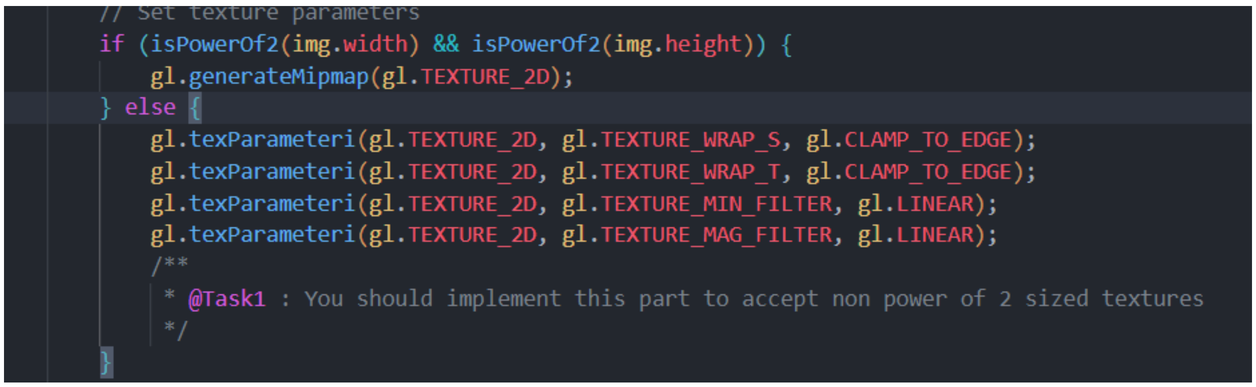
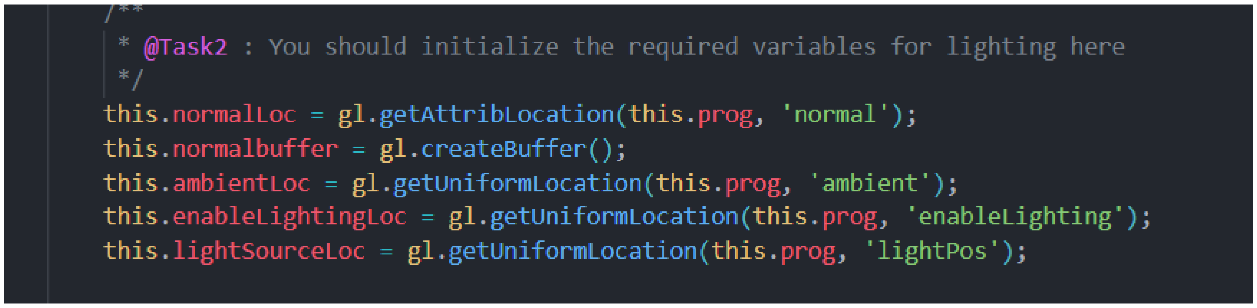
# CS 405 PROJECT 2
Sıla Özinan
28161

**Task 1:**
Normally, this code only accepts pictures that have width and height values of a power of two. In order to change it and make it take the power of two values as well following changes are made in the code. The first one is **texture wrapping** for both the S and T axes with clamp_to_edge to prevent tiled texture or any other problems. Then I applied **minification and magnification filters** for consistent texture scaling.



```
    // Set texture parameters
    if (isPowerOf2(img.width) && isPowerOf2(img.height)) {
        gl.generateMipmap(gl.TEXTURE_2D);
    } else {
        gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_WRAP_S, gl.CLAMP_TO_EDGE);
        gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_WRAP_T, gl.CLAMP_TO_EDGE);
        gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR);
        gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.LINEAR);
        /**
         * @Task1 : You should implement this part to accept non power of 2 sized textures
         */
    }
```

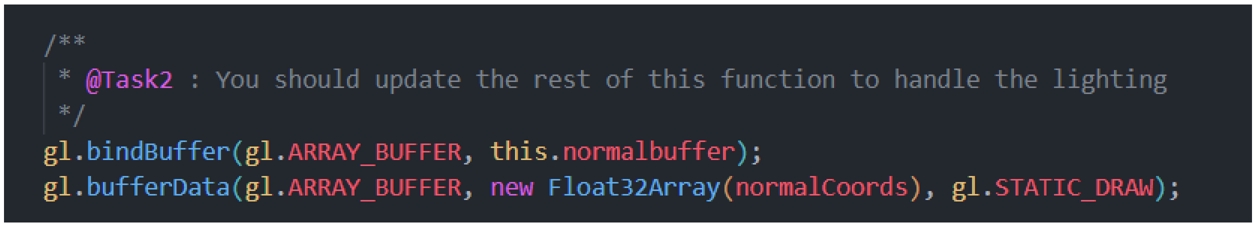
**Task 2:**
**Inside the constructor():**
I initialized the necessary variables for lighting in the constructor. New uniform locations for lighting parameters were added and a buffer to store necessary normals for lighting computations was created.



```
    /**
     * @Task2 : You should initialize the required variables for lighting here
     */
    this.normalLoc = gl.getAttribLocation(this.prog, 'normal');
    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');
```

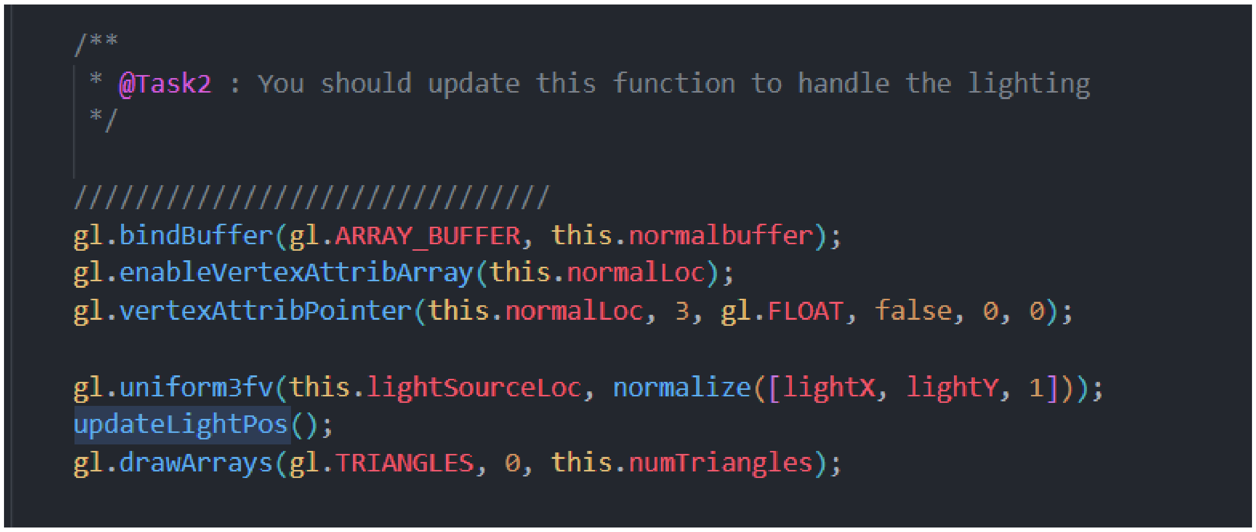
**Inside the setMesh(vertPos, texCoords, normalCoords):**

In addition to updating vertex positions and texture coordinates, the normals are now also buffered.



```
/**
 * @Task2 : You should update the rest of this function to handle the lighting
 */
gl.bindBuffer(gl.ARRAY_BUFFER, this.normalbuffer);
gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(normalCoords), gl.STATIC_DRAW);
```

**Inside draw(trans):**
The shader's dynamic lighting effects are made possible by configuring normal vectors and light source positions as uniforms and vertex attributes, respectively, depending on the surface orientation and light position.



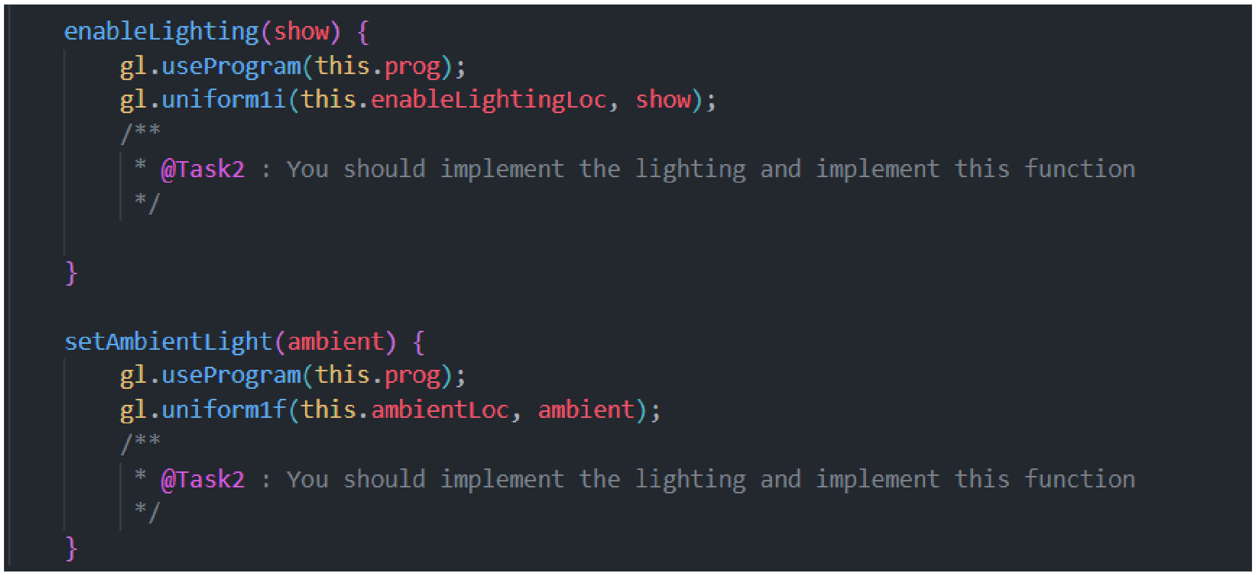
```
/**
 * @Task2 : You should update this function to handle the lighting
 */

////////////////////////////////
gl.bindBuffer(gl.ARRAY_BUFFER, this.normalbuffer);
gl.enableVertexAttribArray(this.normalLoc);
gl.vertexAttribPointer(this.normalLoc, 3, gl.FLOAT, false, 0, 0);

gl.uniform3fv(this.lightSourceLoc, normalize([lightX, lightY, 1]));
updateLightPos();
gl.drawArrays(gl.TRIANGLES, 0, this.numTriangles);
```

**Inside enableLighting(show) and setAmbientLight(ambient):**
These two were used to manage lighting effects. A boolean is set based on the function argument to toggle lighting effects on or off to allow the user to enable or disable lighting during runtime. Additionally in the setAmbientLight uniform variable for ambient light intensity is set based on the function argument. This enables dynamic control over the

ambient lighting in the scene.

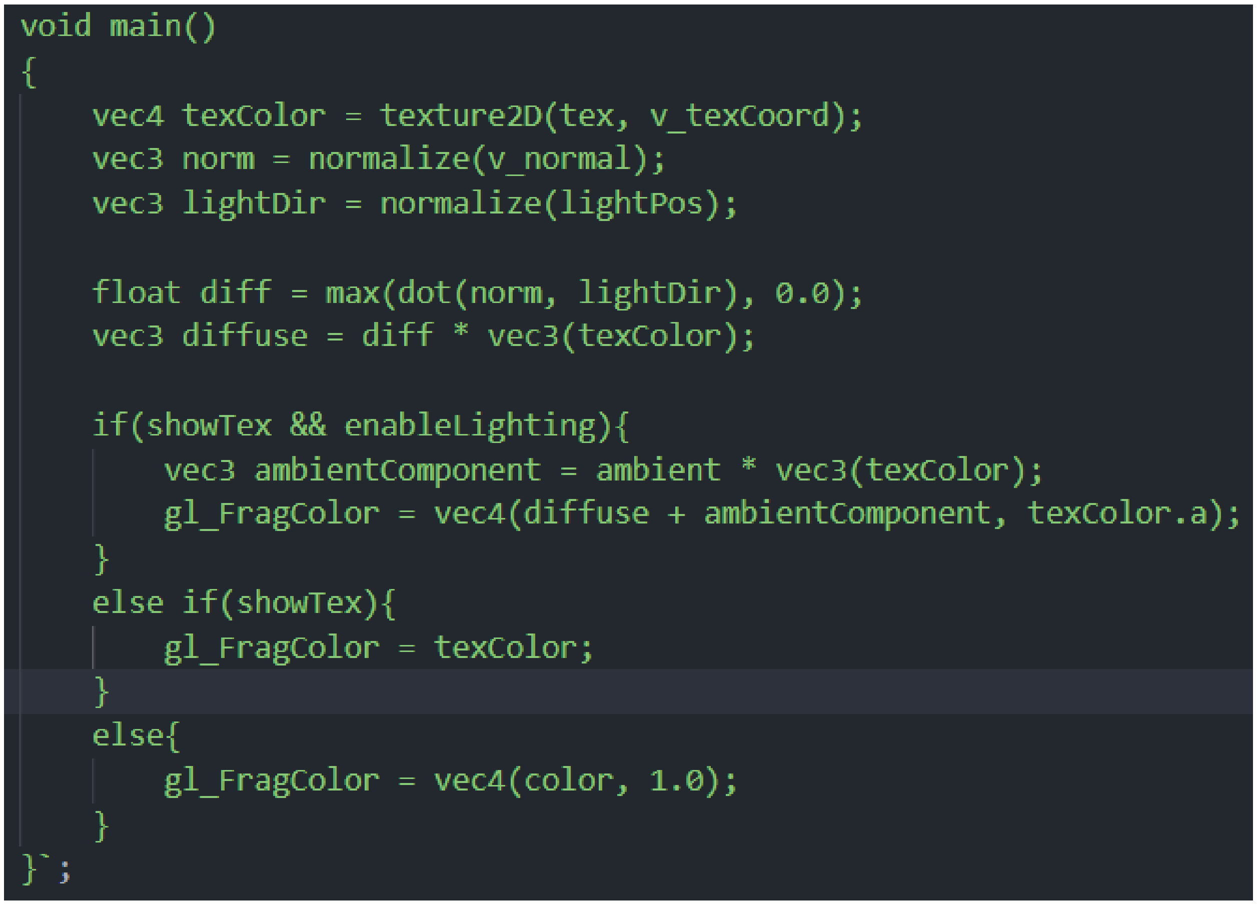


```
enableLighting(show) {
    gl.useProgram(this.prog);
    gl.uniform1i(this.enableLightingLoc, show);
    /**
     * @Task2 : You should implement the lighting and implement this function
     */

}

setAmbientLight(ambient) {
    gl.useProgram(this.prog);
    gl.uniform1f(this.ambientLoc, ambient);
    /**
     * @Task2 : You should implement the lighting and implement this function
     */
}
```

**Inside meshFS:**
I made updates to calculate the effects of lighting in the fragments of mesh. First I find the color from the texture of the given coordinate. After that, I normalized the vectors of surface normal and light direction for accurate calculation. Then, I calculated the diffuse then  I calculated the ambient light by multiplying the ambient light intensity with the texture color. Lastly, I combined ambient and diffuse light. **gl_FragColor = vec4(diffuse + ambientComponent, texColor.a);**


```
void main()
{
    vec4 texColor = texture2D(tex, v_texCoord);
    vec3 norm = normalize(v_normal);
    vec3 lightDir = normalize(lightPos);

    float diff = max(dot(norm, lightDir), 0.0);
    vec3 diffuse = diff * vec3(texColor);

    if(showTex && enableLighting){
        vec3 ambientComponent = ambient * vec3(texColor);
        gl_FragColor = vec4(diffuse + ambientComponent, texColor.a);
    }
    else if(showTex){
        gl_FragColor = texColor;
    }
    else{
        gl_FragColor = vec4(color, 1.0);
    }
}`;
```