CS175 Assignment 9
Projection and Texture Mapping

Due on Monday, Nov 24, 11:59pm

This is a written assignment that improves your understanding of projection and texture mapping. Also since it is really easy to give away answers, please email `lib175@fas.harvard.edu` when you have questions, instead of using Piazza. Moreover, we will only explain topics from the books/lectures or clarify questions. You really need to solve the problems on your own. Also, start early so you have time to think things over!

1. Ex. 10.1 from the book.
2. [**Hint**: For Problem 2, 3, and 4 below, it would be easier if you consider a simplified projection matrix of the form
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$
which should give you the same result]

   Ex. 11.3 from the book.
3. Ex. 11.2 from the book. How does this answer differ from that of Ex 11.3? Where in the graphics pipeline could this possibly make a difference?
4. Ex. 11.4 from the book.
5. Ex. 12.3 from the book. How does this answer differ from that of Ex. 11.4. Be precise in your answer, and be careful of the "half pixel" offset issues described in Section 12.3.
6. [**Hint**: read Section 12.3 carefully before answering this] Suppose we have square texture of 512 pixels by 512 pixels. Consider the "query point" in the texture with coordinates $[0.45, 0.63]^t$. These coordinates are represented in the canonical unit square domain with the lower left corner being $[0, 0]^t$ and right corner being $[1, 1]^t$. What is the coordinates (again in the canonical unit square domain) of the "pixel center" closest to this query point?
7. [**Hint**: read Chapter 13 carefully before answering this] We have learned the correct way of evaluating/interpolating values that are affine with respect to the eye coordinates (or equivalently: object coordinates or clip coordinates over a triangle). In fact, whenever you declare a varying variable in OpenGL shaders, that process is performed automatically for you.

Now suppose v is some value that **we wish** to be affine w.r.t normalized device coordinates (NDCs), (and hence also screen space coordinates, but not eye coordinates), i.e.,

$$v = \begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix}$$

How would you evaluate/interpolate this v in OpenGL? What varying variables would you use? Remember that any OpenGL varying variables will be affinely interpolated with respect to eye coordinates.

[**Hints**]: You should find some values, related but not identical to v, that are, in fact, affine w.r.t. eye coordinates. Some "fix up" calculation may be needed in the fragment shader.

Recall the relationship between NDC and eye coordinates is

$$\begin{bmatrix} x_n w_n \\ y_n w_n \\ w_n \end{bmatrix} = P' \begin{bmatrix} x_e \\ y_e \\ z_e \\ 1 \end{bmatrix}$$

where P' is the projection matrix with the third row removed. Note that for simplicity we are omitting the z component of the NDC here.

It also may be useful to recall that the constant 1 function is affine w.r.t. NDCs:

$$1 = \begin{bmatrix} d & e & f \end{bmatrix} \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix}$$