CMSI 371-01
COMPUTER GRAPHICS
Spring 2015

Assignment 0326b

Now to crack open some major flexibility through the power of vectors and matrices…

Outcomes

This assignment will affect your proficiency measures for outcomes *2a*, *2b*, *3a*, *3d*, and *4a–4f*.

With the inclusion of 3D transforms, outcome *2a* expands to a maximum proficiency of **+**. Outcomes *3a* and *3d* can now reach a **+** due to the amount of functionality involved.

Not for Submission

If you have access to the Angel textbook, read/scan Sections 4.1–4.13 (pages 135–205) for additional depth and detail.

The code you will write for this assignment can be patterned after the *vector* bazaar sample. Make sure that you are clear on what that does before going all Morpheus (or Neo?) on your code.

For Submission

Enter the Matrix

Design and implement a computer graphics matrix library with an accompanying suite of unit tests. As with `Shape`, no specific design is mandated. However, the following capabilities should be provided:

- A basic matrix object that initializes, by default, to the identity matrix
- Matrix multiplication
- 3D translation matrices
- 3D scale matrices
- 3D rotation matrices based on an arbitrary axis (i.e., refactor the sample code to fit your matrix object implementation)
- Orthographic projection matrices
- Perspective (frustum) projection matrices
- Conversion/convenience functions to prepare the matrix data for direct consumption by WebGL and GLSL

Test the Matrix

This item is sufficiently important to deserve its own section:

- A unit test suite (the *vector* example provides one based on QUnit)

The reason for its importance is the reality that the aforementioned matrix functionality is fairly cut-and-dried; it will not be hard to find implementations on the web and in the textbook. Thus, what makes *your* implementation different are the specific design choices that you will make for your library and *your test suite*. Use the test suite's coverage to demonstrate your understanding of matrix and transformations. Choose your cases, edge or otherwise, wisely.

Apply the Matrix

Armed with your newly-minted matrix library, return to Assignment 0326a to give your `Shape` objects one or more transform properties (typically the *instance transformation*: a composition of rotate, scale, and translate) and extend your scene drawing code to apply those transforms. Yes, you will need to touch the vertex shader. You might also need something similar to the 2D `canvas`'s `save` and `restore` functions. The key idea here is to have, after this portion is done, a complete "construction set" of sorts for creating, composing, and now *transforming* your scene objects in a manner that is limited solely by your imagination.

Project the Matrix

The second major capability afforded by your matrix library will now be the ability to break out of that 2×2×2 cube thanks to the availability of projection matrices. Do take advantage of them now, in your main scene drawing code.

How to Turn It In

As before, organize your work as a reusable JavaScript file with clearly separated test files.