MAE 473/573
Graphics in CAD
Fall, 2001

Registration #: 497656
Time: M W F, 16:00 - 16:50
Location: 97 Alumni

Exam #2

Rules:

- Closed book and closed notes
- Write your answers ONLY in the space provided, **or you will lose points**.
- Calculators **are not** permitted
- 3 questions, 55 minutes total duration
- ***Point allocation: Problem 1: 50, Problem 2: 20, Problem 3: 30***
- If you have a question, please raise your hand
- No conversation whatsoever
- Academic dishonesty is not tolerated
- Good luck!

Name: ____

Signature: ____

1. Short answer questions. **Write your answers only in the space provided beneath each question, or you will receive zero credit.** Be concise, or you will run out of time.

a. Very briefly define binocular disparity.

The difference in views depicted by L, R eyes (interocular e = 0)

b. Very briefly describe the difference between the "transform on Draw" method and the "transform then Draw" method.

transform on draw - transform each object in graphics ~~[illegible]~~ each time you draw. transform then draw - transform objects & store their new values - independently of drawing.

c. Z-buffering makes use of 2 distinct <u>buffers</u>, which pertain to what? List them, **using only 2 words**.

depth, color

d. When mapping a texture to a polygon, it is usually the case that the dimensions of the texture do not match the dimensions of the polygon. There are two forms of this occurrence - minification and magnification. Briefly describe what these mean, and suggest one approach that that is used to deal with both problems.

magnification - when the texture pixels cover more than one polygon pixel

minification - the texel covers less than one pixel - averaging is one solution.

e. **Using only 3 words**, list 3 shading shortcuts:

Constant, Phong, Gouraud

f. Authors have argued that *sorting* is the key to algorithmic shading efficiency. VERY BRIEFLY outline the sorting mechanism of the following algorithms. Specifically, **<u>all we want you to list is</u>: which** coordinate axis or axes are involved for each algorithm, and **in what sequence**? (This is right out of your notes).

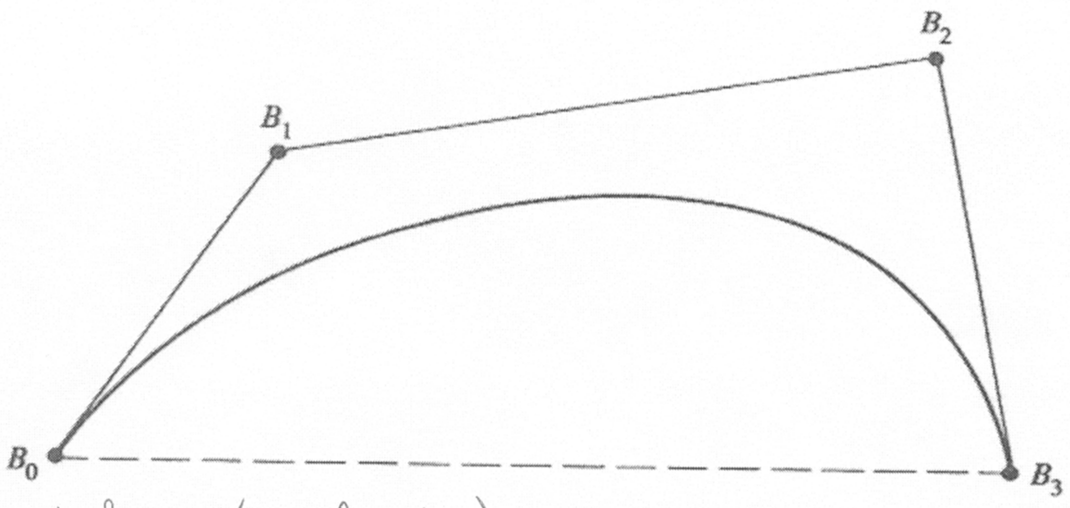
Painter's: Z then X, Y

Scan-line: Y then X then Z

Warnock: Parallel X, Y, then Z

Z-buffer: Z only

g. Briefly list 3 relevant facts about the figure shown below:



- A Bezier (not B-spline) curve
- $B_0 \rightarrow B_3$ are control points
- $B_0 \rightarrow B_3$ comprise a defining polygon
- Internal control points do not lie on curve
- First / last control points lie on curve

h. List one advantage and one disadvantage of Coons patches.

A: combine both linear interpolation models from lofted / ruled surfaces

D: must subtract duplicate terms which arise at corners / edges

i. Observe the following table which describes lighting and material properties in 3 different OpenGL programs. Assume that the minimum and maximum values for material and lighting colors are 0 and 1 respectively. **In the space provided**, briefly describe how each surface will appear:

	lighting			**material**		
	ambient	*specular*	*diffuse*	*ambient*	*specular*	*diffuse*
	r g b	r g b	r g b	r g b	r g b	r g b
1)	1 0 0	0 0 0	0 0 0	1 1 1	1 0 0	0 1 0
2)	1 1 1	0 0 0	0 0 0	0 1 0	0 0 0	0 0 0
3)	0 0 0	0 1 0	1 1 1	1 0 0	1 1 0	0 0 0

1) red dull

2) green dull

3) green shiny

j. An appropriate relation between “e” and “d” is as follows, as recommended by author *Hodges*: **e=0.028d**.

- Please very briefly list what “e” and “d” represent:

e - interocular distance

d - distance from display plane to camera|eye location

- Assuming that the **negative parallax** p=4 inches, and d=150 inches, will hyperstereo (undesirable) occur? Briefly explain using a simple calculation.

$e = 0.028(150'') = 4.2''$; $\boxed{e \geq p}$

No hyperstereo

[redacted]

2. Pseudo-code questions. **Write your answers only in the space provided beneath each question, or you will receive zero credit.** Be concise, or you will run out of time.

- Assume you are writing an OpenGL program that requires navigation with the mouse around a racetrack. Assume the mouse button is down and the user is driving through your world. You are given the following variables to work with:

WindowWidth	//the current width of the window
WindowHeight	//the current height of the window
MouseX	// the current X value of the mouse cursor location
MouseY	//same as above, with Y
RotY	//the number of degrees that the camera is rotated about the Y axis
MaxRotChange	//the maximum change in rotation allowed at a time
deltaRotY	//a temporary holding variable

Write in pseudo-code, a function called UpdateRotY which takes the above mouse cursor location, and the other variables, and computes a new value for RotY (Camera rotation about the Y axis). If the mouse is in the center of the screen, the rotation should not change. If the mouse is all the way to the left, change RotY by -MaxRotChanged (and the opposite if the mouse is all the way to the right of the screen). If the mouse is anywhere else, adjust RotY by its proportion of the screen width. Assume the coordinates of the upper left corner of the window are (0,0), and the coordinates of the lower right corner are (WindowWidth,WindowHeight). **Use only the variables listed above, and do not use any OpenGL functions**. You may ignore translation.

```
UpdateRotY()
{
deltaRotY = $\left(\frac{MouseX * 2}{Screen\ width} \cancel{*} -1.0\right) * MaxRotY$
RotY += delta RotY

}
```

[redacted]

- You are writing an OpenGL program that consists of drawing a number of cubes at various locations and orientations (translations and rotations). Assume you have a generic function called DrawBasicCube which draws a cube at the origin. Assume further that there is a Cube data structure which contains the 3 Rotation values (RotX,RotY, and RotZ) as well as the X, Y and Z translation values.

Assume that you are given the following variables:

NUMCUBES	// the number of cubes
Cube mycubes[NUMCUBES]	//an array of Cube structures

Assume you have the following functions available to you:

Push()	//push the current modelview matrix onto the stack
Pop()	//pop the previous modelview matrix off the stack
Translate(int x,int y,int z)	//multiply the current modelview matrix by a translation matrix where x,y and z are the translation values
Rotate(int rotx, int roty, int rotz)	//multiply the current modelview matrix by a rotation matrix formed with the three rotation variables

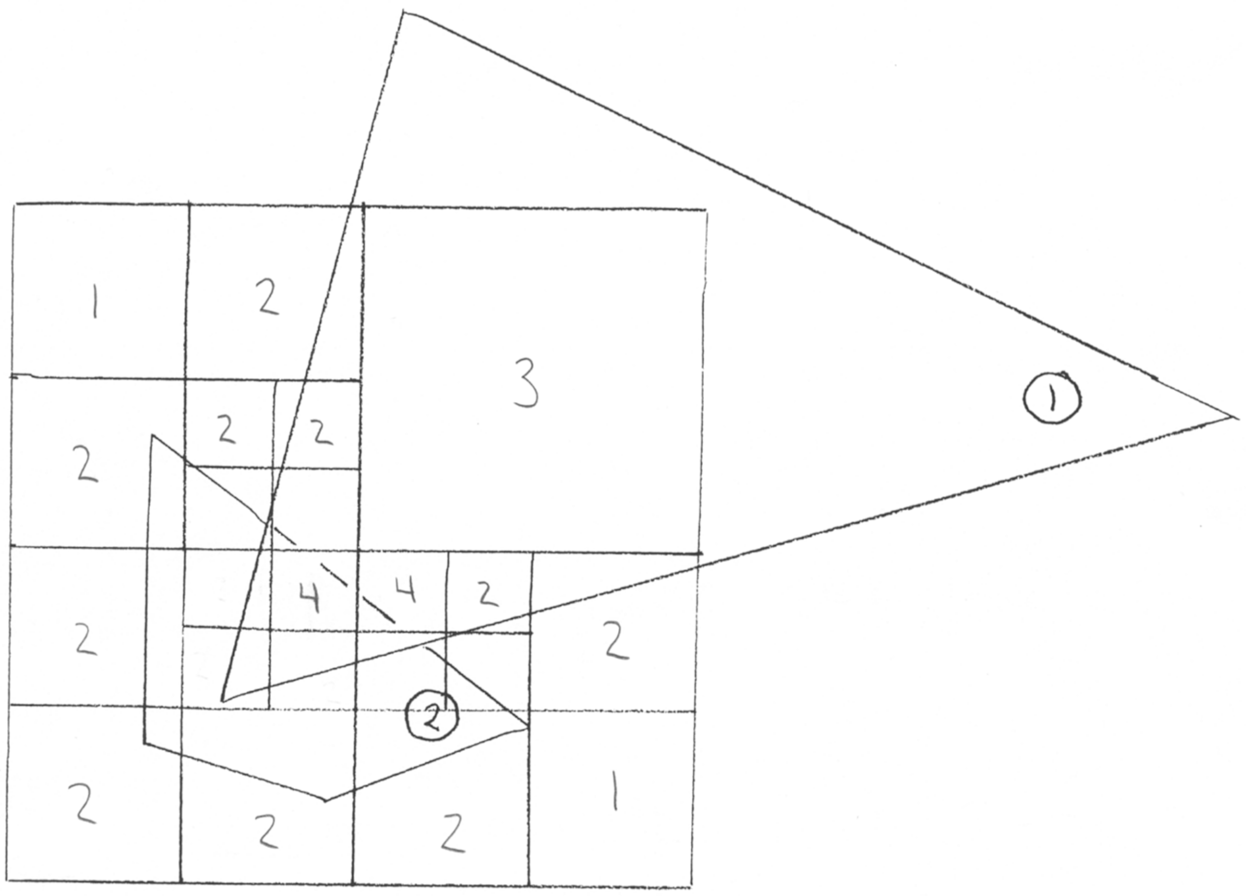
In pseudo-code, write a function called DrawCube. **Use only the pseudo-code functions provided and do not use any OpenGL functions**. The function should take 1 parameter- the index of the cube that you're drawing. This function should set up the modelview matrix with rotation and translation, and then draw the cube itself (you can simply call DrawSimpleCube to accomplish this). Note: be sure to set the rotation in the correct order (i.e. the OpenGL order) and be sure to leave the modelview matrix the way you found it.

```
DrawCube(int index)
{
Push()
Translate (mycubes[index].X, mycubes[index].Y, mycubes[index].Z);
Rotate( mycubes[index].RotX, mycubes[index].RotY, mycubes[index].RotZ);
DrawBasicCube();
Pop()
}
```

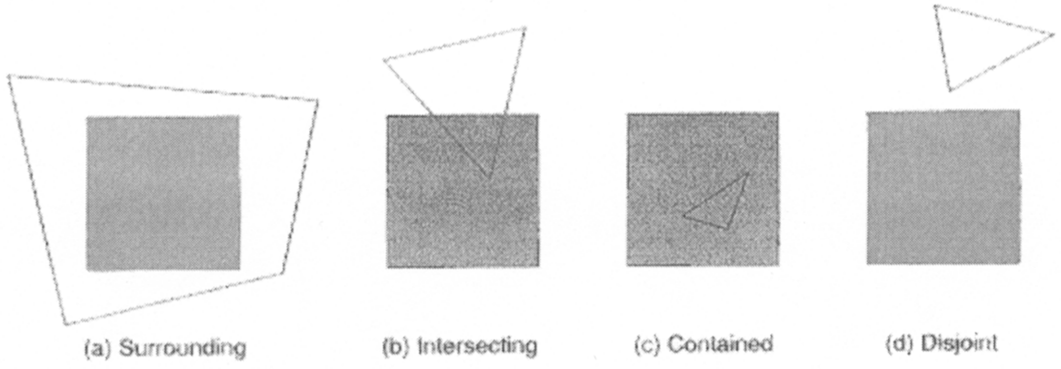
[redacted]

3. Short Calculations. **Write your answers only in the space provided beneath each question, or you will receive zero credit.** Be concise, or you will run out of time.

a. Please observe the following 2 polygon sketch. Note that Polygon 1 (triangle) is **in front** of Polygon 2 (4-sided "trapezoid"). The square (which has been repeatedly subdivided) represents the display plane.



Hints:



- Warnock's algorithm defines 4 types of <u>relationships</u> between a polygon and the display plane – briefly state and number them:

1- All polygons disjoint

2- 1 intersecting OR contained polygon

3- 1 surrounding polygon <u>ONLY</u>

4- 1 surrounding polygon <u>afront</u> other intersecting | contained | surrounding polygons

- Demonstrate your understanding of the Warnock algorithm by labeling (with numbers) the sketch above, just like we did in lecture. If squares exist that require further subdivision, DO NOT LABEL THEM.

b. Given the following Bernstein function for a cubic B-spline:

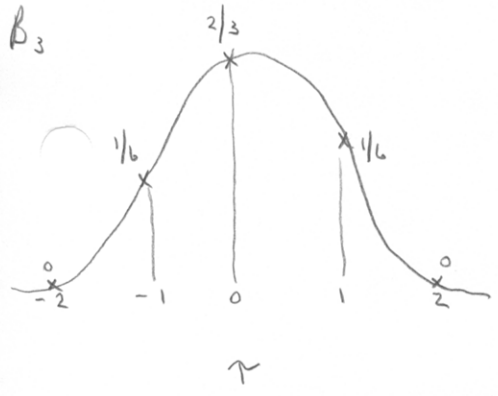
$$\boxed{f(0) = \sum_{i=-1}^{n+1} a_i \beta_3(nt - i)}$$

Assuming that there are 4 control points, IN THE SPACE PROVIDED, write out the above summation making sure to **cancel any terms that zero out**.

$\boxed{C.P = 4;\ n = 3!}$ $\boxed{f(t),\ \text{where}\ \underline{t=0}!}$

$$f(0) = \sum_{i=-1}^{4} a_i \beta_3(3t - i)$$

$$= a_{-1}\beta_3(0 - -1) + a_0\beta_3(0-0) + a_1\beta_3(0-1) + \cancel{a_2\beta_3(0-2)} + \cancel{a_3\beta_3(0-3)} + \cancel{a_4\beta_3(0-4)}$$



$$\boxed{= \frac{1}{6}a_{-1} + \frac{2}{3}a_0 + \frac{1}{6}a_1}$$

Bonus questions:

- Very briefly define Quantitative Invisibility.

Used in Appel's - represents the count of the # of faces between a given polygon edge and the Viewpoint - must be ZERO for visibility.

- State the Phong Illumination model for Specular Reflection, and describe (in one or two words, and underneath the equation) each of its components.

$$I = w(\theta) \cos^{\wedge} \alpha \; I_p$$

intensity of pt. light source

pixel intensity

fraction of specularly reflected light

angle between view vector + reflected ray

specular reflection exponent [0, 100]

- A plane is defined by the following equation: 2x-3y+5z=1. **Using a very brief matrix multiplication procedure discussed in lecture**, show that vertex (1 1 1) <u>does not</u> lie on this plane.

$$[P^T][P] = 0 \ ?$$

$$[P] = ax + by + cz + d = 0 = \begin{bmatrix} 2 & -3 & 5 & -1 \end{bmatrix}^T$$

$$[P] = \begin{bmatrix} x & y & z & H \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ -3 \\ 5 \\ -1 \end{bmatrix} = 2 - 3 + 5 - 1 = \underline{\underline{3 \neq 0}}$$