bmatrix}\cos\theta & -\sin\theta & 0 & 0\\ \sin\theta & \cos\theta & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\quad R_y(\theta)=\begin{bmatrix}\cos\theta & 0 & \sin\theta & 0\\ 0 & 1 & 0 & 0\\ -\sin\theta & 0 & \cos\theta & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\quad R_x(\theta)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos\theta & -\sin\theta & 0\\ 0 & \sin\theta & \cos\theta & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$M_{Hermite}=\begin{bmatrix}2 & -3 & 0 & 1\\ -2 & 3 & 0 & 0\\ 1 & -2 & 1 & 0\\ 1 & -1 & 0 & 0\end{bmatrix}\qquad M_{Bezier}=\begin{bmatrix}-1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 3 & 0 & 0\\ 1 & 0 & 0 & 0\end{bmatrix}$$

Rotate a vector around x axis until it lies in the xz plane

$$V=\begin{bmatrix}a\\ b\\ c\\ 1\end{bmatrix}\qquad R_x=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \frac{c}{\sqrt{b^2+c^2}} & \frac{-b}{\sqrt{b^2+c^2}} & 0\\ 0 & \frac{b}{\sqrt{b^2+c^2}} & \frac{c}{\sqrt{b^2+c^2}} & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Rotate a vector around y axis until it lies in the yz plane

$$V=\begin{bmatrix}a\\ b\\ c\\ 1\end{bmatrix}\qquad R_y=\begin{bmatrix}\frac{c}{\sqrt{a^2+c^2}} & 0 & \frac{-a}{\sqrt{a^2+c^2}} & 0\\ 0 & 1 & 0 & 0\\ \frac{a}{\sqrt{a^2+c^2}} & 0 & \frac{c}{\sqrt{a^2+c^2}} & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Rotate a vector around z axis until it lies in the yz plane

$$V=\begin{bmatrix}a\\ b\\ c\\ 1\end{bmatrix}\qquad R_z=\begin{bmatrix}\frac{b}{\sqrt{a^2+b^2}} & \frac{-a}{\sqrt{a^2+b^2}} & 0 & 0\\ \frac{a}{\sqrt{a^2+b^2}} & \frac{b}{\sqrt{a^2+b^2}} & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

How to convert a general perspective view volume into canonical perspective volume

Step 1: Translate VRP to origin
Step 2: Rotate VPN around x until it lies in the xz plane with positive z
Step 3: Rotate VPN around y until it aligns with the positive z axis.
Step 4: Rotate VUP around z until it lies in the yz plane with positive y
Step 5: Translate PRP (COP) to the origin
Step 6: Shear such that the center line of the view volume becomes the z axis
Step 7: Scale such that the sides of the view volume become 45 degrees
Step 8: Scale such that the view volume becomes the canonical perspective volume