HW4 Report

Math Problems:

USE BROWSER: Google Chrome 53.0.2785.116.

Problems:

These problems were a lot harder and took a lot of time.

I didn't know how to manipulate these matrices using the UTIL classes provided so I had to write a lot of helper functions. Some of them I found online.

Solutions:

I had to do a lot of testing to check if helper functions worked properly.

I had to look back at the lectures and class notes to solve each problem.

Question 4 was especially hard because at first I didn't know how to calculate the up-vector. The teacher cleared my confusion in class.

Coding Problems:

USE BROWSER: Firefox 49.0.2

Just press the web-buttons to make the functions work. For rotation you can also use arrow-keys and A-W-S-D keys on the keyboard.

Problems:

Seeing the rotations was hard while the object was also rotating.

At first I had over-simplified the all the problems but my confusions were cleared after talking to a classmate.

Solutions:

I added an extra button for pausing the rotation to help see the button-rotation.

I had thought only a few lines of code needed to be added but my classmate pointed out that I was not computing things properly.

The following are my solutions after understanding how to properly think about the problems

2.1 – Input file format is already evaluated by given code.

To change to cow simply uncomment this line:

//var objName = 'cow.obj'; //uncomment this to toggle cow

To see the cow look at **Figure 4.**

To display cube uncomment the cube line:

var objName = 'mycube.obj'; //uncomment this to toggle cube

2.2 – object is already normalized to the origin in the given code

2.3 – The obj-loader.js file already computes the face-normal for each face. I added onto this code to get the average normal for the vertices. I tried many ways to compute the average normal in place before the face-normal was entered into the normal array but it is not possible. So, I ended up iterating through every vertex in the vertex array to check if the vertex was already entered then I added the normal to get the average and replaced both vertex normals with the average normal. **See Figure 1** for the code picture.

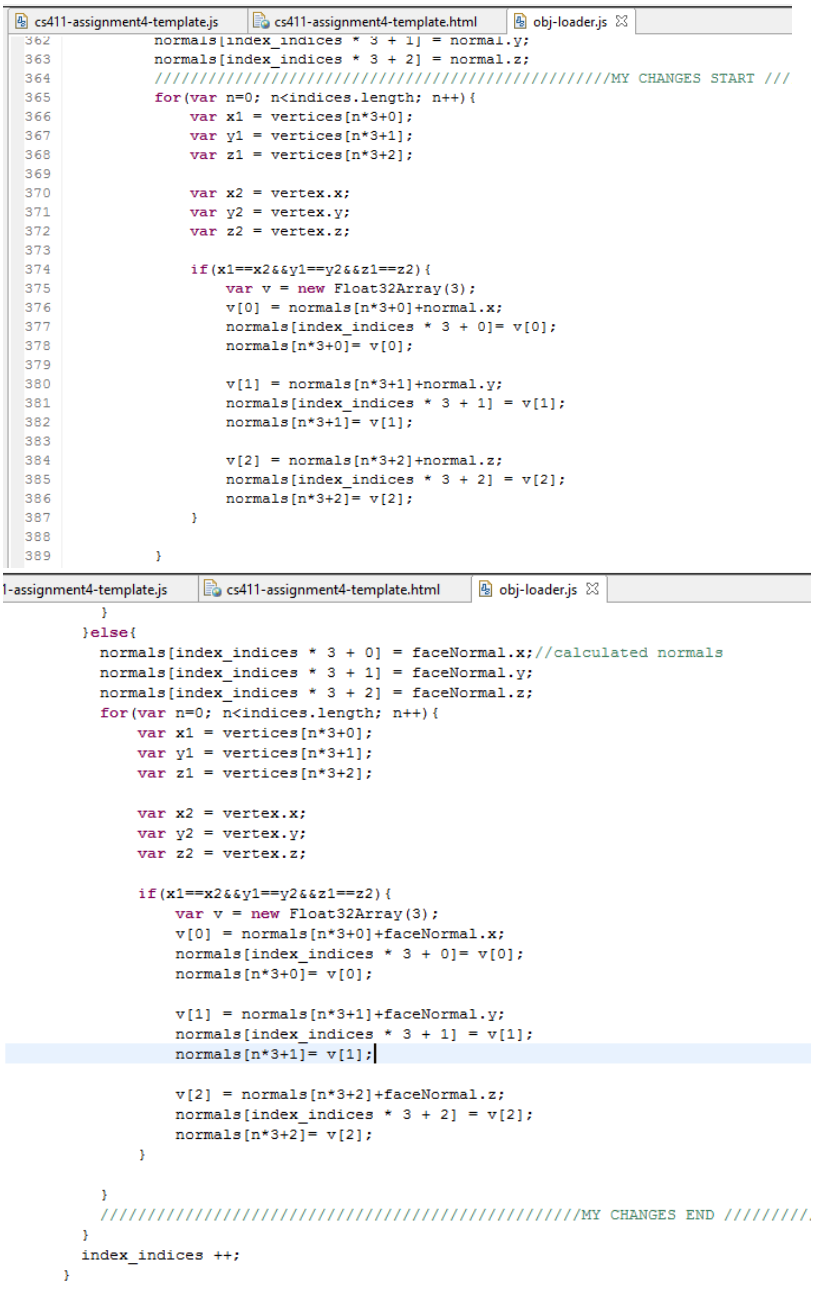
2.4 – I have made it so the user can use web-buttons, arrow keys and A-W-S-D keys to rotate the object. First to make the objet rotate I tried to multiply the mvMatrix in each rotate function but this did not work nicely. Then I found in drawScene after mvPushMatrix() call the angles were being computed so I put the following line after mvPushMAtrix() call:

mvMatrix.multiply(curRot);

Then web-buttons were already setup but for setting up the keyboard-buttons had to find the ASCII code for each key and also find how to process keyboard input. **See Figure 2** for code.

2.5 – Inverting normal did not make sense to me at first but after playing around with zoom-in/out I saw the object became black. Then I saw in initScene() function when we start reading the OBJ file with the readOBJFile() function we compute the Normals as “true”. So, I changed the variable invertNormals to a boolean. Then I copy pasted the start reading OBJ file code from initScene into the invertNormals() function. **See Figure 3** for code and result output. I was not able to find out how to find N, X, C for the computation of N dot (X – C).

Figure 1: Average normal code in obj-loader.js



cs411-assignment4-template.js | cs411-assignment4-template.html | obj-loader.js

```
362            normals[index_indices * 3 + 1] = normal.y;
363            normals[index_indices * 3 + 2] = normal.z;
364            ////////////////////////////////////////////////////MY CHANGES START ///
365            for(var n=0; n<indices.length; n++){
366                var x1 = vertices[n*3+0];
367                var y1 = vertices[n*3+1];
368                var z1 = vertices[n*3+2];
369
370                var x2 = vertex.x;
371                var y2 = vertex.y;
372                var z2 = vertex.z;
373
374                if(x1==x2&&y1==y2&&z1==z2){
375                    var v = new Float32Array(3);
376                    v[0] = normals[n*3+0]+normal.x;
377                    normals[index_indices * 3 + 0]= v[0];
378                    normals[n*3+0]= v[0];
379
380                    v[1] = normals[n*3+1]+normal.y;
381                    normals[index_indices * 3 + 1] = v[1];
382                    normals[n*3+1]= v[1];
383
384                    v[2] = normals[n*3+2]+normal.z;
385                    normals[index_indices * 3 + 2] = v[2];
386                    normals[n*3+2]= v[2];
387                }
388
389            }
```

1-assignment4-template.js | cs411-assignment4-template.html | obj-loader.js

```
            }
          }else{
            normals[index_indices * 3 + 0] = faceNormal.x;//calculated normals
            normals[index_indices * 3 + 1] = faceNormal.y;
            normals[index_indices * 3 + 2] = faceNormal.z;
            for(var n=0; n<indices.length; n++){
                var x1 = vertices[n*3+0];
                var y1 = vertices[n*3+1];
                var z1 = vertices[n*3+2];

                var x2 = vertex.x;
                var y2 = vertex.y;
                var z2 = vertex.z;

                if(x1==x2&&y1==y2&&z1==z2){
                    var v = new Float32Array(3);
                    v[0] = normals[n*3+0]+faceNormal.x;
                    normals[index_indices * 3 + 0]= v[0];
                    normals[n*3+0]= v[0];

                    v[1] = normals[n*3+1]+faceNormal.y;
                    normals[index_indices * 3 + 1] = v[1];
                    normals[n*3+1]= v[1];

                    v[2] = normals[n*3+2]+faceNormal.z;
                    normals[index_indices * 3 + 2] = v[2];
                    normals[n*3+2]= v[2];
                }

            }
            ////////////////////////////////////////////////////MY CHANGES END /////////
          }
          index_indices ++;
        }
```

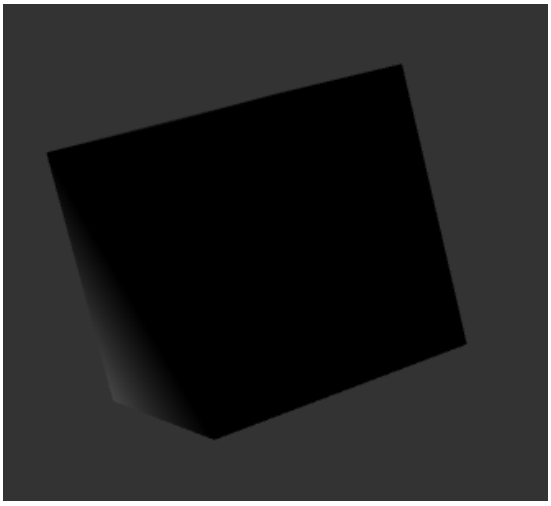
Figure 2: Button and keyboard code for rotation

```
// set button event listeners
var turnLeftBtn = document.getElementById('turnLeftBtn');
turnLeftBtn.addEventListener('click', turnLeft);
document.addEventListener('keydown', function(e){
      e = e || window.event;
      var key = e.which || e.keyCode;
      if(key===65 || key===37){       ///Press letter A key or left arrow for Left
          turnLeft();
          console.log('Turn Left');
      }
});

var turnRightBtn = document.getElementById('turnRightBtn');
turnRightBtn.addEventListener('click', turnRight);
document.addEventListener('keydown', function(e){
      e = e || window.event;
      var key = e.which || e.keyCode;
      if(key===68 || key===39){       ///Press letter D key or right arrow for Right
          turnRight();
          console.log('Turn Right');
      }
});
```

Figure 3: invertNormals code and output

```
function invertNormals()
{
  model = new Object();
  var scale=60; // 1
  readOBJFile(objName, gl, model, scale, invertNorm); // cube.obj
  invertNorm = !invertNorm;
}
```


invertNormals=true

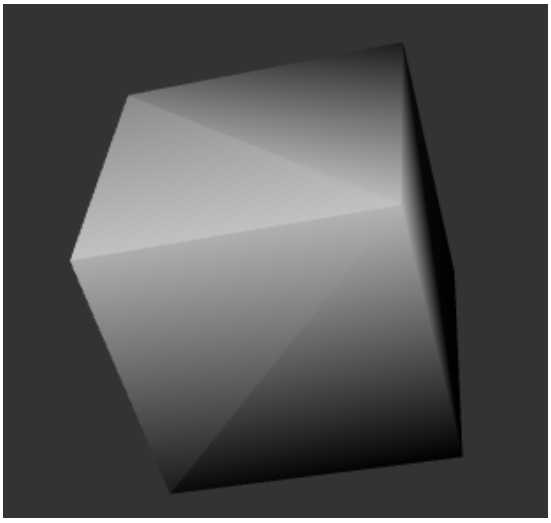

invertNormals=false

Figure 4: Cow