Exam Instructions (Read Carefully)

1. Sign the first page of the exam with your **Signature** in the space provided on the upper left immediately.

2. Continue reading the instructions, but **do not open the exam booklet** until you are told to do so by a proctor.

3. Print your **Name** and **Student Identification Number** on every page in the space provided at the top of each page **before** you start the exam.

4. Cheating is an academic offense. Your signature on the exam indicates that you understand and agree to the University’s policies regarding cheating on exams.

5. Please read the entire exam before answering any of the questions.

6. There are **five** questions on this exam, each worth the indicated number of marks. **Answer as many questions as you can.**

7. Write all of your answers on these pages. If you need more space, there is blank space at the end of the exam. Be sure to indicate when a question is continued, **both** on the page for that question and on the continuation page.

8. Interpret the exam questions as written. **No questions** will be answered by the proctor(s) during the exam period.

9. The exam is **closed book**. There are **no aids permitted**, except for a calculator.

10. You have **2.5 hours** in which to work. **Budget your time wisely.**

11. In the event of a **fire alarm** during the exam, enter the four-character code provided by the proctor(s) in the space on the upper right, then gather your belongings and exit the room, handing your exam to a proctor as you exit.

12. No one will be permitted to leave the exam room during the **last ten minutes** of the exam.

<table>
<thead>
<tr>
<th>Question</th>
<th>Mean</th>
<th>SD</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12.3</td>
<td>4.8</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>3.54</td>
<td>0.3</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>12.93</td>
<td>0.7</td>
<td>15</td>
</tr>
<tr>
<td>4(a)-(c)</td>
<td>5.8</td>
<td>0.5</td>
<td>6</td>
</tr>
<tr>
<td>4(d)-(e)</td>
<td>8.6</td>
<td>2.2</td>
<td>10</td>
</tr>
<tr>
<td>4(f)-(g)</td>
<td>12.0</td>
<td>2.3</td>
<td>14</td>
</tr>
<tr>
<td>5</td>
<td>10.0</td>
<td>5.5</td>
<td>20</td>
</tr>
<tr>
<td>6(a)-(i)</td>
<td>17.5</td>
<td>1.7</td>
<td>18</td>
</tr>
<tr>
<td>6(j)-(r)</td>
<td>15.7</td>
<td>3.0</td>
<td>18</td>
</tr>
<tr>
<td>6(s)-(aa)</td>
<td>16.2</td>
<td>2.9</td>
<td>18</td>
</tr>
<tr>
<td>Name &amp; ID</td>
<td>1.0</td>
<td>0.0</td>
<td>1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>115.4</strong></td>
<td><strong>16.7</strong></td>
<td><strong>150</strong></td>
</tr>
</tbody>
</table>
A rotation matrix is stored in a global array rotation_matrix[16], which is updated whenever mouse actions are taken, either by an object-centred rotation or a world-centred rotation. The mouse callback function determines the angle and axis of rotation based on mouse coordinates, and then calls one of the two functions below, depending upon which mouse buttons are depressed.

Both functions update the global array rotation_matrix so that it has the new rotation matrix and then they force the screen to be updated using the new matrix. Neither function makes any permanent change to the MODELVIEW matrix or matrix stack, which is the selected matrix mode. The following C code fragments might be used in either of the two functions.

A glGetDoublev (GL_MODELVIEW_MATRIX, rotation_matrix);
B glLoadIdentity ();
C glLoadMatrixd (rotation_matrix);
D glMultMatrixd (rotation_matrix);
E glPopMatrix ();
F glPushMatrix ();
G glRotatef (angle, 0.0, 0.0, 1.0);
H glRotatef (angle, 0.0, 1.0, 0.0);
I glRotatef (angle, 1.0, 0.0, 0.0);
J glRotatef (angle, A [0], A [1], A [2]);
K glTranslatef (A [0], A [1], A [2]);
L glutPostRedisplay ();

Fill in each box below with the letter corresponding to the proper code fragments to implement the object-centered and world-centered rotation computations. Not all boxes need to be filled in and not all code fragments need to be used. The order of the code fragments should be exactly the order in which the code fragments should appear in the two routines. The letter for the first code fragment in the first routine is already filled in for you.

void rotate_wrt_body (GLfloat angle, vector A) {
   // Note that ‘A’ is a vector in object coordinates!
   // Perform a body-centered rotation about rotation axis ‘A’
   // by angle ‘angle’.
   
   F C J A E L

}

void rotate_wrt_world (GLfloat angle, vector A) {
   // Note that ‘A’ is a vector in world coordinates!
   // Perform a world-centered rotation about rotation axis ‘A’
   // by angle ‘angle’.
   
   F B J D A E L

}
Question #2 [10 marks total]

This question tests your knowledge of how viewing parameters were set in the planetary visualization system for Assignment #3.

Recall that for the active viewer calculation in Assignment #3, we required that the limb of the eye planet be located at a constant angular distance of $\alpha = 0.4 \cdot fovy$ from the centre of the screen, where $fovy$ is the current field of view (in degrees) in the y-direction.

Suppose we want instead that the limb of the planet be located at a constant linear distance (in window coordinates) below the centre of the screen. That is, the limb of the planet is located $0.4 \cdot Hy$ pixels below the centre of the screen, where $Hy$ is the height of the screen (in pixels).

Give a formula for $\alpha$ (the angular distance) in terms of the quantities $fovy$ and $Hy$. You may find it useful to refer to the diagram given above.

Let $L$ be the distance from the eye point to the vertical line in the diagram. Standard trigonometry tells us:

$$\tan \frac{fovy}{2} = \frac{Hy}{L} \quad \text{and} \quad \tan \alpha = \frac{0.4 \cdot Hy}{L}$$

Dividing these two equations and solving for $\alpha$ gives us the answer.

$$\frac{\tan \alpha}{\tan \frac{fovy}{2}} = \frac{0.4 \cdot Hy}{Hy} \implies \tan \alpha = 0.8 \cdot \tan \frac{fovy}{2} \implies \alpha = \arctan(0.8 \cdot \tan \frac{fovy}{2})$$
Question #3 [15 marks total]
This question tests your knowledge of L-systems and the turtle graphics interpretation discussed in the textbook and in Assignment #4.

Consider the L-system above, with axiom Q, that is used throughout this question.

Q  -->  WF[+A]  
W  -->  Q  
A  -->  AF

(a) [10 marks] Starting with the axiom Q, show the results of the first five productions of the L-system by filling in the last five entries in the following table (the first entry is the axiom).

<table>
<thead>
<tr>
<th>Production 1</th>
<th>Production 2</th>
<th>Production 3</th>
<th>Production 4</th>
<th>Production 5</th>
</tr>
</thead>
</table>

(b) [5 marks] Using a turning angle of 60°, draw the 2D “turtle interpretation” of the string shown in Production 5 on the grid to the right.

Use the standard conventions for interpreting L-systems that were employed on the assignments (in particular, ignore any characters that do not have a turtle interpretation).

The initial configuration for the turtle is at the black dot, and the turtle is heading in the upward direction.

The mirror image (about the vertical axis) of this diagram is also OK.
Question #4 [30 marks total]

This question tests your knowledge of hierarchical display list structures as discussed in lecture, and how transformations are represented as matrices.

When the \texttt{Walk()} function discussed in lecture is called to process the root node of the DAG, multiple instances of the unit cube will be displayed at various locations on the screen, often with changes in the shape and orientation of the cube. Refer to this diagram for all parts of this question.

\begin{itemize}
\item \texttt{T0} is a \textbf{translation} that takes \((x, y, z)\) to \((x, y - 3, z)\).
\item \texttt{T1} is a \textbf{translation} that takes \((x, y, z)\) to \((x, y, z)\).
\item \texttt{T2} is a \textbf{translation} that takes \((x, y, z)\) to \((x, y + 5, z)\).
\item \texttt{T3} is a \textbf{translation} that takes \((x, y, z)\) to \((x - 7, y, z)\).
\item \texttt{T4} is a \textbf{translation} that takes \((x, y, z)\) to \((x - 2, y, z)\).
\item \texttt{T5} is a \textbf{translation} that takes \((x, y, z)\) to \((x + 11, y, z)\).
\item \texttt{R0} is a \textbf{rotation} of 90° about the \(y\)-axis.
\item \texttt{R1} is a \textbf{rotation} of 90° about the \(z\)-axis.
\item \texttt{S} is a \textbf{scaling} that \textbf{contracts} everything by a factor of three along the \(x\) axis, \textbf{expands} everything by a factor of seven along the \(y\) axis, and leaves things \textbf{unchanged} along the \(z\) axis.
\item \texttt{C} has the geometry for a \textbf{unit cube}, axis-aligned and centered at the origin, that is drawn with whatever the current transformation matrix is each time this node is visited.
\end{itemize}
(a) [2 marks] Write the 4 × 4 matrix representation for $T_0$.

\[
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & -3 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{pmatrix}
\]

(b) [2 marks] Write the 4 × 4 matrix representation for $R_1$.

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

(c) [2 marks] Write the 4 × 4 matrix representation for $S$.

\[
\begin{pmatrix}
\frac{1}{3} & 0 & 0 & 0 \\
0 & 7 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{pmatrix}
\]
(d) [5 marks] When the unit cube is subjected to a non-uniform scaling it turns into a parallelepiped. Assume that the viewing transformation is just the $4 \times 4$ identity matrix with no perspective and that the DAG on page 5 is traversed using the `Walk()` function described in lecture with $T_0$ as its argument (the root of the DAG). A sequence of parallelepipeds will be drawn on the screen, each with a different transformation matrix that results from the traversal. Show the contents of the matrix stack when the first parallelepiped is being drawn, by writing the symbolic expression (various products of the matrices stored in the DAG) for each matrix in the stack. Stack entries that are not active should be left blank. Explicitly indicate the top of the stack by labeling that entry “top”.

(1)

(2) top: $I \cdot T_0 \cdot T_4 \cdot R_1 \cdot S$

(3) $I \cdot T_0 \cdot T_4 \cdot R_1$

(4) $I \cdot T_0 \cdot T_4$

(5) $I \cdot T_0$

(6) bottom: $I$ (the identity matrix)

(e) [5 marks] Write the numeric value of the $4 \times 4$ matrix that is on the top of the stack in part (d) of this question (i.e., evaluate the product of the various matrices using the actual numeric values for all entries).

\[
\begin{bmatrix}
0 & -7 & 0 & -2 \\
\frac{1}{3} & 0 & 0 & -3 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
= T_0 \cdot T_4 \cdot R_1 \cdot S =
\]

\[
\begin{bmatrix}
1 & 0 & 0 & -2 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & -3 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
0 & -1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
= 
\begin{bmatrix}
\frac{1}{3} & 0 & 0 & 0 \\
0 & 7 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\]
If we assume that the function `Walk()` is invoked with \( T_0 \) as its argument, there will be a number of 3D parallellipipeds (possibly scaled and rotated cubes) draw at various places on the screen, each appearing as a 2D rectangle because an orthographic projection is being used. The center of each rectangle will be determined by how the origin \((0, 0, 0, 1)^T\) is translated by the matrix, and the width and height will be determined by how the unit cube is stretched along the \( x \)- and \( y \)-axes of the screen, respectively.

(f) **[2 marks]** How many rectangles are drawn on the screen?

six rectangles appear on the screen

(g) **[12 marks]** For each of the 2D rectangles that is drawn, describe the \((x, y)\) position of its center on the screen, its width (size along the \( x \)-axis of the screen), and its height (size along the \( y \)-axis of the screen). Enter the information for each rectangle in the order in which they are drawn by the `Walk()` function. Unused entries in the table should be left blank.

<table>
<thead>
<tr>
<th>Object</th>
<th>X-center</th>
<th>Y-center</th>
<th>Width</th>
<th>Height</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-2</td>
<td>-3</td>
<td>7</td>
<td>( \frac{1}{3} )</td>
<td>( T_0-T_4-R_1-S-C )</td>
</tr>
<tr>
<td>2</td>
<td>11</td>
<td>-3</td>
<td>1</td>
<td>1</td>
<td>( T_0-T_4-T_5-C )</td>
</tr>
<tr>
<td>3</td>
<td>-7</td>
<td>0</td>
<td>1</td>
<td>( \frac{1}{3} )</td>
<td>( T_0-T_1-T_3-R_0-R_1-S-C )</td>
</tr>
<tr>
<td>4</td>
<td>-2</td>
<td>0</td>
<td>7</td>
<td>( \frac{1}{3} )</td>
<td>( T_0-T_1-T_3-T_4-R_1-S-C )</td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>( T_0-T_1-T_3-T_4-T_5-C )</td>
</tr>
<tr>
<td>6</td>
<td>11</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>( T_0-T_1-T_2-T_5-C )</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Question #5 [20 marks]
This question tests your knowledge of the projection matrices and the perspective transformation used in OpenGL.

If $ax + by + cz + d = 0$ is the equation for a plane $\pi$, then the matrix

$$
M = \begin{pmatrix}
    d & 0 & 0 & 0 \\
    0 & d & 0 & 0 \\
    0 & 0 & d & 0 \\
    -a & -b & -c & 0
\end{pmatrix}
$$

represents the transformation that projects a 3D point in homogeneous coordinates onto the 2D plane through a center of projection at the origin if we follow the multiplication of the matrix with a point by the standard conversion from homogeneous to non-homogeneous coordinates. You may assume that this statement is correct throughout this question.

(a) [5 marks] When $d = 0$ the matrix $M$ is highly singular. Explain why this makes sense. **Hint:** Give a physical interpretation of where the plane is located when $d = 0$.

If $d$ is zero, the plane passes through the origin. Any line of projection from a point not on the plane $\pi$ only intersects $\pi$ at the origin. The fact that the matrix is singular means that the homogeneous result for a point $(x,y,z,1)^T$ is the point $(0,0,0,−ax−by−cz)^T$. This reflects the fact that every point projects onto the origin.

(b) [5 marks] When $d$ is not zero, what is the result of applying the matrix $M$ to the origin followed by the standard conversion from homogeneous to non-homogeneous coordinates? Explain why this makes sense.

If $d$ is not zero, the plane does not pass through the origin, but the homogeneous result for the origin $(0,0,0,1)^T$ is the point $(0,0,0,0)^T$, which cannot be converted to a non-homogeneous point (or even to a direction). This reflects the fact that the origin does not uniquely project onto any single point of the plane $\pi$. 
(c) [5 marks] Suppose that there are two planes \( \pi_1 \) and \( \pi_2 \), determined respectively by the equations \( a_1x + b_1y + c_1z + d_1 = 0 \) and \( a_2x + b_2y + c_2z + d_2 = 0 \), with corresponding projection matrices

\[
M_1 = \begin{pmatrix} d_1 & 0 & 0 & 0 \\ 0 & d_1 & 0 & 0 \\ 0 & 0 & d_1 & 0 \\ -a_1 & -b_1 & -c_1 & 0 \end{pmatrix} \quad \text{and} \quad M_2 = \begin{pmatrix} d_2 & 0 & 0 & 0 \\ 0 & d_2 & 0 & 0 \\ 0 & 0 & d_2 & 0 \\ -a_2 & -b_2 & -c_2 & 0 \end{pmatrix}
\]

Prove that multiplying a point in homogeneous coordinates by the product matrix \( M_2 \cdot M_1 \) followed by a conversion from homogeneous to non-homogeneous coordinates correctly computes the position of a 3D point that is first projected onto \( \pi_1 \) and then onto \( \pi_2 \).

Multiplying a point by \( M_2 \cdot M_1 \) is equivalent to multiplying just by \( M_1 \), doing the conversion to non-homogeneous coordinates, doing a conversion back again to homogeneous coordinates using \( w = a_1x + b_1y + c_1z + d_1 \) instead of \( w = 1 \), and then multiplying by \( M_2 \) and then converting back to non-homogeneous coordinates. This works because we can choose any \( w \) that we like without changing the final result. Mathematically, we have the following:

\[
M_2 \cdot M_1 = \begin{pmatrix} d_i & 0 & 0 & 0 \\ 0 & d_i & 0 & 0 \\ 0 & 0 & d_i & 0 \\ -a_i & -b_i & -c_i & 0 \end{pmatrix} \begin{pmatrix} d_j & 0 & 0 & 0 \\ 0 & d_j & 0 & 0 \\ 0 & 0 & d_j & 0 \\ -a_j & -b_j & -c_j & 0 \end{pmatrix} = \begin{pmatrix} d_i d_j & 0 & 0 & 0 \\ 0 & d_i d_j & 0 & 0 \\ 0 & 0 & d_i d_j & 0 \\ -a_i d_j & -b_i d_j & -c_i d_j & 0 \end{pmatrix} = d_j \cdot M_i
\]

and thus

\[
\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = d_1 \cdot M_2 \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = d_1 \cdot M_2 \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}
\]

After conversion to non-homogeneous coordinates, the factor of \( d_1 \) simply vanishes and we get the same result as if we had only applied \( M_2 \).

(d) [5 marks] Explain why \( M_1 \) and \( M_2 \) do not in general commute.

The matrices \( M_1 \) and \( M_2 \) cannot commute because the result of projecting twice is a mapping of 3D space onto either the 2D plane \( \pi_1 \) or \( \pi_2 \), so unless the two planes are the same, the matrix products \( M_1 \cdot M_2 \) and \( M_2 \cdot M_1 \) have to be different because the resulting planes are different. Mathematically, we see from the general form of \( M_i M_j \) in part (c) that the two products are the same iff \( d_1 = d_2 = 0 \) or \( (a_1, b_1, c_1, d_1) = \alpha(a_2, b_2, c_2, d_2) \) for some \( \alpha \neq 0 \). In the first case both products are identically zero, and in the second case the two planes \( \pi_1 \) and \( \pi_2 \) are the same. Clearly \( d_j M_i \neq d_i M_j \) in general.
Question #6 [54 marks – 2 marks each]

This question tests your general knowledge of the concepts and terminology introduced in the course.

The following terms or people’s names are possible answers for the questions on subsequent pages. Use the number corresponding to a term or name below as an in the space provided answer if you think it is the best match for one of the concepts or terms on subsequent pages. Each term may be used once, more than once, or not at all.

(1) affective
(2) affine
(3) associative
(4) CIE chromaticity diagram
(5) CIE RGB
(6) CIE XYZ
(7) commutative
(8) concave
(9) cone
(10) convex
(11) DAG
(12) dominant
(13) dominatrix
(14) doping
(15) Fitts’s law
(16) graphics pipeline
(17) matching
(18) non-uniform
(19) non-uniform scaling
(20) outcodes
(21) persistence
(22) pseudodepth
(23) pseudopod
(24) purple line
(25) refresh rate
(26) rod
(27) rotation about x
(28) rotation about y
(29) rotation about z
(30) saturated
(31) screen window
(32) Snell’s law
(33) translation
(34) tree
(35) tristimulus
(36) uniform
(37) uniform scaling
(38) unsaturated
(39) update rate
(40) viewport
(41) window
(42) winged edge
For each statement below, write the number of the term listed on the previous page that best fits into the missing space marked by ********* in the sentence.

[2 marks each]

_2_  (a) A weighted sum of vectors whose weights sum to unity (1) is called affine.

_10_  (b) A weighted sum of vectors whose weights are all non-negative and sum to unity (1) is called convex.

_22_  (c) In a z-buffer algorithm, we use the pseudodepth as an approximation for the actual distance from the eye to a point on an object because this is easier to compute than the true distance, but it still gives a correct depth comparison.

_27_  (d) A $4 \times 4$ matrix
\[
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & a & -b & 0 \\
0 & b & a & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]
where $a^2 + b^2 = 1$ is a rotation about $x$ transformation.

_33_  (e) A $4 \times 4$ matrix
\[
\begin{pmatrix}
1 & 0 & 0 & a \\
0 & 1 & 0 & b \\
0 & 0 & 1 & c \\
0 & 0 & 0 & 1
\end{pmatrix}
\]
corresponds to a translation transformation.

_19_  (f) A $4 \times 4$ matrix
\[
\begin{pmatrix}
a & 0 & 0 & 0 \\
0 & b & 0 & 0 \\
0 & 0 & c & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]
corresponds to a non-uniform scaling transformation.

_7_  (g) Two matrices $A$ and $B$ are commutative if $AB = BA$.

*18*  (h) In general, two scaling matrices $A$ and $B$ satisfy the equation $AB = BA$ only if they are both non-uniform scaling matrices.

NOTE: Either 18 or 19 (non-uniform scaling) will be accepted as a correct answer for this question, as will any one of 36 or 37 (uniform scaling) or 2 (affine).

_20_  (i) The one-bit quantities that specify whether a point is “in” or “out” with respect to each of the boundaries of a clipping region are called outcodes.
[2 marks each]

_32_  (j) The index of refraction is used in **Snell’s law** to determine the path of light across the interface between two materials.

_9_  (k) The **cone** photoreceptors in the retinas of our eyes provide colour vision.

_26_  (l) The **rod** photoreceptors in the retinas of our eyes provide black & white vision.

_35_  (m) The **tristimulus** theory tells us that human colour vision can be modeled as a three-dimensional vector space.

_6_  (n) The 3D **CIE XYZ** colour space is derived from the colour space obtained from experimental data by choosing a new set of axes so that all spectral colours have non-negative coordinates.

_12_  (o) The wavelength of the pure colour associated with a particular spectral distribution is called the **dominant** wavelength with respect to an (arbitrary) white point.

_24_  (p) When the pure color associated with a particular spectral distribution falls on the **purple line** we use the wavelength of the complementary pure colour to specify hue.

_30_  (q) Another colour term that means the same as “pure” is **saturated**.

_17_  (r) The CIE colour system was determined in the 1920’s and 1930’s by a series of colour **matching** experiments conducted with human observers.
[2 marks each]

_39_  (s) The **update rate** is the number of times per second (usually 30-72 Hz, but a minimum of 10-15 Hz) that the image being displayed on a screen must be modified to produce the illusion of continuous, smooth motion.

_25_  (t) The **refresh rate** is the number of times per second (usually 60-72 Hz, but a minimum of 40 Hz – the “critical fusion frequency”) that an image must be displayed on a screen in order to maintain the illusion of a continuous image (“persistence of vision”).

_14_  (u) Normally the chemical compounds used as CRT phosphors are selected for their colour and other visible attributes, but when a lightpen is used with the CRT **doping** is sometimes added to the phosphor so the circuitry in the lightpen can better detect when it sees light produced by the CRT.

_21_  (v) A measure of how long a phosphor glows after excitation, the **persistence** of a phosphor, is the length of time until the light emitted drops to 10% of its original intensity.

_16_  (w) The **graphics pipeline** is the entire process of creating and perceiving an image on a CRT or other display device starting with a software application program and ending with the human visual system and brain.

_41_  (x) A **window** is the subset of the World Coordinate System that is eventually rendered as an image on the face of the CRT.

_40_  (y) A **viewport** is the portion of the virtual frame buffer provided by OpenGL into which a graphics program renders pixels for an image.

_31_  (z) A **screen window** is the portion of the face of the CRT that displays an image stored in the physical frame buffer.

_42_  (aa) A data structure that stores a record for each vertex, face, and edge, where the edge record is the most important, is called the **winged edge** data structure. The edge record has pointers to the two vertex endpoints $v_1$ and $v_2$ incident with the edge, pointers to the two faces $f_1$ and $f_2$ on either side of the edge, and pointers to the clockwise and counter-clockwise predecessor and the successor edges incident with the edge on the boundaries of each of $f_1$ and $f_2$. 