CMPS 160 F07
Introduction to Computer Graphics
Nov 2, 2007
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 handwritten notes
Calculators allowed
No computers
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.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} -2 \\ 1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$$

1. $a + 0b + 1c = -2$
2. $0a + 1b + 1c = 0$
3. $0a + 0b + 1c = -1$

$\Rightarrow c = -1, d = 1, a = -1$

$$\begin{bmatrix} -1 & 1 & -1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
The table below will show which of the following properties are preserved under translation, rotation, uniform scaling, and perspective projection transformations: position, lengths, angles, parallel, and relative sizes. **Fill in each cell in the table with either a Yes or No.**

*Position* – is the position of an arbitrary vertex the same after transformation?  
*Lengths* – is the length of an arbitrary line segment the same after transformation?  
*Angles* – is the angle between two arbitrary vectors the same after transformation?  
*Parallel* – are arbitrary parallel lines still parallel after transformation?  
*Relative sizes* – does the ratio relating the lengths of two arbitrary line segments remain unchanged after transformation?

<table>
<thead>
<tr>
<th></th>
<th>Positions</th>
<th>Lengths</th>
<th>Angles</th>
<th>Parallel</th>
<th>Relative Sizes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Translation</strong></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td><strong>Rotation</strong></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td><strong>Uniform Scaling</strong></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td><strong>Perspective</strong></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>
Suppose we set up a camera using gluLookAt(eye=[5,2,0], center=[5,1,0], up=[1,0,0]), and then render a scene with three spheres of radius 1, centered at [4,-3,4], [5,-8,-2], and [7,-1,1].

a. Can we use a near plane with distance zero and a far plane of "really far"? Why or why not?
   - Zero will not work, since there will be a divide-by-zero in the projection matrix.
   - Large values in general will cause precision issues in Z-buffers which have a finite number of bits for "Z"'s storage.

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)

\[ z \text{ is not related since camera is flat in } z. \]

\[ (0,2) \]

\[ (0,1) \text{ center} \]

\[ (0,1) \text{ center} \]

\[ (0,z-1) \]

\[ \text{near} - \text{clip} = 2 \text{ units} \]

\[ (4,3) \text{ center} \]

\[ (4,3) \text{ center} \]

\[ (5,8) \]

\[ (5,8) \]

\[ \text{far} = 11 \text{ units} \]
Suppose we have a minimal graphics language called SimpleGL which does not support a full range of commands. It has only the following:

- \( SGL_{-}TranslateX2() = \text{glTranslate}(2,0) \)
- \( SGL_{-}TranslateX3() = \text{glTranslate}(3,0) \)
- \( SGL_{-}TranslateY() = \text{glTranslate}(0,1) \)
- \( SGL_{-}PushMatrix() = \text{glPushMatrix}() \)
- \( SGL_{-}PopMatrix() = \text{glPopMatrix}() \)
- \( SGL_{-}DrawBox() = \{\text{glBegin()};\text{glVertex}(0,0);\text{glVertex}(0,1);\text{glVertex}(1,1);\text{glVertex}(1,0);\text{glEnd()}\} \)

Write out the shortest program, using this library, that will produce the image above.

```c
T_X()
T_Y()
Draw()
Push()
T_Y()
Draw()
Pop
T_X()
Draw()
```
The diagram below shows a BSP tree. The numbered darker lines are polygons, and the arrows show the orientation of the polygon.

![BSP Tree Diagram]

a) Suppose we look at this scene from the top side, instead of the left. Use the BSP tree to determine the order in which triangles should be rendered.

\[ 5b, 4, 1, 8, 2, 5a \]

\[ 4, 5b, 3, 1, 2, 5a \]

b) Construct a valid BSP tree starting at triangle 2, instead of triangle 3. (Starting from the left as shown.)
Describe the difference between Gouraud and Phong shading.

Gouraud - calculate light at vertices and interpolate color
Phong - interpolate normal and then calculate lighting at pixels

If we have a polygon that looks like

```
A = (0,6,0)  A = (-1,0,1)
B = (10,6,0)  B = (1,0,1)
C = (10,0,0)  C = (1,0,1)
D = (0,0,0)   D = (-1,0,1)
X = (5,3,0)
```

The polygon has material property parameters:

Ka = 1,  Kd = 0.2,  Ks = 0.3,  ns = 5

The scene world has an ambient light of RGB color (0.1,0.1,0.1)
Also there is a both a light and eye point located at (5,3,1) with RGB color (0.2,0.2,0.2)

a) Write the OpenGL lighting equation.

\[
I = KA + KD(N \cdot L) + KS(R, V)^n_s
\]

b) Using this equation, compute the color at point X using Phong shading. When doing calculations, remember to first normalize vectors.

```
A + B \over 2 = (0,0,1)
Handed norm of A = (-1,0,1)  B = (1,0,1)
Thus halfway is (0,0,1)
```

\[
I = \frac{ka}{|ka|} \cdot \frac{kd}{|kd|} \cdot \frac{(N,L)}{|N,L|} \cdot \frac{ks}{|ks|} \cdot \frac{ls}{|ls|} \cdot (0,0,1) \cdot \frac{R}{|R|} \cdot \frac{V}{|V|} \cdot \frac{n_s}{n^s}
\]