Name / Username (unix.ic)

CMPS 160 F10
Introduction to Computer Graphics
Oct 28, 2010
Midterm Exam

You have the entire class period to complete this exam.

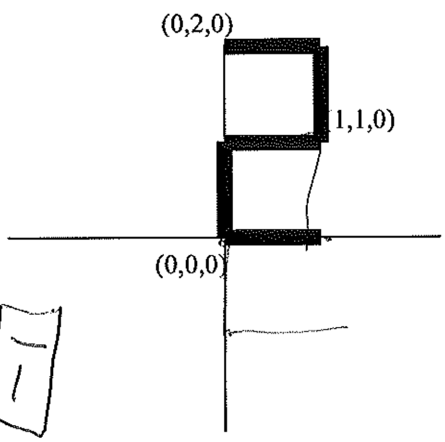
All *pages* are worth an equal amount.

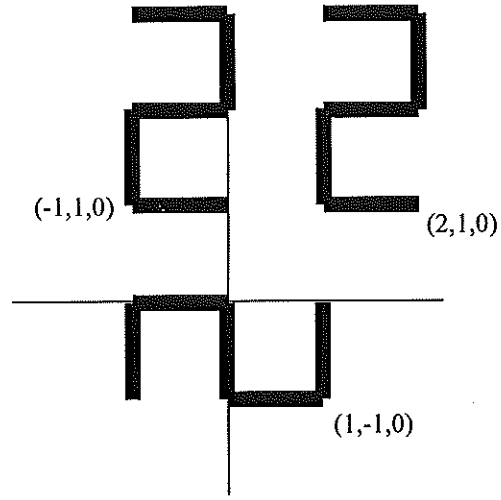
Partial credit will be given for clear evidence of correct reasoning even if the final solution is incomplete.

No books
One page hand written notes
Simple calculators allowed
No computers

~~3~~ 1

6. Consider the following sample displays (x+ to the right, y+ to the top and z+ out of the page). The left display is the result of calling drawShape without any special transformations. The right display is the result of calling drawCollection (which makes use of drawShape and some transformations). The shape is a series of line segments, all one unit long.





In your code below, make use of any standard OpenGL functions you need. If you are unsure about how a specific function works make a note of what you think it does.

Write out a full definition for the C function drawShape.

```
void drawShape () {
    glBegin (GL_LINE);     ← GL_LINES
    glVertex (1,0,0);
    glVertex (0,0,0);

    glVertex(0,0,0);
    glVertex(0,1,0);

    glVertex(0,1,0);
    glVertex(1,1,0);

    glVertex(1,1,0);
    glVertex(1,2,0);

    glVertex(1,2,0);
    glVertex(0,2,0);

    glEnd();
}
```

3

Write out a full definition for the C function drawCollection using drawShape.

```
void drawCollection() {
    glPushMatrix();
    glTranslate (1,1,0);
    drawShape();
    glPopMatrix();

    glPushMatrix();
    glTranslate (-1,1,0);
    drawShape();
    glPopMatrix();

    glPushMatrix();
    glTranslate (1,-1,0);
    glRotate (90, [0,0,1]);
    drawShape();
    glPopMatrix();
}
```

2 2 3 2

Also,
glRotate(90,0,0,1)
glTranslate(-1,-1,0)

(-90,0,0)
(0,-1,0)

Axis wrong

Numbers wrong, lose all points.

Missing ~~gl~~ glBegin(), glEnd() -1

Missing glPushMatrix/PopMatrix ~~and it~~ and therefore messing up transforming

Forgets glVertex, uses some other (but everything else ok) -1

LINE vs LINE-STRIP OR

~~translate & Rotate wrong order but~~ numbers ok, Correct ✓.

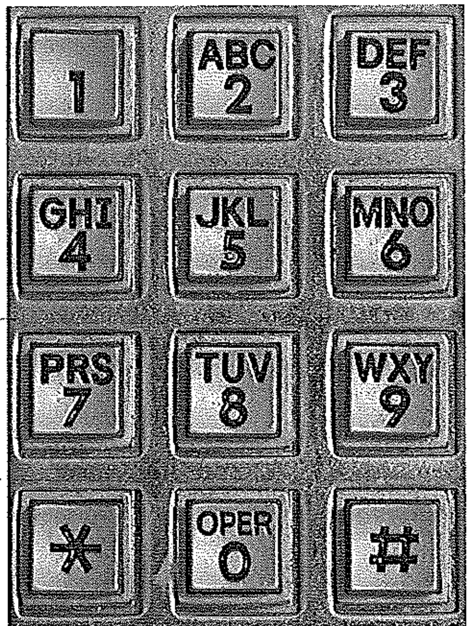
If you do not use push/pop but get the tx ~~#~~ full points 2

5. This is a texture map with U,V coordinates listed.

(0,1) **(1,1)**

.5

.25



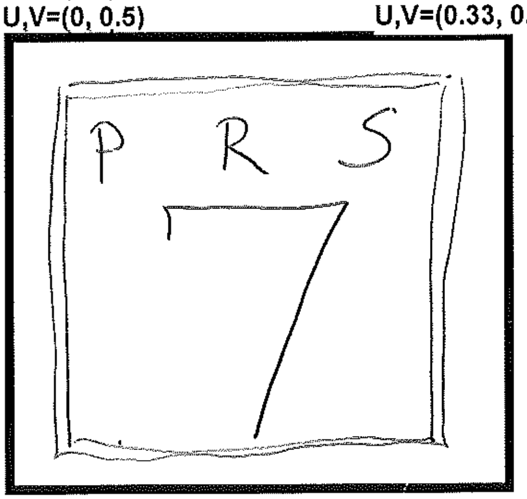
.33 .66

(0,0) **(1,0)**

+1 right patch
+1 sub patches in right place
+1 oriented right but forgot mirror

Draw the approximate mapping on each quad if they were textured using the above image.

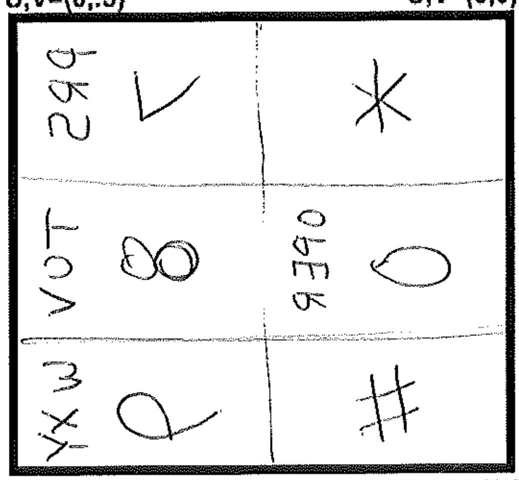
X,Y=(0,1) **X,Y=(1,1)**
U,V=(0, 0.5) **U,V=(0.33, 0.5)**



X,Y=(0,0) **X,Y=(1,0)**
U,V=(0, 0.25) **U,V=(0.33, 0.25)**

3 points

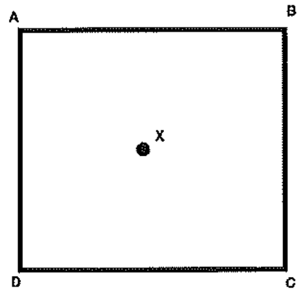
X,Y=(0,1) **X,Y=(1,1)**
U,V=(0,.5) **U,V=(0,0)**



X,Y=(0,0) **X,Y=(1,0)**
U,V=(1,.5) **U,V=(1,0)**

7 points

4. If we have a polygon that looks like



Positions:
A = (0,4,0)
B = (12,4,0)
C = (12,0,0)
D = (0,0,0)
X = (6,2,0)

Normals:
A=(1,0,1)
B=(1,0,1)
C=(1,0,1)
D=(1,0,1)

+1 N= 1,0,1
+1 L= 1,0,1
+1 R= 1,0,1
+1 V= 1,0,1
+2 Normalize before dot product
~~+4~~

The polygon has material property parameters:

Ka = 0, Kd = 0.2, Ks = 0.4, ns = 2

Also there is a both a light and eye point located at (7,2,1). Assume the light has intensity of 1.0.

a) Write the openGL lighting equation.

x3

$$I = L_a k_a + L_d k_d \max(0, N \cdot L) + L_s k_s \max(0, (R \cdot V)^n)$$

b) Using this equation, compute the scalar intensity value at point X using **Phong** shading (we don't have RGB here, just compute a single value). When doing calculations, remember to first **normalize** vectors.

+7

$K_a = 0$ so no ambient.

All normals are (1,0,1) so normal at X is same.

Light(L) is (7,2,1) - (6,2,0) = (1,0,1), same as normal.

$N \cdot L = 1$ since these are the same direction

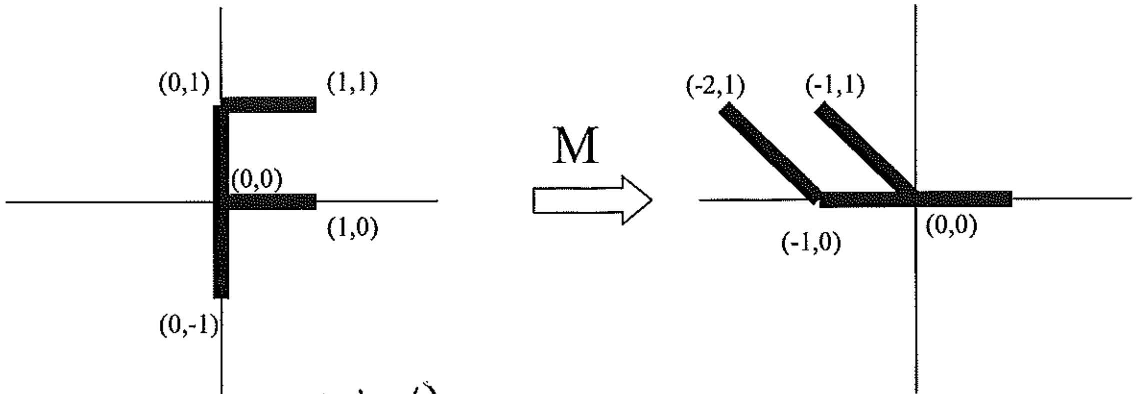
Since normal(N) is same as light(L), we have the reflection is same also, so R also equals (1,0,1).

thus $R \cdot V = 1$.

$$I = 1 \cdot 0 + 1 \cdot 0.2 \max(0,1) + 1 \cdot 0.4 \max(0, 1^2)$$

$$I = 0.6$$

1. A two-dimensional **affine** transformation **M** maps the shape on the left to the shape on the right. **Specify the 3x3 homogenous matrix that represents M using numbers.**



$(x, y) \rightarrow (x', y')$

Eqn-A $(1,0) \rightarrow (-1,1)$

Eqn-B $(0,0) \rightarrow (0,0)$

Eqn-C $(0,1) \rightarrow (-1,0)$

$$\left.\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}\right\} +3$$

↓ fill values into eqns

↓ write the matrix as eqns

$$\left.\begin{aligned} x' &= ax + by + c \\ y' &= dx + ey + f \end{aligned}\right\} +3$$

Eqn-A $-1 = a1 + b0 + c$

Eqn-A $1 = d1 + e0 + f$

Eqn-B $0 = a0 + b0 + c$

$0 = d0 + e0 + f$

Eqn-C $-1 = a0 + b1 + c$

$0 = d0 + e1 + f$

↓ solve these eqns

$c = 0$

$f = 0$

$a = -1$

$d = 1$

$b = -1$

$e = 0$

get an element wrong −2

write in matrix form →

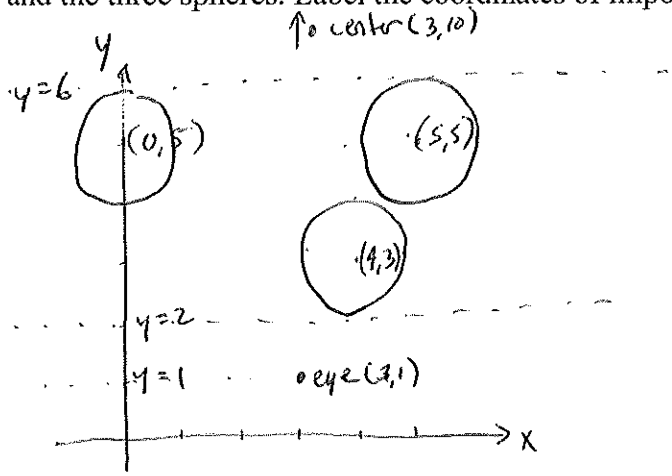
$$\begin{bmatrix} -1 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rot & Shearing – Setting up Matrices R(2) – S(2)

Multiply Correct Matrices +1 (5)

2. Suppose we set up a camera using gluLookAt(eye=[3,1,0], center=[3,10,0], up=[0,0,1]), and then render a scene with three spheres of radius 1, centered at [4,3,4], [5,5,-8], and [0,5,10].

a. Draw a picture of this scene in 2D, showing the X and Y axes, the camera, the lookAt point, and the three spheres. Label the coordinates of important points.



+5

-1 spheres with no area or incorrect size

b. What is the ideal distance to the near plane in this scene? What is the ideal distance to the far plane in this scene? (Give numbers)

+1 near = 1 Not the y value, but the distance.

+1 far = 5

c. Suppose we render this scene with the projection matrix with each of the following calls. In each case how many spheres will we see in the image?

`gluPerspective` (*fovy=1 degree, aspect=1, zNear=1, zFar=100*);

~~Just barely sees the edge of 1 sphere, so 1.~~

+1 Ø The narrow FoV clips the ~~other two~~ spheres.

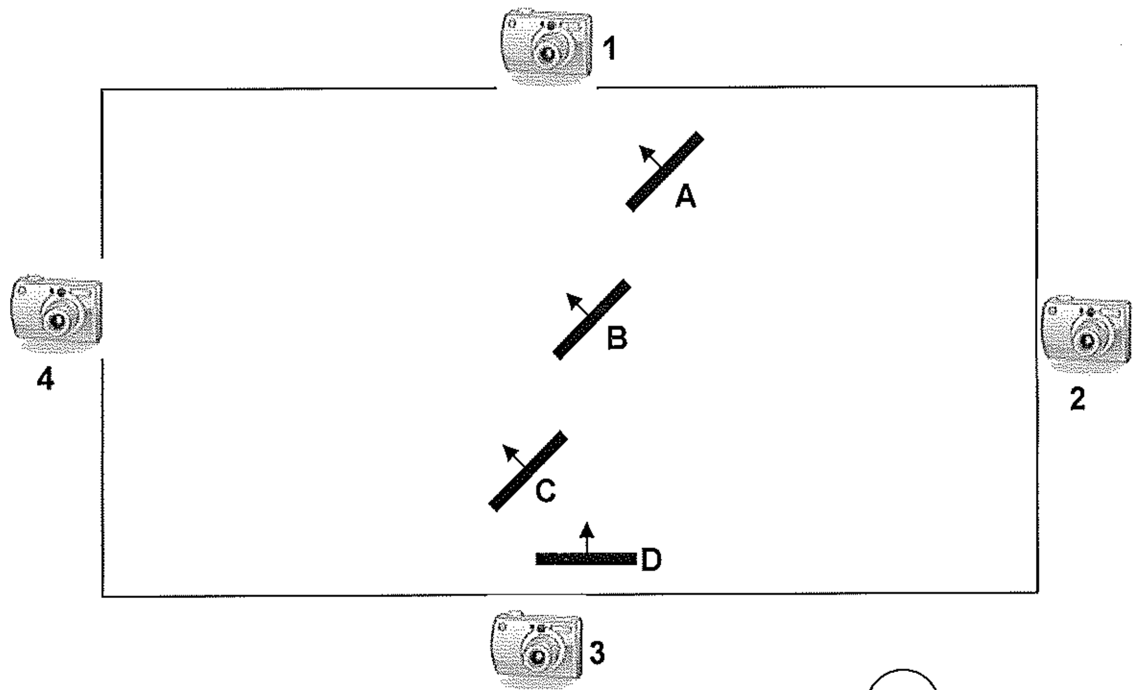
`gluPerspective` (*fovy=175 degree, aspect=1, zNear=3, zFar=10*);

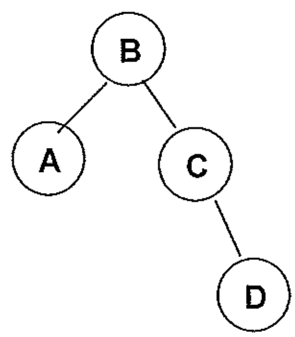
+1 2 The near plane clips the front sphere, So 2 spheres.

`gluPerspective` (*fovy=40 degree, aspect=1, zNear=5, zFar=100*);

+1 Ø The near plane clips all spheres, So Ø spheres.

3. The diagram below shows a scene. The numbered darker lines are polygons, and the arrows show the orientation of the polygon. Below that is a BSP tree constructed from this scene.





a) Suppose we want to render polygons back to front for painters algorithm using this BSP tree. For each of cameras 1,2,3,4 give the rendering order of the polygons.

1) D C B A
2) A B C D
3) A B C D
4) D C B A

b) Suppose polygon B is rotated 90 degrees clockwise. It turns out the BSP tree is still valid. Now what order are the polygons rendered from each camera?

1) D C B A
2) C D B A
3) A B C D
4) A B D C