CS559 Final Exam
December 22, 2005

Mean score = 64.3 Median score = 61
I tried two grading systems: one was the "it was a hard semester, give everyone a Holiday Gift"
A > 78 AB > 64 B > 55 BC > 48 low scores given C

The other was the "this is how it really should have been graded" scale
A >= 90 AB >= 80(*) B >= 65 BC >=55 C >= 51 CD >= 43 D >= 40

In all 32 exams, only 1 person was "on the fence" so I gave them the extra point (*).
What's amazing about this is that switching between these two grading systems give very different final exam scores, but did not change anyone's grades. (Actually, its not so amazing, the final doesn't count for much. And we were very generous with the grades for the Assignments – 2/3s of the students got As.)

So I've chosen to use the harsher scale. That way, I can feel like the exam was graded fairly. Your exam grade indicates how well you really did on the final. And no one's permanent record is affected.

Overall, the class grades were very good. The mean and median grade is an AB!
(I don't mind the final grades being so high – this is a hard class, and you've all put in a lot of effort.)

Question 1: Visibility (15 pts)

A) Give two situations when drawing order matters when using a Z-Buffer, even if there is infinite precision for the Z-values.

1) Objects at the same distance

2) Semi-Transparent / compositing / alpha-blending

3) Z-buffer aliasing (partial pixel coverages)

Many people said intersecting polygons. Technically, this is right because it causes a form of either 1 or 3 (and we gave you credit). However, if you gave this answer, your explanation in the next part had to be carefully thought out.
Another correct answer given by someone is a particle system which must be drawn last because it combines 2 and 3.

B) Can a BSP tree help with these problems? If so, explain how. If not, explain why not.

1) No (choice still made as an artifact of the ordering). Credit was given for yes, since at least the tree will make sure its consistent – if you explained that.
2) Yes (since things get ordered front to back)
3) Yes (higher precision in the split polygons make things work out)

Basically, even if we didn't agree with your answer, we gave you points if you had a reasonable argument.

C) There are certain situations in which the painter's algorithm does not work. Describe one, and explain how BSP tree methods address the issue.

Any non-orderable triangles (intersection or cycles) are a problem for the painter's algorithm.
BSP Tree IS the painter's algorithm, so the key point is that it splits problematic triangles.

Question 2: Methods for Graphics (15 pts)

Here is a list of graphics methods discussed in class:

1) Line-integral Convolution
2) Focus+Context Displays
3) Direct Volume Rendering
4) Hedgehogs (arrow glyphs)
5) Marching Cubes
6) Gooch Shading
7) Brezenham's Algorithm
8) Flood Fill
9) Photon Mapping
10) Median Cut
11) Mean Shift
12) Radiosity
13) Recursive Ray-Tracing
14) Bi-Directional Ray Tracing
15) Local Lighting (like in the OpenGL model)
16) Environment Maps
17) Displacement Maps
18) Bump Maps
19) B-Splines
20) Cardinal Splines
21) Catmull Clark Surfaces
22) Loop Subdivision Surfaces
23) Modified Butterfly Surfaces
24) Reaction-Diffusion
25) Floyd-Steinberg Error Diffusion
26) Dithering
27) Ordered Halftoning
28) Vector Quantization
29) Huffman Coding
30) Lempel-Ziv Coding
31) Run-Length Encoding

For each of the following, give the numbers of the method(s) that apply. If there are multiple approaches, list all of them. Incorrect and missing answers will be penalized.

A) Generate curves or surfaces that interpolate their control points:

20, 23 - 7 is actually correct, but we didn't take points off if you missed it

B) Convert a scalar field to an iso-surface (or contour plot):

5 – 3 isn't actually correct, but we counted it as such

C) Do lighting for all LDS*E and LS*E paths, but not LDDE paths:

13, 14 (9 can do LDDE paths)

D) Convert a grayscale image to black and white:

25,26,27 - we didn't count 28 as wrong, but it is a misuse of the term

E) Make a mirrored ball reflect the sky and ground:

13, 14, 16 (9 is right too, but you weren't expected to know that)

F) Perform Lossy Compression of an image or video sequence:

28 (palette reduction can be considered lossy compression so 25-27 are OK)

Question 3: Hack (er… Real-Time) Rendering (12 pts)

A) An object is drawn using a texturing trick to make it look bumpy. How can you tell if the method was Bump Mapping or Displacement Mapping?

With bump mapping, the surfaces is really smooth – so you can:
(any of these is OK)
look at its profile to see if there are bumps
get really close
look for self-shadowing
turn off the lighting and see if things still look bumpy

B) Some rendering methods require first drawing the scene from a different viewpoint than the camera and then using the result to create complex lighting effects. Give two examples of such techniques, and explain the effect that they provide.

Shadow Maps – draw from light position to make interobject shadows
Environment Maps / Reflection Maps – draw from "mirror view point" to make reflection

One person mentioned the forwards pass of a bi-directional ray tracer, which actually is a correct answer!

C) Describe the simplifying assumptions made by environment mapping.

That the object is very small relative to the outside world
(the other main assumptions follow from that, so if you got that, you got full credit)

Other assumptions:
The object is a sphere viewed by a distant viewer so only normals matter
The object only reflects the distant stuff (no inter-reflections)

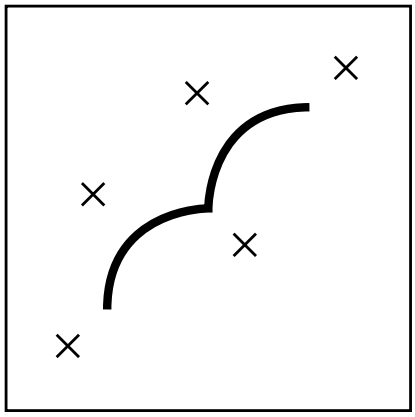
Name: __KEY_____

Question 4: B-Splines (6 pts)

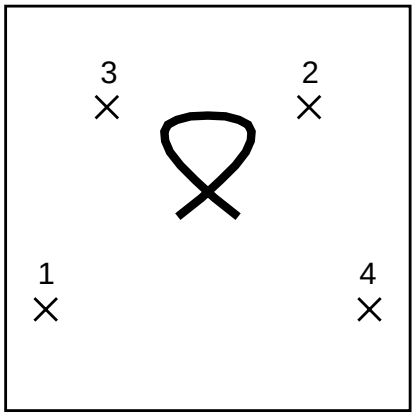
Each of the following curves is shown with its control points. No control points are repeated. For each one, either give a reason why it could not possibly be a cubic B-Spline or say that it could be a cubic B-Spline. If a curve cannot be a B-Spline, there will be some obvious violation of a property of B-Spline curves – state what that property is.

A) is a B-Spline
B) doesn't stay inside of the convex hull
C) violates the variation diminishing property (e.g. it has too many wiggles)

Example:

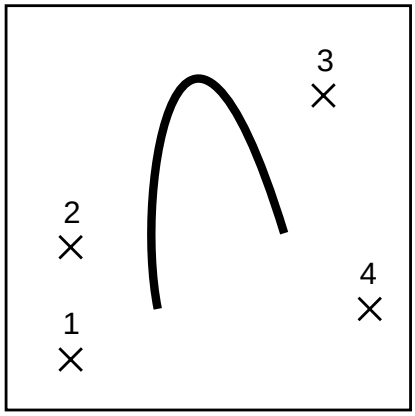


A)

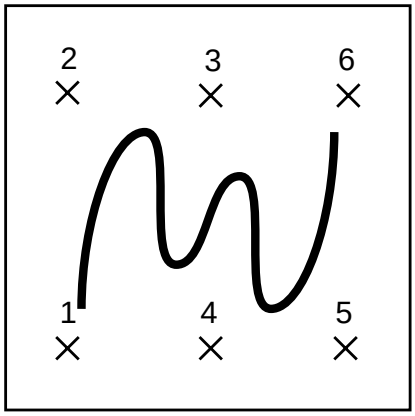


Not a B-Spline since B-Spline would be C(2) and this curve has a kink (e.g. is only C(0)).

B)



C)



Question 5: Continuity (10pts)

Consider the curve:

F(u) = 0 < u <= ½ 0,u
½ < u <= 1 a+bu, c+du

If a and b are 0 can the curve be c(1) (if so, give the values for c and d)?

Yes, c=0, d=1 (1pt for each)

If a and b are 0, can the curve be c(0) but not g(1) (if so, give the values for c and d)?

No (2pts)

If a and b are 0, can the curve be g(1) but not c(1) (if so, give the values for c and d).

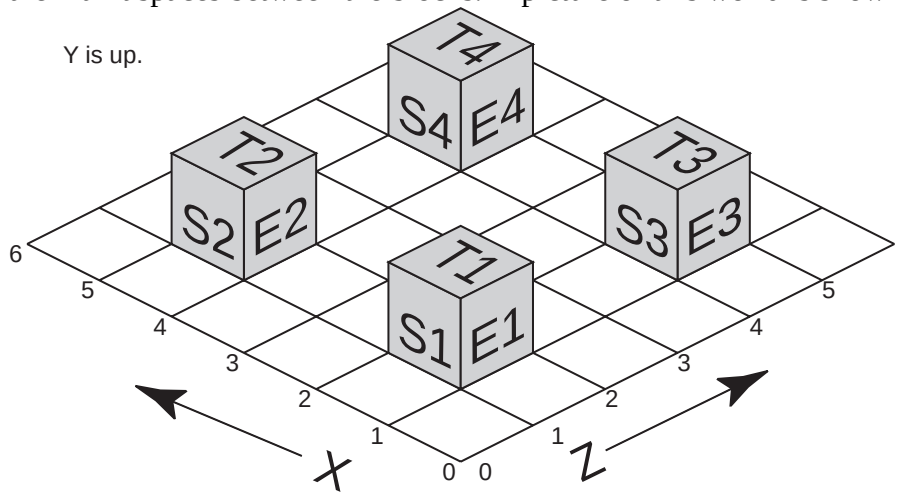
Yes – any values where d is not equal to 1 and c = $-\frac{1}{2}$ d + $\frac{1}{2}$ (1pt for each)

If b is **not** zero, can the curve be g(1) (if so, give the values for c and d)

No (2pts)

Question 6: Camera Models (15 pts)

Consider a world with 4 blocks (numbered 1 to 4 in it). Each face of the blocks has the letter (N,S,E,W,T,B) for which side it is on (North, South, East, West, Top, Bottom). The blocks are unit size (each edge has length one). All blocks are placed on the ground (y=0). The first block is placed one unit away from the origin (its corner is at 1,0,1), and there are 2 unit spaces between the blocks. A picture of this world is shown:



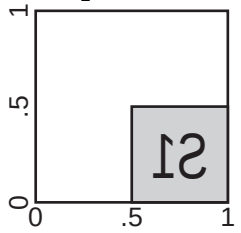
Consider viewing this world with an orthographic camera. The camera's film plane is the XY plane (Z=0), and is the unit square (between 0 and 1 in X and Y – note: it is NOT centered at the origin). The viewing direction is down the negative Z axis.

Provide the viewing matrix (4x4) that generates the following views:

This question is actually pretty hard. We gave lots of partial credit, and even gave full credit for "almost right" answers on part C.
The secret is not to try to make rotations (or to invert the matrix) but rather, to look at where the different vectors need to go. The thing that makes it hard is that in the pictures, the X axis is going backwards, and that the camera has to look in the negative Z direction.
Also, there are some choices in how the Z axis is mapped. It needs to point away from the thing you're looking at, but its position and scale can be set in many different ways

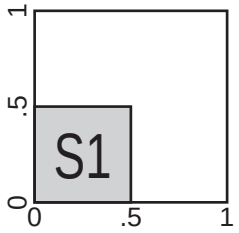
Name: __KEY_____

Example:



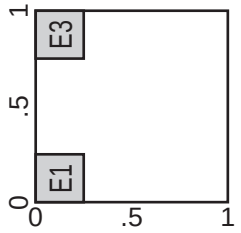
1/2	0	0	0
0	1/2	0	0
0	0	-1	0
0	0	0	1

6A:



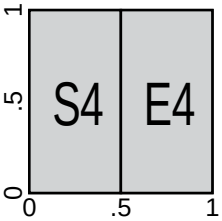
- ½	0	0	-1
0	½	0	0
0	0	-1	0
0	0	0	1

6B:



0	-1/4	0	¼
0	0	¼	-1/4
-1	0	0	0
0	0	0	1

6C:

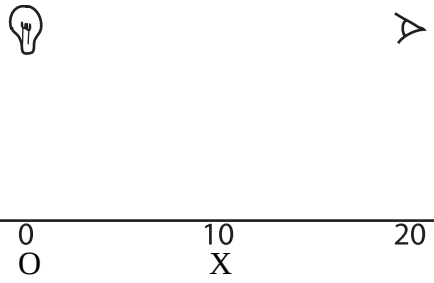


- ½	0	½	½
0	1	0	0
- ½	1	½	11/2
0	0	0	1

Yes, there were multiple people who got the entire thing correct, and at least one more who got full credit.

Question 7: Lighting and Shading (16pts)

(Assume everything in this question is in the Z=0 plane, and that the Phong lighting model is used).
A point light source L is placed at (0,10,0). The eye point E is placed at (20,10,0).



(the O goes under the lightbulb, the X goes under the 10)

Place an X on the point on the ground where the specular component of the lighting has its maximum brightness.

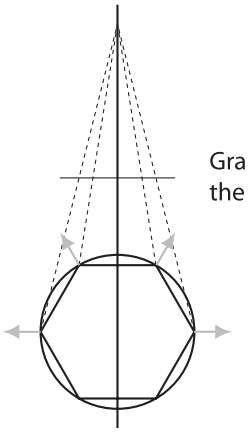
Place an O circling the point on the ground where the diffuse component of the lighting has maximum brightness.

Name: __KEY_____

7. Lighting and Shading (continued)

A sphere at the origin is approximated by a very small number of polygons. The camera is placed at (0,20,0) and faces the sphere. A directional light source sends light downward from infinity along the Y-axis, as shown here:

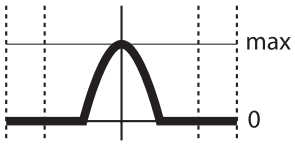
Light is DIRECTIONAL downward



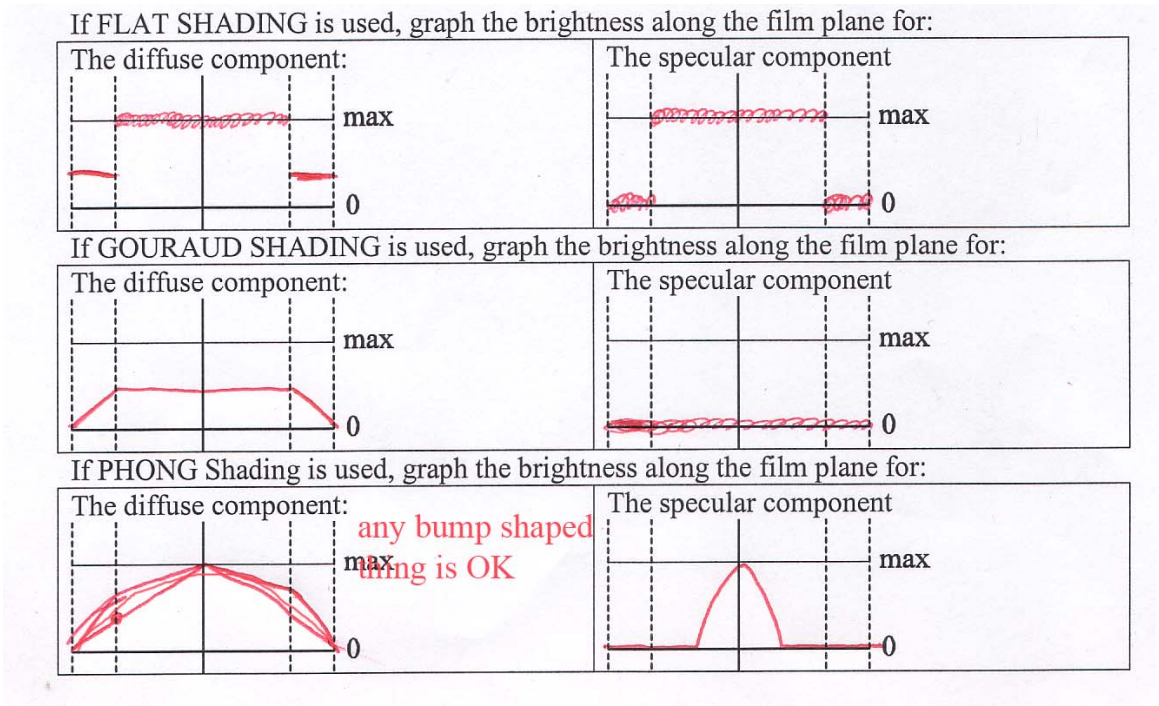
Graphs are drawn for the film plane, the 5 marking correspond to those here.

Normals are always computed for the point on the actual sphere.

Specular brightness along the X-axis for the actual sphere.



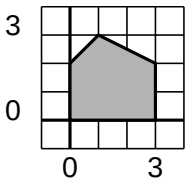
If the actual sphere (not the polygonal approximation) was lit perfectly, the specular component of the light along the film plane (in the X-direction for Z=0) is shown in the graph on the right. For the following questions, sketch a similar graph. (The important thing is the rough shape)



Question 8: Transformations (11 pts)

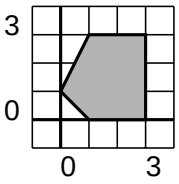
A simple 2D graphics toolkit has the following commands:

Translate x,y - multiply the current matrix by a translation
Rotate t - multiply the current matrix by a rotation of t degrees counter clockwise
Scale sx,sy - multiply the current matrix by a scale
DrawHouse - draws the following house (note: it is not symmetric)



Note: the transformation commands work like those in OpenGL

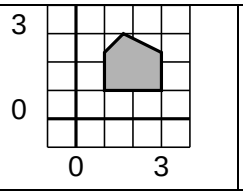
A) Here's an example program and its output. Give a program that does the same thing, but has the transformation commands in reverse order (e.g. fill in the parameters):



Translate 3,0	Rotate _____
Rotate 90	Translate _____
DrawHouse	DrawHouse

Rotate is 90
Translate is 0,-3

B) Give a program that produces the following output (your program should be 2 or 3 lines long). Give a second program that reverses the order of the transformations but does the same thing.



translate 1,1
scale 2/3, 2/3
drawhouse

scale 2/3,2/3
trans 3/2, 3/2
drawhouse