Programming

- In order to accomplish this assignment I attempted to modify the lib/obj-loader.js file as little as possible by post-processing the values it produces
- I modified readObjFile to return a Promise
    - this way I can use a callback function to initialize the model once it's finished loading
- I added a function initModel that does processing on the verticies and normals every time we load the object file
    - extracts object arrays (loads data to use used in array buffers)
    - sets the vertex normals
        - averages vertex normals as the average of the normals of the faces it is a component in
        - normalize vertex normal vectors
        - replace normals array that was generated with the face
    - normalizes vertices according to largest component of bounding box
    - calculates the center of mass of the object
    - stores transformation to move object to its center of mass in cmMatrix
    - assigns vertex buffers the data from the model
- I modified initScene() to run the initModel code after it's finshed downloading the model
- I modified drawScene()
    - disable the default spinning behavior
    - move the object to align it's center of mass to the origin (mvMatrix.multiply(cmMatrix))
    - to enable the rotation buttons, using the curRot matrix to store rotation
    - to enable the zoom in and zoom out buttons using translation equal to camZ

Questions

- let m = new Matrix4();
- 1.1 R(45, <0, 1, 0>) =
    - m.setRotate(45, 0,1,0).str()

```
[ 0.7071067690849304,  0, 0.7071067690849304,  0
 0,                    1, 0,                   0
 -0.7071067690849304, 0, 0.7071067690849304,  0
 0,                    0, 0,                   1 ]
```

- 1.2
    - theta = arccos((<1,1,0> • <0,1,0>) / (√(1^2 + 1^2 + 0^2) * √(0^2 + 1^2 + 0^2)))
    - theta = 45 degrees
    - axis = <1, 1, 0> x <0, 1, 0> = <0, 0, 1>
    - R(theta, axis) =
    - m.setRotate(45, 0,0,1).str()

```
[ 0.707…, -0.707…, 0,   0
  0.707…, 0.707…,  0,   0
  0,      0,       1,   0
  0,      0,       0,   1 ]
```

- you can multiply this and the <1,1,0> to get the resulting vector <0, 1, 0>

- 1.3 R(45, <1, 1, 0>)

```
[0.8535534143447876, 0.1464466154575348, 0.5, 0
 0.1464466154575348, 0.8535534143447876, -0.5, 0
 -0.5, 0.5, 0.7071067690849304, 0
 0, 0, 0, 1]
```

- not sure what problem means by "in terms of", if it means to applying them in order then the resulting matrix is
  - `m.setIdentity().rotate(45, 0,1,0).rotate(45, 0,0,1).rotate(45, 1,1,0).str()`

```
[-6.172263589121485e-9, 6.172263589121485e-9, 0.9999999403953552, 0
 0.7071067690849304, 0.7071067690849304, 0, 0
 -0.7071067690849304, 0.7071067690849304, 1.2688051498344066e-8, 0
 0, 0, 0, 1]
```

- 1.4
  - rotation of <1, 0, 0> to <-1,-2,-3>
  - translation from <0,0,0> to <2,4,6>
- 1.5,
- 1.6. (1,2,3)
- 1.7.
- 1.8.