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 three 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 proctors during the exam period.
9. The exam is closed book. There are no aids permitted except for a calculator.
10. You have 45 minutes 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.
Question #1 [30 marks total]

This question tests your knowledge of picking. The following diagram was used in Midterm #1 for describing the various OpenGL function calls that are required to render the colour cube and other objects in the 3D scene for Assignment #1. The nodes in the diagram correspond to one or more OpenGL function calls, as indicated.

“N1” corresponds to OpenGL calls that deal only with setting the parameters of the synthetic camera.

“N2” corresponds to the VTrackball modeling transformation.

“N3” corresponds to CIE-to-RGB coordinate transformation.

“N4” corresponds to a projection modeling transformation.

“N5” corresponds to OpenGL calls that are specific to drawing the Monitor RGB colour cube.

“N6” corresponds to OpenGL calls that are specific to drawing the labeled CIE XYZ axes.

“N7” corresponds to OpenGL calls that are specific to drawing the 3D CIE tristimulus curve.
For each node $N_i$ in the diagram on the previous page, a call is made to `glPushName(i)` when that node is rendered. This is done before any other OpenGL functions are called for that node. Thus the node $N_i$ has the integer name $i$ associated with it in the name stack if a hit occurs during picking.

After initializing the name stack, the scene is drawn in select (picking) mode with back-face culling turned on. This means that faces of the cube that cannot be seen are not drawn at all and thus they cannot be picked.

After manipulating the cube with the VTrackball, there is only one face of the Monitor RGB cube that has a black corner visible (the other two faces that have a black corner are facing away from the viewer and thus are not drawn). Using the mouse, the black corner is picked.

As many students found out when they did Assignment #1, one of the points on the 3D CIE tristimulus curve is almost at the black point in the CIE XYZ coordinate system, so that point will also be picked when the mouse button is clicked on the black corner.

Assuming all of the above, and referring to the diagram on the previous page, write your answers for the following four parts of this question in the spaces provided (one of the parts is on the next page of the exam).

(a) **[5 marks]** What other points will be picked at the same time that the black point and the really faint point on the tristimulus curve are picked?

The XYZ axes will be picked because they include the origin. Other points might be picked if they line up correctly but this would depend on how the VTrackball leaves things. As stated, it would be difficult to pick other objects unless the picking region were unrealistically large.

(b) **[5 marks]** How many records will there be in the selection buffer after these hits if the selection buffer is allocated to be 1000 ints in size?

The buffer is large enough to hold all of the hit records. Although there may be many objects that are hit, only those with unique names will be recorded as hits. Assuming that there are only hits associated with the RGB cube, the CIE horseshoe, and the XYZ axes (as discussed above), there will be exactly three hit records in the selection buffer because there are three distinct sets of names for these objects. Note that the 2D CIE horseshoe is not picked because the projection is entirely in the $X+Y+Z=1$ plane, which does not include the origin.

(c) **[5 marks]** How would your answer to Part (b) change if the selection buffer had been allocated to be only 10 ints in size?

Each hit record has a minimum of three ints associated with it, plus however many names it contains. Since every hit has at least one name (in fact they range from three to four), the total size required exceeds what is allocated. So not all of the hit records will be in the selection buffer, even though the value returned by `glRenderMode()` will give the correct hit count. There will be only one complete hit record in this case, with one partial record as well. The (first) top 10 items in the list will on the next page will be included.
(d) [15 marks] Draw the contents of the entire selection buffer after these hits. You may simply write "zmax" or "zmin" for any entry that is a z-value, but you should write the appropriate integers for all of the other entries. The first entry in the selection buffer should be at the top (or left) in your diagram.

There are three records in the selection buffer (as noted in Part (b) of this question). The records are listed here in the order in which they would be generated assuming that the DAG is traversed as discussed in lecture. Any other ordering of the records is an acceptable answer.

Within each hit record there is a count (indicated by * in the diagram on the left), followed by zmin and zmax information, and then a list with exactly "count" names. The list of names is ordered least recent first, so that the "1" at the bottom of the name stack is the first name in every one of the lists.

The first record is the result of traversing N1, N2 and finally N5 (in that order). Similarly, the second record is N1, N2, N3 and N6, and the third record is N1, N2, N3 and N7. There is no record that results from N1, N2, N3, N4 and N7 because that is the 2D CIE horseshoe, which does not get picked.

Note: It was assumed that as the DAG was traversed, there would be calls to glPopName() after the sub-DAG at each node is traversed (just as one would pop transformation matrices from the transform stack). But this was not stated in the question. For this reason there are alternate answers (also correct) that would take into account no popping of the name stack.

3* zmin zmax 1 2 5
5* zmin zmax 1 2 5 3 6
6* zmin zmax 1 2 5 3 6 7
Question #2 [30 marks total]
This question tests your knowledge of transformations and their matrix representations.

(a) [20 marks] The following table lists ten different sets of transformations in its first column. For each set of transformations, indicate by marking “Yes” or “No” in the second column whether the set of transformations is commutative and similarly in the third column whether the set of transformations forms a group under composition (concatenation or multiplication).

All transformations are 3D transformations unless otherwise specified.

+1 for each correct answer
–1 for each incorrect answer
Don’t guess about answers!

<table>
<thead>
<tr>
<th>Set of Transformations</th>
<th>Commutative?</th>
<th>Forms a Group?</th>
</tr>
</thead>
<tbody>
<tr>
<td>affine transformations</td>
<td>No</td>
<td>No or Yes</td>
</tr>
<tr>
<td>invertible transformations</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>projections</td>
<td>No No</td>
<td></td>
</tr>
<tr>
<td>rigid body transformations</td>
<td>No No</td>
<td>Yes</td>
</tr>
<tr>
<td>translations</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>2D rotations</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>3D rotations</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>uniform scaling by factors &gt; zero</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>non-uniform scaling by factors &lt; zero</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>2D shears along the x direction</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Note that the text does not give a clear definition for "affine". If it includes scaling by zero, then some affine transformations do not have inverses and thus the set of affine transformations does not form a group. If scaling by zero is not permitted (the normal definition of affine), there are always inverses.
(b) \([5 \text{ marks}]\) Write all of the entries for the \(4 \times 4\) matrix that corresponds to the \textbf{translation} that moves the point \((1, 1, 1)\) to \((3, -2, 5)\).

The parameters are \(\Delta x = (3) - (1) = 2, \Delta y = (-2) - (1) = -3,\) and \(\Delta z = (5) - (1) = 4.\) The standard translation matrix is

\[
\begin{pmatrix}
1 & 0 & 0 & \Delta x \\
0 & 1 & 0 & \Delta y \\
0 & 0 & 1 & \Delta z \\
0 & 0 & 0 & 1
\end{pmatrix}
= \begin{pmatrix}
1 & 0 & 0 & 2 \\
0 & 1 & 0 & -3 \\
0 & 0 & 1 & 4 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]

(b) \([5 \text{ marks}]\) Write all of the entries for the \(4 \times 4\) matrix that corresponds to a \textbf{projection through the origin} onto the plane \(4x - 6y + 8z = 2.\)

The plane equation is \(4x - 6y + 8z - 2 = 0\) if we put it into the standard form \(Ax + By + Cz + D = 0.\) Thus \(A = 4, \ B = -6, \ C = 8\) and \(D = -2.\) We could orient and normalize the equation, but this is not necessary. The projection matrix, as discussed in lecture, is computed by using these values in the general form for a projection through the origin onto a plane.

\[
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
-A/D & -B/D & -C/D & 0
\end{pmatrix}
= \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
-(4)/(-2) & -(6)/(-2) & -(8)/(-2) & 0
\end{pmatrix}
= \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
2 & -3 & 4 & 0
\end{pmatrix}
\]
Question #3 (40 marks – 2 marks each)

Explain the following terms as they have been used so far in our course on computer graphics. If the term is an acronym, explain for what it is an acronym and its meaning, usage, or how it functions (i.e., do not just provide the full name, but also explain what the acronym identifies). Where appropriate, include explanatory diagrams or formulae in your answers.

(a) gamma correction

A technique for compensating for various non-linearities in the graphics pipeline, usually related to how monitors convert the RGB voltages to light intensities, but sometimes related to other steps in the graphics pipeline where intensities are calculated.

(b) glBeginList()

An OpenGL function that causes display primitives to be retained in a compiled display list for future use by glEndList(). Optionally, the primitives can be executed immediately in addition to being compiled.

(c) glEndList()

An OpenGL function that causes the top-most (most recent) name to be removed from the name stack.

(d) glEndList()

An OpenGL function that switches the rendering mode between "rendering" and "picking") and also returns the number of hits since it was last called.

(e) homogenizing

The conversion from non-homogeneous to homogeneous coordinates by the addition of a w coordinate (often, but not always, a one).

(f) left-handed coordinate system

An orthogonal Cartesian coordinate system in which the positive z-axis goes into the screen, assuming the positive x-axis goes to the left and the positive y-axis goes up.
(g) **linear transformation**

A transformation which has matrix representation for non-homogeneous coordinates (or, equivalently, one for which the identity \( T(au + \beta v) = aT(u) + \beta T(v) \) always holds for any scalars \( a \) and \( \beta \) and any vectors \( u \) and \( v \)).

(h) **matrix transpose**

The transpose of the matrix \( M = (m_{ij}) \) is the matrix \( M^T = (m_{ji}) \).

(i) **name stack**

In OpenGL the list of pick names is recorded in a stack (most recent on top) and the entire contents of the stack are returned with each hit record during picking.

(j) **near clipping plane**

Objects (or parts of objects) closer than the near clipping plane are not rendered. Usually the near clipping plane is set to be closer than all objects in the scene, but sometimes it is used to clip away objects that are obscuring other objects of interest.

(k) **normalized plane equation**

The plane equation \( Ax + By + Cz + D = 0 \) is normalized iff \( A^2 + B^2 + C^2 = 1 \).

(l) **orthographic projection**

A parallel projection (all lines of projection are parallel to each other) in which lines of projection are also parallel to the normal to the projection plane.

(m) **Phong illumination model**

An approximation for specular reflection that uses a \( \cos^n \theta \) term in addition to ambient and diffuse (Lambertian) reflection. The choice of \( n \) determines how specular a surface will look.
(n) point at infinity

In homogeneous coordinates, any point of the form \((x,y,z,0)\) where at least one of \(x\), \(y\), or \(z\) must be non-zero. These correspond to vectors or directions in non-homogeneous coordinates.

(o) quadric surface  

\[ \text{[give the general formula]} \]

An implicit surface whose equation has the form:

\[
F(x, y, z) = ax^2 + by^2 + cz^2 + 2dxy + 2eyz + 2fzx + 2gx + 2hy + 2jz + k = 0
\]

Note: The specific letters used for coefficients are not important, but all of the terms must be present.

(p) special orthogonal matrix

A matrix whose inverse is its transpose (or having the properties that make this true).

(q) Turner Whitted

Generally credited with creating the first complete ray tracing program.

(r) \(w\)-coordinate

The fourth "homogeneous" coordinate.

(s) yon plane

Another name for the far clipping plane. Objects (or parts of objects) farther away than this are not rendered.

(t) z-buffer quantization error

Effects such as farther objects over-writing nearer objects in the z-buffer due to the finite precision of the z-depth. This is particular noticeable when the ration of the near and far clipping distances is very large.
(extra space to continue work)