UNIVERSITY OF CALIFORNIA
College of Engineering
Department of Electrical Engineering
and Computer Sciences
Computer Science Division

Computer Science 184 - Foundations of Computer Graphics

Fall 1994 - Midterm Exam

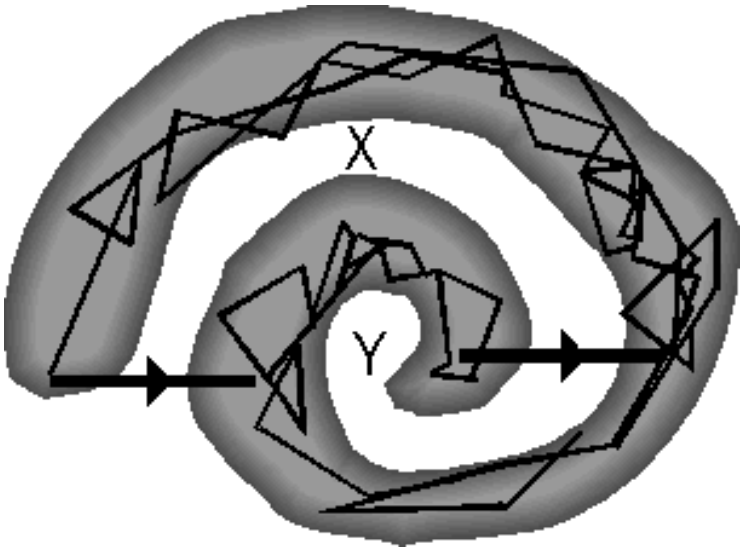
Professor Brian A Barksy
TAs: Dan Garcia and Zijiang Yang

Relax. You have 80 minutes. You will be graded on the best 4 out of 5 problems. This means you should allocate about 16 minutes per question. Remember to pace yourself. Feel free to use the back of each page for additional answer space. Do not panic. You will have time.

GOOD SKILL

Question 1: **Scan Conversion** [25 points]

In the figure to the right, the grey area represents a dense concentration of edges of a very degenerate polygon. You *cannot* tell how many edges there are within that region, nor the direction of them. There are two edges we *are* sure of - these are labeled in bold with the directions specified. There are two topologically distinct regions we care about, X and Y. Each of A-D is worth 3 points for a correct answer, -1 point for a wrong answer, and 0 points if left blank. This is so that a random guessing strategy will, on average, yield 0 overall points. In the future, this will be written [3/-1 points] to indicate the value of a correct/incorrect answer, and will be used for multiple-choice questions.



A)	For the *Odd/Even* rule, is **X**: (circle one)	IN	OUT	*depends-what's-in-the-grey-region*
B)	For the *Non-Zero-Winding* rule, is **X**: (circle one)	IN	OUT	*depends-what's-in-the-grey-region*
C)	For the *Odd/Even* rule, is **Y**: (circle one)	IN	OUT	*depends-what's-in-the-grey-region*

D) For the *Non-Zero-Winding* rule, is **Y**: (circle one) IN OUT *depends-what's-in-the-grey-region*

Questions E-G concern rotational invariance (i.e. does the category in question remain constant after an arbitrary rotation)

E) Is *region classification* (i.e. whether a region is in or out) by the *Non-Zero-Winding* rule *rotationally invariant*? [3/-1 points]

(circle one) Yes No Depends on _____

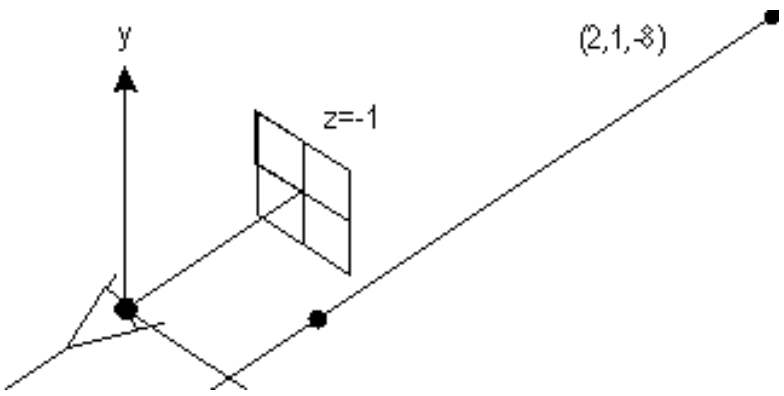
F) We specify the color (R,G,B) for every corner of a polygon (as in one of you assignments). We are then given an ideal system in which each pixel is infinitely small (thus we don't have to worry about all of the nitpicky details that arise due to corners falling between scanlines like special cases for convex corners, incrementing the attributes by that additional wierd factor when first added to the edge_y_start list, etc.). We then color the inside of the polygon using a linear interpolation ("lerping") strategy exactly as in your handout. Think very carefully of different types of polygons and the colors that could be chosen when you answer this question. Is the *color* inside the polygon *rotationally invariant*? ("Yes" mean that EVERY polygon's color is rotationally invariant, regardless of the colors and type/size of the polygon, "No" means that NO polygon's color is rotationally invariant, regardless of the colors and type/size of polygon)[6/-2 points]

(circle one) Yes No Depends on _____

G) *Why*? (provide a sketch if it helps)[4 points]

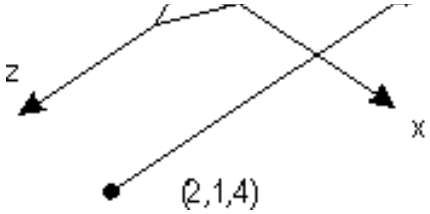
Question 2: **Projections** [25 points]

The plane of projection is the z=-1 plane. You place your eye at the origin and look toward the negative z-axis. Your view-up vector is the y-azis. There is a line segment up and to your right running from the point (2,1,4) to the point (2,1,-8). For this question *ignore clipping*.



A) Where do the line segment endpoints *project* on

the plane? [10 points]



B) *Sketch* the projection of the line segment on the graph below. [5 points]

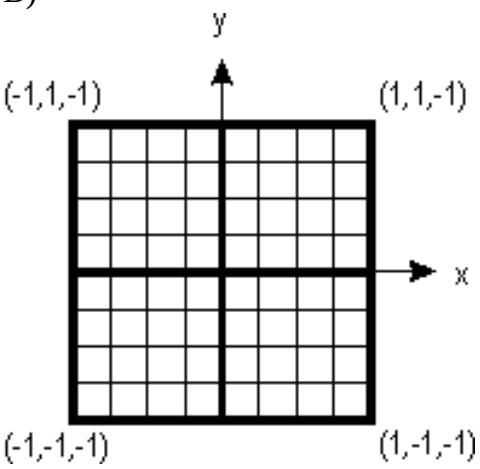
C) Express, in homogeneous coordinates, the *instersection* of the line defined by the projector fro the eye to the point (2,1,0) with the plane. [10 points]

Answer 2: **Projections**

A) (2,1,4) projects to (___, ___, ___)

(2,1,-8) projects to (___, ___, ___)

B)



C) The projector from the eye to the point (2,1,0) intersects the plane at the following homogeneous coords: (___, ___, ___, ___)

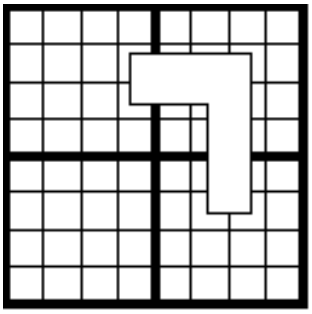
Question 3: **Potpourri** [25 points]

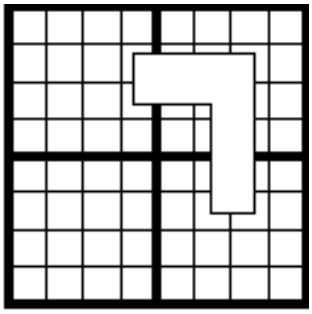
A) What is the *magnitude* of the vector in the second column of a 4× 4 homogeneous rotation matrix that rotates 57 degrees about the point (1,3,4)? [3 points]

B) At a swap meet you buy a 1000× 1000× 9 bit framebuffer system with no colortable. You decide to bread up each byte allocated for your pixel the following way: you map 4 of those bits to red, 3 to green and 2 to blue. The DACs for red, green and blue each have 4 bits. When the number of bits of a signal is less than the number of bits of the DAC (as in the green and blue case), you short to ground (binary 0) the *upper*, most significant pins. What is the *maximum number* of *non-grey-colors* you can have with your current scheme? Grey is as we've defined it before, R=G=B. A DAC converts digital information to analog information - e.g. if 0010 were present on the digital input lines, the DAC would output 6 volts, and if 0000were present on the digital input lines, the DAC would output 0 volts. [3 volts]

C) Using the numbers from question (B), what is the *maximum number* of *non-grey-colors* you would have on the screen at one time if you bought a colortable? [3 points]

D) When using the tiling method of damage repair, it is often crucial to choose a reasonable tile size. Student #4 chooses a tile size of 4 pixels/tile and student #1 chooses a tile size of 1 pixel/ tile as shown in the figures to the right. If the user removes the L-shaped object to the right, #1 has to search *more* tiles than #4, and overall do *more work*. In the two figures to the right, assume the L-shaped figure is present. Assume we are going to add a graphical primitive which is contained entirely within a single pixel We are going to then remove that primitive and see which student does more work. There are some pixels that force #4 to do more work than #1, and some pizesl that have both #1 and #4 doing the samm amound of work. Shade in *any pixel* for which work(#4)>work(#1) in the picture on the left, and shade in *any pixel* for which work(#4)>work(#1) in the picture on the right. [3 points each]





E) Given a polygonal dataset of Soda Hall containing over a million polygons, a student suggests using a calligraphic display for it. Give two resons why this would be a bad thing. [4 points]

F) In one sentence, how is a 2-D draw or paint application (as discussed in lecture) *modal*? [3 points]

G) What is one reasion someone might want to allow *cycles* within hierarchies? [3 points]

Answer 3: **Potpourri**

A)

B)

C)

D) (please write your answer on the figure above)

E 1)
E 2)

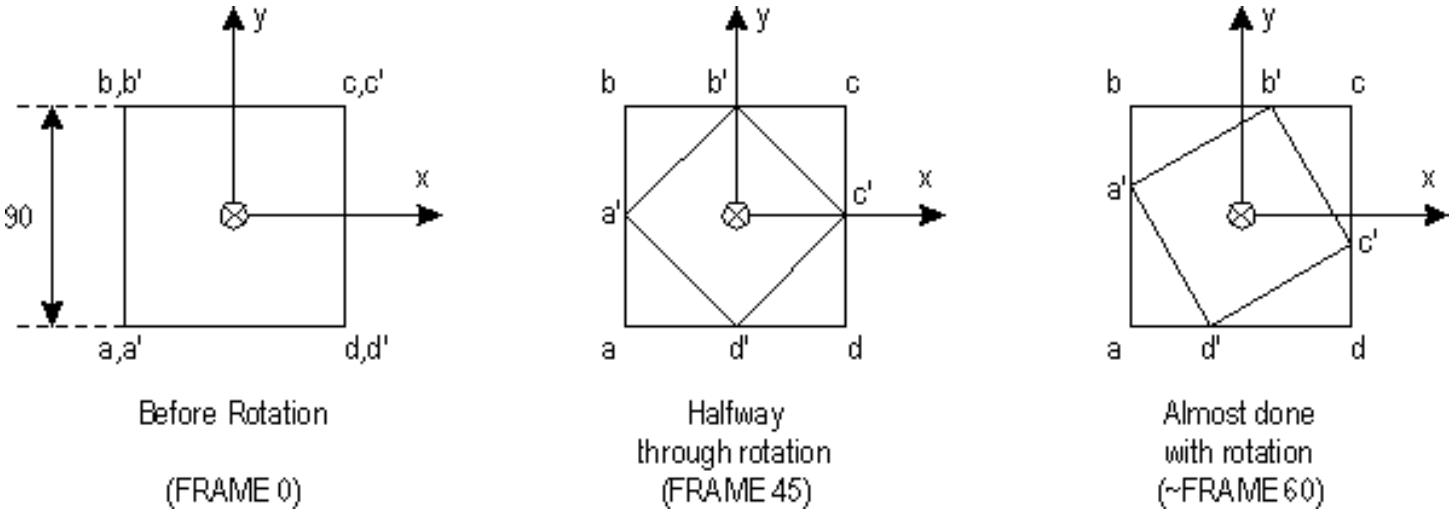
F)

G)

Question 4: **Object Motion Specification** [25 points]

(This question was suggested by a student and has been tweaked a bit)

We are interested in spinning a 90× 90 square around its azis in a clockwise direction. By FRAME number 90, we would like the square to have completed a full 90 degree rotation. The complicated part is that we'd like the square to remain touching the edge of the original bounding square. In the diagram below, the bounding square is defined by the polygon abcd, and the moving square by a'b'c'd'. Point a' should move between a and b from FRAMEs 0 through 90, and similarly for the other points. Your job is to fill in the proper parameters to the scene description file. We have provided an outline for you, all you need to do is fill in the missing values. The language below is very much GLIDE-like, but we've added some embellishments for this quesiton (e.g. strong identifiers). *We do NOT care what happens when FRAME grows larger than 90*.



A) The point abcd should move at a *constant linear velocity*. [12 points]

B) The square should rotate at a *constant angular velocity clockwise* as shown above. Keep in mind that in this case, you might want to reference the square (a'b'c'd') *without* a rotation in other groups. [12 points]

C) Are the motions specified by A and B the same? [1 point]

Answer 4: **Object Motion Specification**

```
# Valid transformations in this language are exactly as in GLIDE, i.e.
#
# transformation-statement ::= translate translationvalue |
#                              scale scalevalue |
#                              rotate axisvalue anglevalue |
#                               <empty-string>
#
# Also, note that all the values above are triples which can be either
# static or dynamic.
#
# value ::= ( <FLOAT> <FLOAT> <FLOAT> )
#
# Dynamic values are surrounded by $ signs.  The only dynamic value you should
# need for this program is FRAME : an int indicting the current frame number
# since the application started running.
```

A,B) Please write your answer on the next page. You may express any numberic computation algebraically.

C) Are the two motions the (circle one) Same? Different?

```
## PART A - constant linear velocity
##          of points.

point a (-45 -45 0)
point b (-45  45 0)
point c ( 45  45 0)
point d ( 45 -45 0)

## Here is where you need to fill in
## the arguments to point.

point a' (                        )
point b' (                        )
point c' (                        )
point d' (                        )

# We don't care about the surfaces

face abcd_face  (a  b  c  d )
face abcd'_face (a' b' c' d')

object abcd_obj (abcd_face)
object abcd'_obj (abcd'_face)

group abcd_group
    instance abcd_obj
    endinst
endgroup

group abcd'_group
    instance abcd'_obj
    ## Here is where you fill in
    ## transformation statements,
    ## if you need any.




    endinst
```

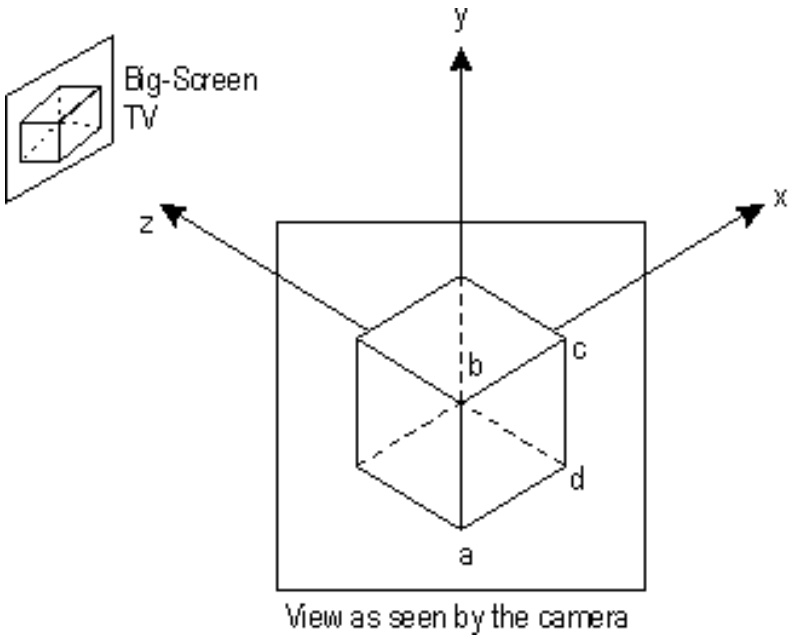
```
## PART B - constant rotational
##          velocity of object.

point a (-45 -45 0)
point b (-45  45 0)
point c ( 45  45 0)
point d ( 45 -45 0)

## Here is where you need to fill in
## the arguments to point.

point a' (                        )
point b' (                        )
point c' (                        )
point d' (                        )

# We don't care about the surfaces

face abcd_face  (a  b  c  d )
face abcd'_face (a' b' c' d')

object abcd_obj (abcd_face)
object abcd'_obj (abcd'_face)

group abcd_group
    instance abcd_obj
    endinst
endgroup

group abcd'_group
    instance abcd'_obj
    ## Here is where you fill in
    ## transformation statements,
    ## if you need any.




    endinst
```

```
endgroup

group ourworld_group
    instance abcd_group
    endinst
    instance abcd'_group
    endinst
endgroup

# Ignore hidden and shadingflag

render ourworld_group
```

```
endgroup

group ourworld_group
    instance abcd_group
    endinst
    instance abcd'_group
    endinst
endgroup

# Ignore hidden and shadingflag

render ourworld_group
```

Question 5: Projections and Football Games [25 points]

Cal decides to add TV coverage to their football games. Every once in a while the camera will turn to the crowd and focus on a prticular fan, and the smiling fan will appear on the huge outdoor screen. Most of the time the fan will not be able to figure out from looking at the image of themselves on the big screen where the camera is. Your job is to figure this out given a particular example. You may make the assumption that the camera's view plane normal is parallel to its direction of projection).



Your head can be modeled as a unit cube centered at the origin. You ar looking directly down the positive z axis. We've labeled the back face of your head - the lower-left point is a, and b, c and d follow clockwise around the back of your head. *Assume that the camera is in a blimp and has a very powerful lens - it is very very far away*.

A) What *classification* of projection is it? *Please be as specific as possible.* (e.g. Perspective->Seven Point->Spamographic->etc.) [3 points]

B) *Where* is the camera? (i.e. What is its COP / DOP as it would have been specified in a descriptor file, e.g. Glide) [3 points]

C) If you were to rotate your head to face the camera about two fo the orthogonal axes, what *two axes* and *angles* would you rotate around? e.g. Rotate(axis[either x, y, or z], θ) [8 points]

D) What *single* axis would lyou rotate about to get the same *effective rotation* so that you were facing the camera? (ignoring twist - i.e. your final vuv might not be the same, but don't worry about that.). Your answer should be

expressed as a single axis. (Hint: that axis will probably *not* be one of the orthoganol axes) [6 points]

E) By how many *degrees* should the rotation be in question (D)? Please leave the answer in symbolic form. (i.e. don't use a calculator...) [5 points]

Answer 5: **Projections and Football Games**

A)

B) (___, ___, ___)

C) First: Rotate (___, ___) then Rotate (___, ___)

D) (___, ___, ___)

E) (_____)
