CMPS 160 – Midterm Exam #1

Name / Username (unix.ic)

Adam Smith [Answer Key]

CMPS 160 F06
Introduction to Computer Graphics
October 19, 2006

Midterm Exam #1

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
No notes
No calculators
No computers
No cooperation
No unsound reasoning

3. Find \(x\).

Mini OpenGL Man Pages:

```c
void glMatrixMode( GLenum mode ) – sets the current matrix mode
   GL_MODELVIEW – modeling and viewing transformation
   GL_PROJECTION – camera projection transformation
void glLoadIdentity( void ) – replaces the current matrix with the identity matrix
void glMultMatrixf( const GLfloat *m ) – multiplies the current matrix by the one specified
void glRotatef( GLfloat angle, GLfloat x, GLfloat y, GLfloat z ) –
   multiplies the current matrix by a rotation by angle degrees around the vector (x,y,z)
void glScalef( GLfloat x, GLfloat y, GLfloat z ) –
   multiplies the current matrix by a nonuniform scale along the x, y and z axes by the given (x,y,z)
void glPushMatrix( void ) – pushes the current matrix stack down by one, duplicating the current matrix
void glPopMatrix( void ) – pops the current matrix stack, replacing the current matrix
void glBegin( GLenum mode ) – delimits the vertices that define a group of primitives
   GL_POINTS – single vertices
   GL_LINES – each pair of vertices make separate line segments
   GL_LINE_STRIP – connects each vertex to the next to make a single line from several segments
void glEnd( void ) – delimits the vertices that define a group of primitives
```
For each call to `drawObject` below, identify the matrix that transforms the modeling coordinates resulting from the `drawObject` call to the final device coordinates used when rasterizing. Report this matrix as a product of matrices mentioned in the code (ex: "PSLM") assuming the scene’s vertices (represented as column vectors) will be placed to the right of your matrix for multiplication (this is the convention in OpenGL). Watch out, the code may do things that would be silly to put in a real program.

The function `glMultMatrix` is part of the OpenGL specification and works much like many of the other matrix manipulation functions. If $M_1$ was the state of the current matrix (as selected with `glMatrixMode`), then $M_1M_2$ is the state of the matrix after a call to `glMultMatrix` with $M_2$ as the argument.

```gl
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glMultMatrix(P);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
drawObject();
glMultMatrix(L);
drawObject();

glPushMatrix();
  glMultMatrix(O);
  glPushMatrix();
    glMultMatrix(S);
    drawObject();
  glPopMatrix();
  drawObject();
  glMultMatrix(M);
  drawObject();
  glPopMatrix();
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
drawObject();
```
We want to composite several partially transparent image layers to build a final scene. Let the layers be (listed from front to back) called A, B, C, D, E, F, G, H.

We are looking for the result of properly combining these layers using the Porter and Duff "over" operator. **Indicate which of the following are valid expressions for the compositing result with T or F for each expression.**

(A over (B over (C over (D over (E over (F over (G over H)))))

(((((((A over B) over C) over D) over E) over F) over G) over H)

(((((G over H) over F) over E) over D) over C) over B) over A

(((A over B) over (C over D)) over ((E over F) over (G over H)))

(((A over C) over (B over D)) over ((E over G) over (F over H)))

(((A over B) over (((C over D) over E) over (F over G) over H))))

Now we are looking for the result of properly combining these layers using the "plus" operator (also mentioned in P&D). **Indicate which of the following are valid expressions for the compositing result with T or F for each expression.**

(A plus (B plus (C plus (D plus (E plus (F plus (G plus H)))))

(((((((A plus B) plus C) plus D) plus E) plus F) plus G) plus H)

((((((G plus H) plus F) plus E) plus D) plus C) plus B) plus A

((A plus B) plus (C plus D)) plus ((E plus F) plus (G plus H))

((A plus C) plus (B plus D)) plus ((E plus G) plus (F plus H))

((A plus B) plus (((C plus D) plus E) plus (F plus G) plus H)))
A two-dimensional affine transformation $A$ maps the shape on the left to the shape on the right. Specify the $3 \times 3$ matrix that represents $A$ over homogenous coordinates using numbers.

- "Origin" shifts by $(-1, 0)$
- Unit vector in $y$ becomes new $x$ $(1, 0)$
- Unit vector in $x$ becomes diagonal $(1, 1)$

$$A = \begin{bmatrix} 1 & 1 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
The table below will show which of the following properties are preserved 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 Y or N.**

*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>Positions</th>
<th>Lengths</th>
<th>Angles</th>
<th>Parallel</th>
<th>Rel. Sizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translation</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Rotation</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Uni. Scaling</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Perspective</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

*only I does this!*
Consider the 3D object below. It is a solid cube with a cube-shaped notch removed from one corner. Hidden edges are indicated with dotted lines.

Sketch a wire frame top view of this object as it would be displayed using an orthogonal projection.

Sketch a wire frame top view of this object as it would be displayed using a perspective projection. The eye point should be somewhere along a line extending out along the normal of the top of the cube and should illustrate a non-zero field of view. Indicate hidden lines with dotted lines as in the diagram above. Add textual notes to clarify your sketch if you are not confident in your drawing-with-perspective skills.
In each of the following basic linear algebra questions, refer to the diagram below.

Note:
- A, B, C are arbitrary 3D points in the plane of the page
- \( u = B - A \)
- \( v = C - A \)

You may use vertical bars to represent taking the magnitude of a vector and the dot and cross operators where needed. You may mention any symbols used in the diagram but not make an argument related to how the points are arranged beyond the definitions above.

- What is the area of the triangle ABC? \( \frac{|u \times v|}{2} \)

- How would you form the normalized version of \( u \)? \( \frac{u}{|u|} \)

- If \( u \) dot \( v \) is zero, what can you say about the angle between \( u \) and \( v \)? \( \cos \theta = 1 \)

- If B and C could be anywhere on the plane of the page, what is the possible range of values \( u \) dot \( v \) could take on in terms of \(|u|\) and \(|v|\)?

- Now (only for this question) assume the diagram is drawn to scale. Does the vector \( u \) cross \( v \) point up out of the page or down into it? 
  \( u \times v \) points \underline{out of the page} by the right hand rule
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.

```c
void drawShape (void) {   
    glBegin (GL_LINE_STRIP);  
    glVertex3f (0.0, 0.0);    
    glVertex3f (0.0, 2.0);     
    glVertex3f (2.0, 2.0);     
    glVertex3f (2.0, 0.0);     
    glEnd();                  
}
```

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

```c
void drawCollection (void) {   
    glPushMatrix ();          
    glTranslatef (0.0, 1.0);  
    glRotatef (90, 0, 0, 1);  
    drawShape ();             
    glPopMatrix ();           
    glPushMatrix ();          
    glTranslatef (0.0, 1.0);  
    glRotatef (45, 0, 0, 1);  
    drawShape ();             
    glPopMatrix ();           
}
```
What is the 4x4 transformation matrix for translation by (a,b,c) for homogenous 3D coordinates?

\[
\begin{bmatrix}
1 & 0 & 0 & a \\
0 & 1 & 0 & b \\
0 & 0 & 1 & c \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

What is the 4x4 transformation matrix for reflection through the origin for homogenous 3D coordinates?

\[
\begin{bmatrix}
-1 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 \\
0 & 0 & -1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Is there a rotation matrix that will undo the transformation done by the previous reflection-through-origin matrix? If so, what is it?

*Not possible! An odd number of reflections changes the handedness of the space - so is not correctable by rotation.*

Consider the matrix that will transform a column vector of individual midterm scores into a new column vector of scores where the new score for each person has 50% weight on their individual score and 50% weight on the average score of everyone on their homework team.

For a class with \( N \) students this would be an \( N \)-by-\( N \) matrix. However, only one slice (a vector of \( N \) elements) of this matrix is needed if you only want to compute your own weighted score. **Is this slice a row or a column?**

\[
\begin{bmatrix}
\text{row}
\end{bmatrix}
\]

This slice will include a weight \( w_{\text{you}} \) corresponding to how much your individual score influences your weighted score, several occurrences of a \( w_{\text{them}} \) corresponding to how much another team member’s score will influence yours, and several zeros corresponding to how the scores of people outside of your team will affect your weighted score. **Supposing you had \( k \) people in your group (including yourself), what should \( w_{\text{you}} \) be and what should \( w_{\text{them}} \) be?** If you are unsure of your answer, back it up with an explanation of your guess.

\[
w_{\text{you}} = \frac{1}{2} \left( 1 \right) + \frac{1}{2} \left( \frac{1}{k} \right) = \frac{k+1}{2k}, \quad w_{\text{them}} = \frac{1}{2} \left( \frac{1}{k} \right) = \frac{1}{2k}
\]

For your own reference (not graded), compute these weights with \( k \) equal to your own team’s size.

\( k = 3 \)

\[
w_{\text{you}} = \frac{4}{6}, \quad w_{\text{them}} = \frac{1}{6}
\]