Lab 11

In this Lab we will explore the concatenation of transformations, and how the non-commutativity of matrix-matrix multiplication affects this in practice.

Run the code provided, `Lab11ConcatTransform.html`.

Note that a matrix R_z is defined in the vertex shader in `Lab11ConcatTransform.html` for a rotation of `uTheta` around the z-axis. Also note that constants `uTheta`, `uScale`, and `uTranslate` are defined in the application code `Lab11ConcatTransform.js`.

Note that the transformation matrices are defined in column-major order, which would be the transpose of how they are defined in class and in the textbook.

In the vertex shader, comment out:

```
gl_Position = aPosition;
```

And uncomment:

```
//gl_Position = rz * aPosition;
```

Next, create a translation matrix T_x for a translation `uTranslate` in the x direction, then in the vertex shader, comment out:

```
gl_Position = rz * aPosition;
```

And uncomment:

```
//gl_Position = Tx * aPosition;
```

Create a scaling matrix, S_x for a scaling `uScale` in the x direction, then in the vertex shader, comment out:

```
gl_Position = Tx * aPosition;
```

And uncomment:

```
//gl_Position = Sx * aPosition;
```

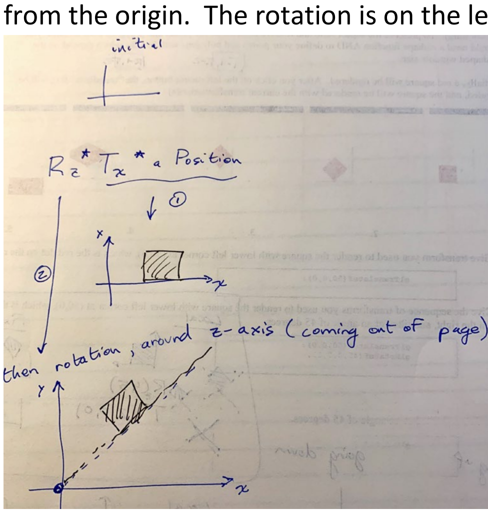
Finally, uncomment the remaining sequences of transforms, paying attention to the order. For instance, why does

(a) `gl_Position = rz * Tx * aPosition;`

Give a different output than

(b) `gl_Position = Tx * rz * aPosition;`

In short, in general, matrix multiplies do not commute (ie. order matters). In ***(a)*** , the translation is next to `aPosition`, so it "happens first," in this example the object is translated in the positive x direction from the origin. The rotation is on the left, so it happens next, as shown in the image:



In ***(b)***, the rotation is closest to `aPosition`, so it "happens first," followed by the translation.