Assignment 0403
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, 3d, 3e, and 4a–4f.
With the inclusion of 3D transforms, outcome 2a expands to a maximum proficiency of +. Outcomes 3d and 3e remain at a maximum of | because full coverage has not been reached yet in either outcome.

Not for Submission
If you have access to the Angel textbook, read Sections 3.1–3.12 (pages 116–180) 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, matrix4x4.js. Include:
- A basic Matrix4x4 object that initializes, by default, to the identity matrix
- A multiply function which multiplies two Matrix4x4 objects and returns the result (as a Matrix4x4 object, of course)
- A translate function which takes three parameters dx, dy, and dz, returning a Matrix4x4 object that accurately represents this transformation
- A scale function which takes three parameters sx, sy, and sz, returning a Matrix4x4 object that accurately represents this transformation
- The rotate function given in the sample code, but refactored to fit your Matrix4x4 object
- The ortho projection function given in the sample code, but, as with rotate, refactored to fit your Matrix4x4 object
- A frustum projection function based on the matrix derived from the course handout
- Conversion/convenience functions to prepare the matrix data for direct consumption by WebGL and GLSL
- A unit test suite based on QUnit (as seen in the vector example).

Add Transforms to Your Objects
Now that you have the power, give your 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.