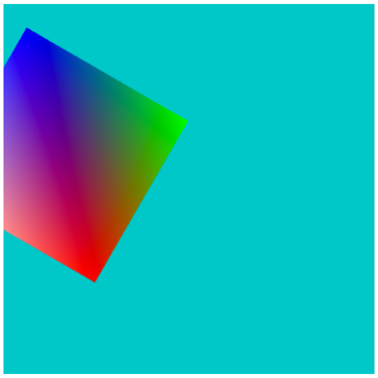
Computer Science 336
Fall 2015
Homework 2b

You can find the sample code for these problems in
http://web.cs.iastate.edu/~smkautz/cs336f15/examples/hw2/
Other transformation examples discussed in class are in
http://web.cs.iastate.edu/~smkautz/cs336f15/examples/transformations/

5. The example RotatingSquare.html (see the **`examples/transformations`** directory) gives several examples of updating a transformation during the animation loop. RotatingSquare2 is similar, but uses the shaders from GL_example2 to render a square with a different color at each vertex, and adds a simple keyboard handler. At this point, the animation loop does nothing and the keyboard handler just displays to the console the key that was pressed ('r', 'g', 'b', or 'w').

Your task is to add the following to RotatingSquare2. Initially, the square should rotate counterclockwise about its lower left corner, colored red, at a rate of two degrees per frame. When a key is pressed, the square should, starting in its **current** position, begin rotating about a different corner, depending on the key pressed: 'g' for the green corner, 'b' for the blue corner, 'w' for the white corner, or 'r' for the red corner.



You can see a flash movies of the end result here:
http://web.cs.iastate.edu/~smkautz/cs336f15/homework/hw2/hw2b.html

(*Hint*: To compute the current location of one of the corners you can take the current model transformation and apply it to the corresponding vector; for example, if you want the current location of the blue corner, use the vector $[0.5, 0.5, 0.0, 1.0]^T$. To do the multiplication, notice that there is a Vector4 class in the teal book matrix utilities, and the Matrix4 type has an operation multiplyVector4 that you can use.)

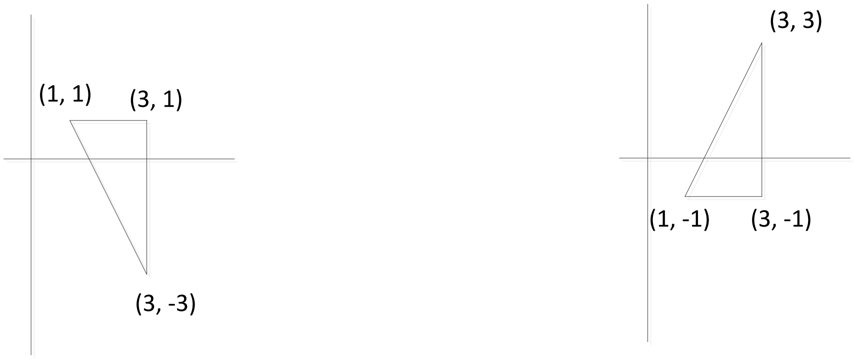
6. Reflection.js is like Transformations2.js, but adds a couple of features. There is a place to enter a slope and y-intercept for a line, and a checkbox for making the line appear on the canvas.

There is also a feature that draws the most recent position of the triangle in grey, to make it easier to see the effect of a transformation.

Also, there are two additional keyboard controls: "x" to perform a reflection about the x-axis, and "l" (lower case L) to perform a reflection about the line whose slope and intercept are provided. These operations are illustrated below. These are the two features you will implement.

The places where you need to modify code are marked "TODO". You might also want to read lines 240 - 241, where it gets the slope and intercept values from the text boxes and draws the indicated line by transforming the x-axis and drawing it in green.

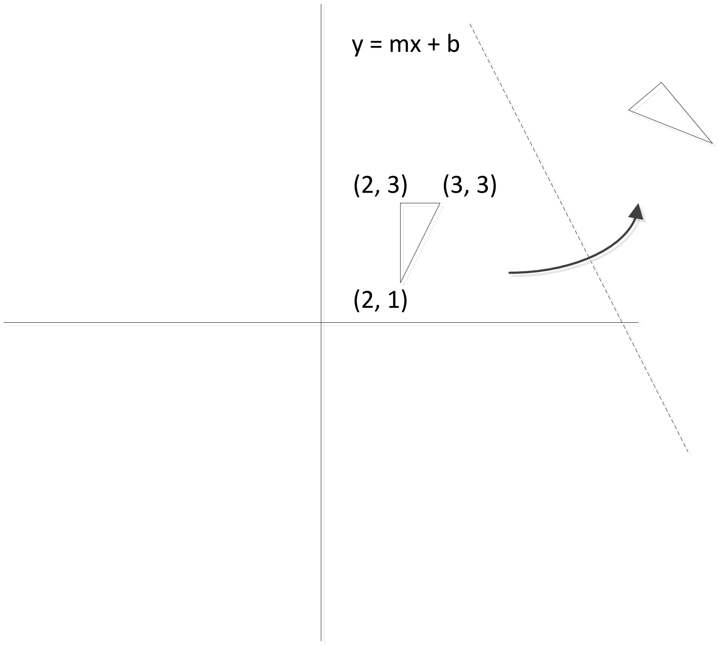
a) The matrix Scale(1, -1, 1) performs a *reflection* about the x-axis:

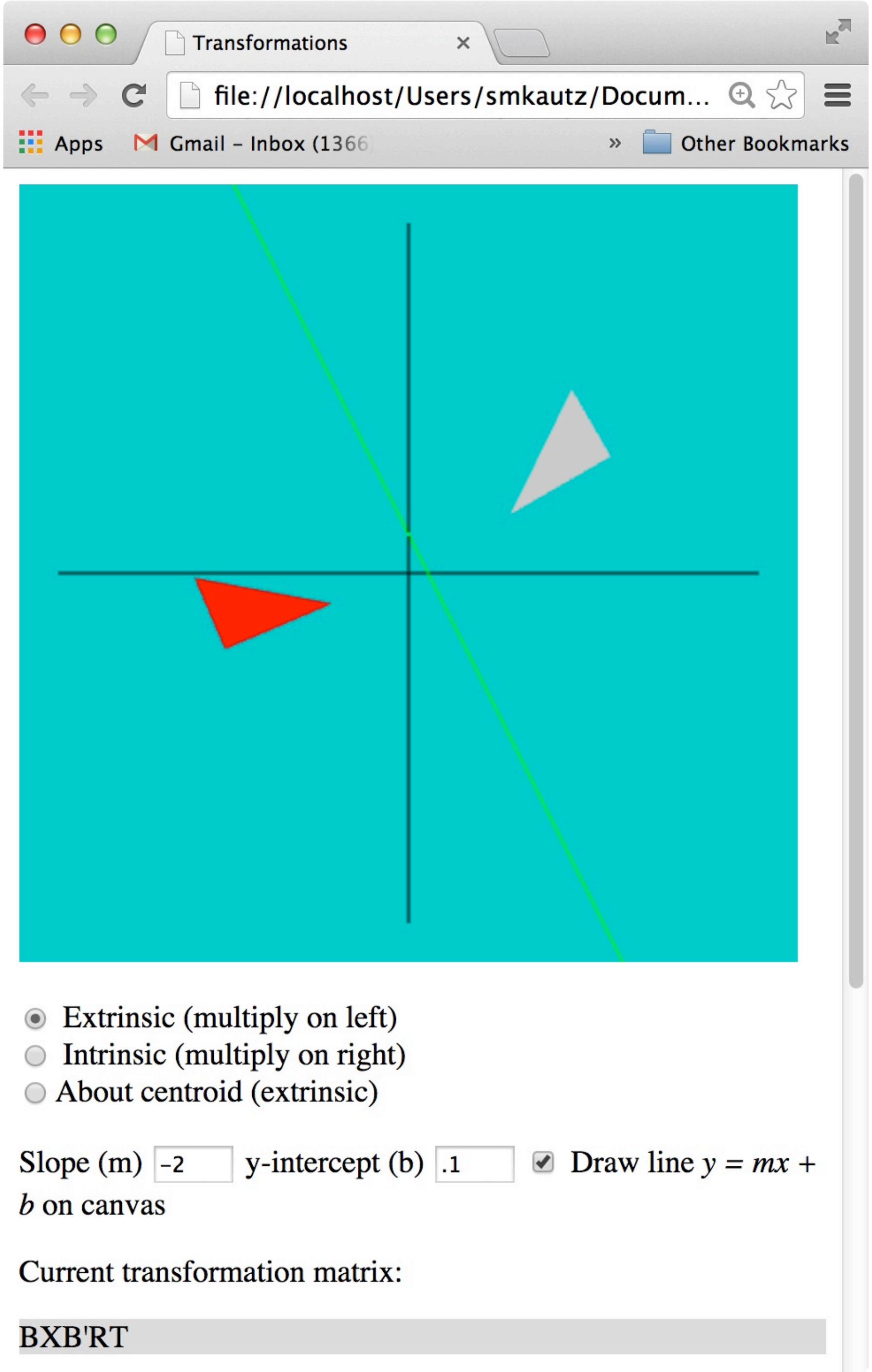


Add a line of code to Reflection.js in order to make this feature work (the "x" keyboard control).

b) Write a JavaScript function `makeReflection(slope,intercept)` that returns a Matrix4 representing a transformation in which all points are reflected across the line with given slope and y-intercept. Assume the z-coordinates are unchanged. (*Hint:* First find a transformation that transforms the x-axis into the line $y = mx + b$. Then use the matrix from (a).)

Incorporate your function into Reflection.js and make the reflection feature work (the "l" keyboard control). See illustration below and screenshot on the following page.


y = mx + b
(2, 3)
(3, 3)
(2, 1)



● Extrinsic (multiply on left)
○ Intrinsic (multiply on right)
○ About centroid (extrinsic)

Slope (m) -2 y-intercept (b) .1 ☑ Draw line $y = mx + b$ on canvas

Current transformation matrix:

BXB'RT