# CS405 Project Report


## Ahmet Melih Afşar

29457


## Introduction

For this project, the task was to modify the given code such that the diffuse lighting and ambient lighting is working correctly.

## Task 1

The main task was to change how our WebGL code handles textures. Before, it only accepted images where the width and height were a power of two.

To fix this, we took out the part of the code that was checking the image size. Then, we set the texture wrapping to GL_CLAMP_TO_EDGE. This means that if the texture coordinates go outside the image, it will just use the edge colors instead of repeating the image. After that, We changed the setting for making textures smaller (minification) to GL_LINEAR. This was necessary because non-power-of-two textures can't use the default setting that relies on mipmaps.

## Task 2

The second task in our WebGL project involved adding  basic lighting elements into our 3D scene. The objective was to enhance the visual aspects by adding ambient and diffuse lighting effects.

To complete the task, I took some help from the project4 file under “week 12”  recitation files.

For “setAmbientLight” I added the following lines:

```
        gl.useProgram(this.prog);
        gl.uniform1f(this.ambientLoc, ambient);
```

The first line tells WebGL to use the shader program referred to by “this.prog”.

The second line sets the value of a uniform variable for the current shader program. Uniforms are variables that remain the same for all vertices and fragments during a single draw call.

Similarly, for “enableLighting” I added these:

```
        gl.useProgram(this.prog);
        gl.uniform1i(this.enableLightingLoc, show);
```

For “setMesh”, I added these:

```
        gl.bindBuffer(gl.ARRAY_BUFFER, this.normalbuffer);
        gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(normalCoords),
gl.STATIC_DRAW);
```

The first line tells WebGL to use this.normalbuffer as the current array buffer. Array buffers are used to store vertex attributes, such as vertex coordinates, texture coordinate data, or vertex color data.

The second line creates and initializes the current array buffer with the normal coordinate data. The new Float32Array(normalCoords) part creates a new typed array with the normal coordinate data, and gl.STATIC_DRAW is a hint to WebGL about how the buffer will be used. gl.STATIC_DRAW means that the buffer's contents will be used many times and are not likely to change.

Fot the constructor, we needed to add some new variables and initialize them. For example,

```
        this.normalLoc = gl.getAttribLocation(this.prog, 'normal');
```

Here we return the location of the “normal” in a given WebGLProgram object (this.prog in here).

```
this.normalbuffer = gl.createBuffer();
        this.ambientLoc = gl.getUniformLocation(this.prog, 'ambient');
        this.enableLightingLoc = gl.getUniformLocation(this.prog,
'enableLighting');
        this.lightSourceLoc = gl.getUniformLocation(this.prog, 'lightPos');
```

Here, we create a buffer for normal and get the locations of several attributes, so that we can use them later.

For the last part, I needed to change the fragment and the vertex shaders to function as intended. In this part, I used the code from the recitations as a reference, and built up from there.

For the meshVS, I had to change the following line:

```
                v_normal = vec3(mvp * vec4(normal,1));
```

to

```
                v_normal = vec3(mvp * vec4(normal,0));
```

This change is done to treat the normal vector as a direction rather than a position.

For the meshFS, I changed the lighting strategy so that after we calculate the ambient and diffuse light intensities, we take whichever one is bigger. This ensures that the parts of the object that does not see the light source has can be seen from the camera, and it also ensures that the light source facing part of the object is illuminated correctly. This is done by the following code:

```
                    gl_FragColor = max(texture2D(tex, v_texCoord) * (ambient),
texture2D(tex, v_texCoord) * (diffuse + ambient));
```

The update location function is working as intended as well, since we used locations we declared for the light source and it is updated every frame.