Notes on midterm exam

This will be a 75 minute written test covering everything we’ve done up to but not including texture mapping. This is roughly sections 1.7 and 1.8, chapters 2 through 5 (excluding section 3.14), and section 6.11. Some topics and review questions are listed below. The written homework from the beginning of the course provides some good examples for exam questions on transformations. Some other ideas for review or exam questions are shown in italics.

You may need to write short snippets of code or shaders, but you do not have to memorize specific OpenGL functions (though you should have a basic idea what they do).

The graphics pipeline:

(Model -> )
- Vertex processing -> Primitive assembly -> Rasterization -> Fragment processing
  (-> framebuffer)
- Behavior of GL_LINES, GL_TRIANGLE_FAN, etc

Role of shaders in the pipeline
- Could you write a vertex shader that uses two adjacent vertices to calculate a normal vector? Explain briefly.

Linear interpolation
Basic color representation
Clip coordinates, clipping volume, depth testing, left handedness
- Which point is closer to the viewer in clip space, (0.0, 0.0, -0.5) or (0.0, 0.0, 0.5)?
- Why does clipping potentially create new polygons?

Vertex attributes, “in” and “out” variables
- Write a fragment shader that colors each pixel with a greyscale value obtained from the average of the red, green, and blue values of an in variable called fColor.
- Write a complete shader program (vertex shader and fragment shader) that will color pixels according to their depth in clip space. Points that are closest to the viewer are unaffected, and points farthest from the viewer should be darkened by 50%. Assume that the vertex position and color are “in” variables in the vertex shader and that the position is in clip coordinates.

OpenGL primitives
- Given a set of vertices, show what is rendered using glDrawArrays(GL_LINES, ...), glDrawArrays(GL_TRIANGLE_FAN, ...), etc.

Indexed rendering

Linear transformations as matrices M = [f(e1)  f(e2)  f(e3)], where f is the transformation and e1, e2, e3 are basis vectors. (This is a much simpler way to look at it than the author’s derivation...)

Using homogeneous coordinates to represent translations
Affine transformation = linear transformation followed by a translation
   - Prove that the 4d matrix for an affine transformation always has (0 0 0 w) in bottom row.

2D rotation, translation, scaling
   - Given a 2D triangle with vertices (0, 0), (2, 0), (2, 1), sketch the triangle after
     a) translation by (4, 5) followed by rotation by 90 degrees
     b) rotation by 90 degrees followed by translation by (4, 5)
     c) translation by (4, 5) followed by scaling by 2 in the y direction
     (show the approximate new coordinates in each case, e.g., on graph paper)
   - For each of a, b, c write down the transformation matrix

Frame = basis vectors, plus an origin
Frames used in OpenGL: model, world, eye/view, clip, NDC, window/viewport
Points vs vectors in homogeneous coords
Standard matrices: Translate(), RotateX(), RotateY(), RotateZ(), Scale(), also shearing
   - Suppose an object is centered at (x, y, z) and we want to rotate it through angle r about
     its z axis. Write down the necessary matrix in terms of the standard matrices (e.g.
     Translate(), etc.) (Don’t multiply them out!)
   - Suppose an object is centered at (x, y, z) and we want to rotate it through angle r about
     an axis through its center, where the axis is oriented at angle theta from the positive y-
     axis and angle phi from the positive x-axis (as in spherical coordinates). Write down the
     necessary matrix in terms of the standard matrices.

Inverting a product of standard transformations
   - An affine transformation consists of scaling by (1, 2, 1), then a translation to (1, 2, 3).
     Give the matrix for this transformation, and calculate its inverse. (You don’t have to
     multiply out the result.)

Changing frames; transformation M vs. change of frames M^{-1}
   - Suppose you are working in 2D and your new frame has origin (5, 10) and that its x-
     axis is given by the vector [0, 1] and its y-axis is given by [-1, 0].
     a) What are the coordinates of the point (1, 2) with respect to this new frame?
        Draw a picture.
     b) Find the change-of-frame matrix
   - Write the C function LookAt(eye, at, up), assume you have basic functions such as
     RotateX(), Translate(), etc.

Orthographic projections, view volume
   - Write the C function Ortho(left, right, bottom, top, near, far)
   - Why do we have to specify near and far clipping planes?

Perspective projection, basic matrix (as used for flat shadows)
   - How would you create a flat shadow of an object on the plane x = -5?

Perspective transformation in OpenGL
   - How does it differ from the basic perspective projection? Why?
- What's a frustum?
- What happens to the z-values?

Perspective division
- What is it? Where does it occur in the pipeline?

Roles of the model, view, and projection matrices
- Describe an example showing why we keep them separate

Normal vectors
How to calculate face normals from vertex coordinates

Lambert shading
- Why do we have to normalize L and N in our lighting calculation?

Blinn-Phong (ADS) lighting model
- If L, N are normalized and L points to the light and N is the normal vector, show that the reflection vector R is 2(L dot N)N - L. (Hint: Draw the vector R - L; since N bisects the angle between R and L, R - L is orthogonal to N so you can find it by projecting L on N.)
- Assume L, N, and R are as above, and V points to the view point, H is halfway between V and L, phi is the angle between R and V, and psi is the angle between H and N. Draw a picture. Show that phi is 2 times psi.
- Using N dot H instead of R dot V in the Phong model, do you need to adjust the specular exponent to be larger, or smaller, to get the same effect? (A sketch of the two cosine functions might help...)
- We normally do lighting calculations in eye coordinates. Could you do it just as well in world coordinates? Explain what might be easier/harder/different.

Directional vs point lighting
The normal matrix (M_{3x3}^{-1})^T (where M_{3x3} represents the upper-left 3x3 submatrix of M)
- Give a concrete example showing that a transformation M might not do the right thing to normal vectors.
- Under what conditions does M_{3x3} correctly transform normal vectors?

Gourad vs Phong shading
- Briefly explain the difference
- Why do we have to normalize after interpolating vectors? Give an example showing that interpolating between two vectors of length 1 does not necessarily give you a vector of length 1.

The viewport transformation
The z-buffer algorithm, z-fighting