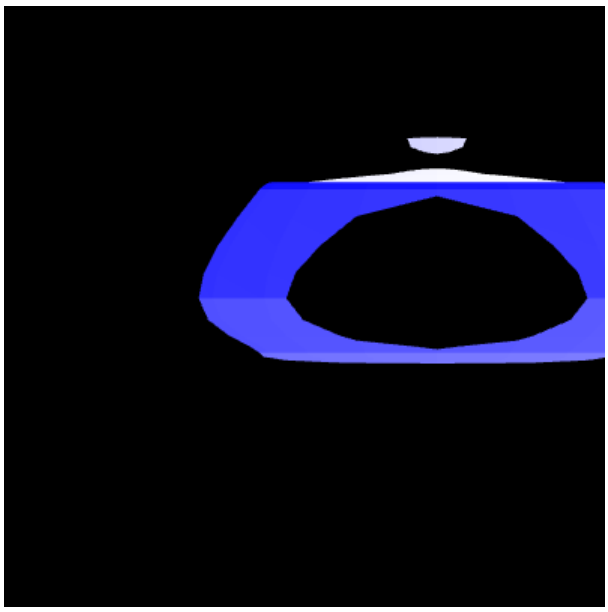
Part 1: Orthographic Matrices

While implementing the camera for your project, you were experimenting with the orthographic camera and managed to construct the following interesting scene (**scene 1**):



However, your laptop died right before you could take a note of the numbers you used. With the values to your projection matrix now gone, your scene reverted to its default view using its default projection, resulting in the image shown below:

Exercise 1. Write down the near, far, top, bottom, left, and right values you provided to the orthographic matrix to construct something visually similar to **scene 1**:

```
near = 1.53, far = 1.91, left = -1.07, right = 0.43, bottom = -1.2, top = 0.81
```

Exercise 2. Write down the orthographic matrix associated with the values calculated in Exercise 1. You may find it helpful to print out the projection matrix being used in the

orthographic_matrix.js demo when you provide these values:

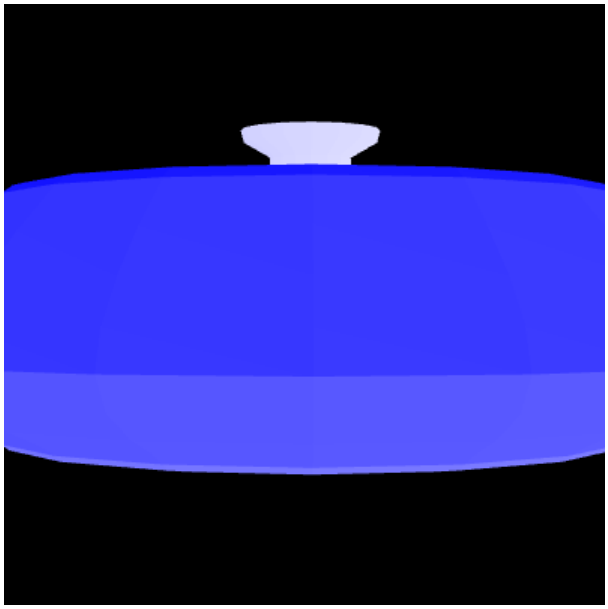
```
1.33,0.00,0.00,0.43

0.00,1.00,0.00,0.19

0.00,0.00,-5.26,-9.05

0.00,0.00,0.00,1.00
```

Exercise 3. If you were instead using an perspective projection, could you construct a similar scene? If not, explain why. If you can, write down the values that produce a scene that "looks similar":

```
Since a perpsective matrix can't adjust the left/right shape of the camera
independently, you can't really position this teapot without moving the teapot.
That being said, a fovy=60, aspect=.78, near=1.54, far=1.92 gets pretty close for
me
```

Part 2: Perspective Matrices

The next day, while implementing perspective for your projection matrix, you run your project to find that your scene looks like the following (**scene 2**):



Exercise 4. Estimate and write down what FOVY, Aspect Ratio, Near, and Far values could have been provided to achieve the (strange-looking result) in **scene 2**. Recall that trying out some numbers in the provided demo code may be helpful to estimate these numbers:

```
FOVY=47.16, Aspect=.5, Near=1.0, Far=4.0
```

After doing some calculations, you work out that a reasonable perspective matrix looks as follows:

```
[
1   0   0   0
0   1   0   0
0   0 -1.66 -2.66
0   0  -1   0
]
```

Which then should produce the (reasonable) scene:

When you look through the library implementation of the perspective, you recall that there is some code to convert from degrees to radians as follows::

```
fovy = Math.PI * fovy / 180 / 2
```

Exercise 5. Write down the FOVY, Aspect Ratio, Near, and Far values that you provide to achieve the exact matrix you calculated as correct (hint: check out the slides from lecture 10 for alpha/beta equations, and also consider printing out the demo matrix to see how the numbers change with FOVY/Aspect):

```
The default values of FOVY=90, Aspect=1, Near=1, and Far=4 produce this matrix
(I suspect this question should be changed, it's a bit confusing to just copy
these numbers in, oops!)
```