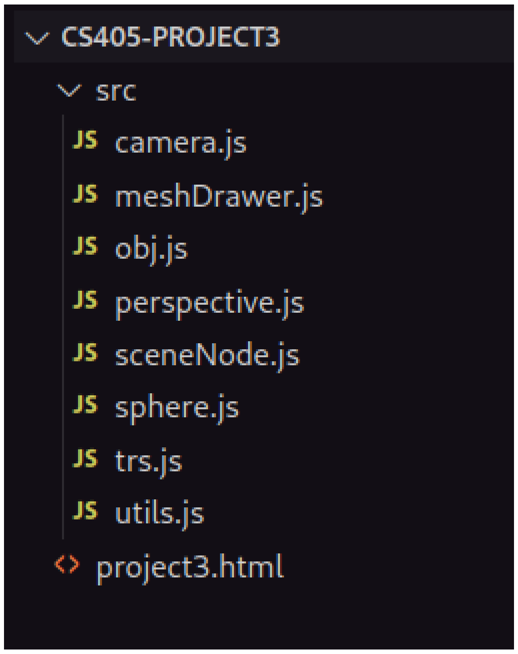
CS 405 Project 3: Scene Graph + Illumination

In this project, you will modify an existing scene graph implementation to create a basic solar system simulation. Please read the instructions below about the project carefully.

Instructions

Please download the assignment folder from SUcourse. Inside the folder, you should see multiple js files under the src directory.



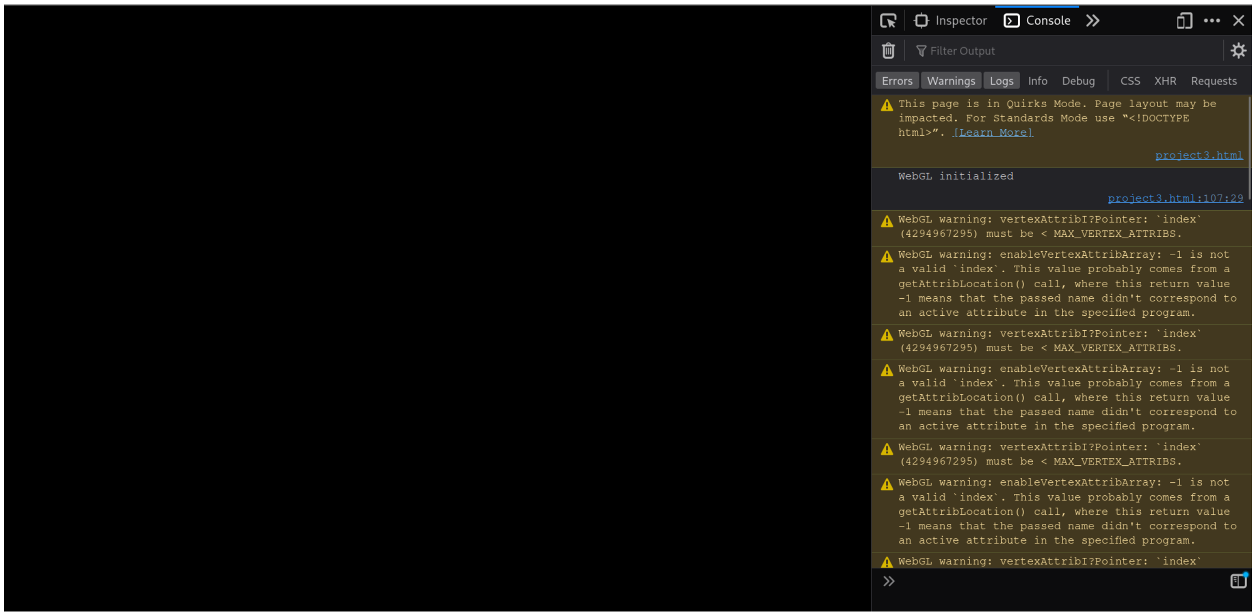
```
CS405-PROJECT3
  src
    camera.js
    meshDrawer.js
    obj.js
    perspective.js
    sceneNode.js
    sphere.js
    trs.js
    utils.js
  project3.html
```

In this project, you have 3 tasks to perform. Each task requires a specific file to be modified. For each task, **you should only modify the respective file.** Any submission that modifies any other file will not be accepted. You can find the purpose of each file below:

- **camera.js:** Helper class for the camera. Used to calculate the view (lookAt) matrix.
- **meshDrawer.js:** Helper class for drawing meshes to the scene. In task 2, you will modify the fragment shader in this file to implement diffuse and specular light.
- **obj.js:** Contains the object helper class.
- **perspective.js:** Helper class for the perspective. Used to calculate the perspective (projection) matrix.
- **sceneNode.js:** Implementation of the nodes in a scene graph. In task 1, you will implement the draw method of the scene node to apply transformations from parent to child.

- **sphere.js:** Contains the sphere object you will use in this assignment.
- **trs.js:** Helper class for transformations (translation, scaling, rotating).
- **utils.js:** Contains helper functions for mathematical operations.
- **project3.html:** Contains the main functionality of the project. In task 3, you should modify the specified regions to add a node to the scene graph.

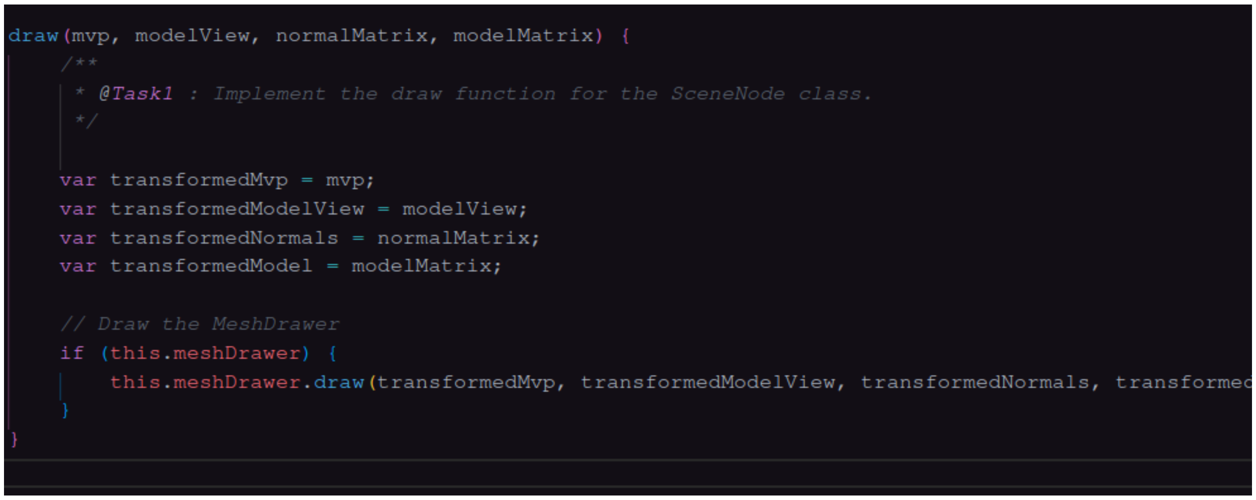
When you open the project3.html, you should see the empty screen below:



```
This page is in Quirks Mode. Page layout may be
impacted. For Standards Mode use "<!DOCTYPE
html>". [Learn More]
                                        project3.html
WebGL initialized
                                project3.html:107:29
WebGL warning: vertexAttribI?Pointer: `index`
(4294967295) must be < MAX_VERTEX_ATTRIBS.
WebGL warning: enableVertexAttribArray: -1 is not
a valid `index`. This value probably comes from a
getAttribLocation() call, where this return value
-1 means that the passed name didn't correspond to
an active attribute in the specified program.
WebGL warning: vertexAttribI?Pointer: `index`
(4294967295) must be < MAX_VERTEX_ATTRIBS.
WebGL warning: enableVertexAttribArray: -1 is not
a valid `index`. This value probably comes from a
getAttribLocation() call, where this return value
-1 means that the passed name didn't correspond to
an active attribute in the specified program.
WebGL warning: vertexAttribI?Pointer: `index`
(4294967295) must be < MAX_VERTEX_ATTRIBS.
WebGL warning: enableVertexAttribArray: -1 is not
a valid `index`. This value probably comes from a
getAttribLocation() call, where this return value
-1 means that the passed name didn't correspond to
an active attribute in the specified program.
WebGL warning: vertexAttribI?Pointer: `index`
```

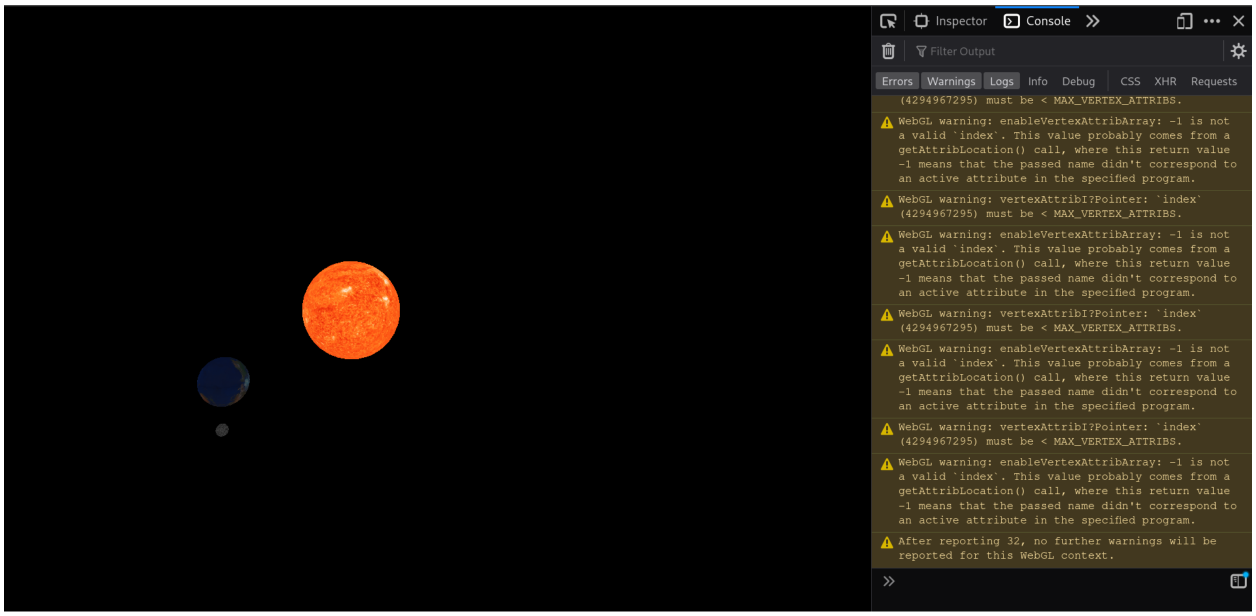
Task 1

In task 1, you will implement the **"draw"** function for the scene graph. If you open the **sceneNode.js** file and check the draw function inside the sceneNode class, you should see an incomplete implementation below:



```
draw(mvp, modelView, normalMatrix, modelMatrix) {
    /**
     * @Task1 : Implement the draw function for the SceneNode class.
     */

    var transformedMvp = mvp;
    var transformedModelView = modelView;
    var transformedNormals = normalMatrix;
    var transformedModel = modelMatrix;

    // Draw the MeshDrawer
    if (this.meshDrawer) {
        this.meshDrawer.draw(transformedMvp, transformedModelView, transformedNormals, transformed
    }
}
```

You should implement this method properly such that any transformation applied to the parent should propagate to the children nodes. If you implement the draw method properly, you should be seeing the following output:

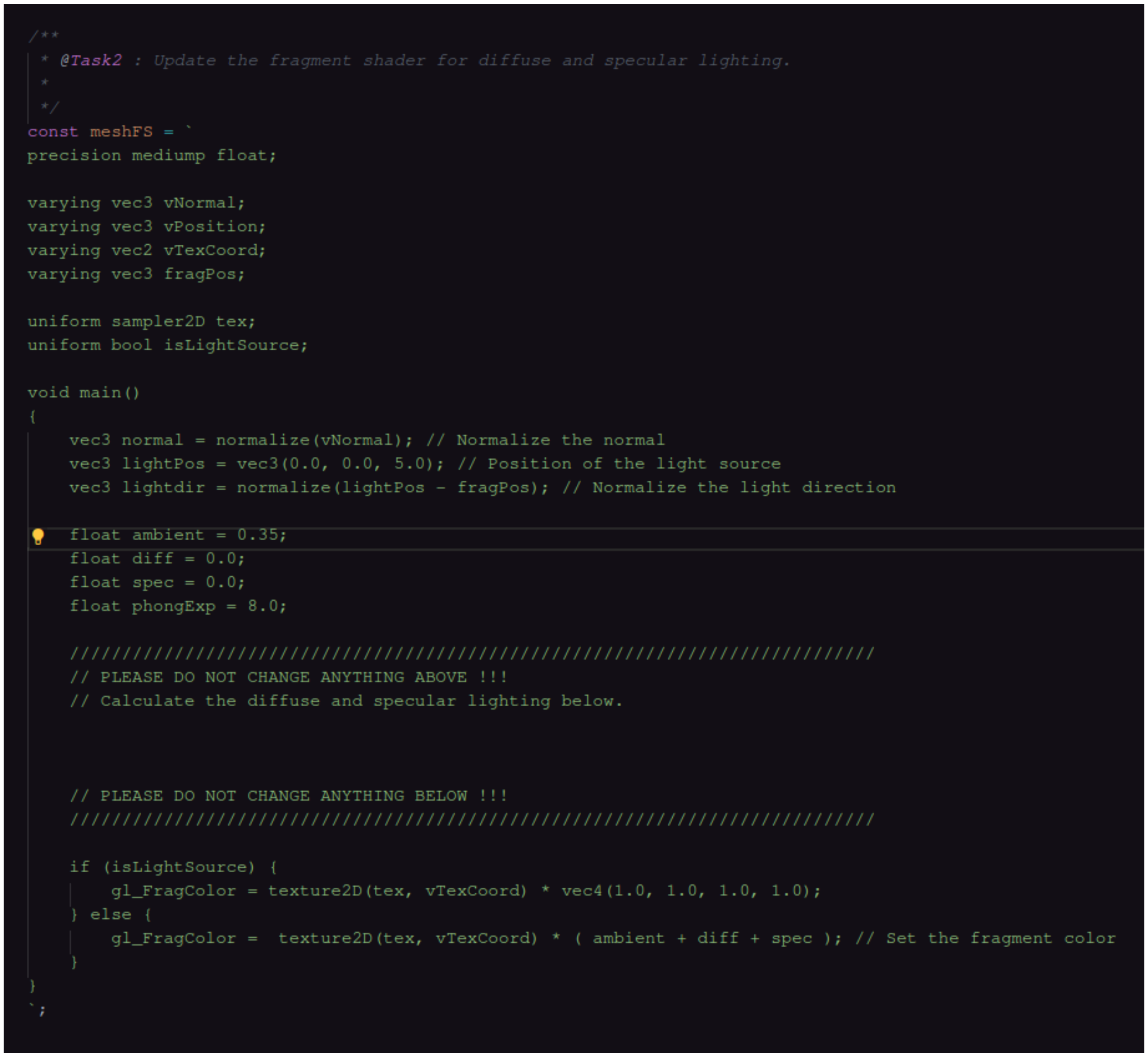


```
 (4294967295) must be < MAX_VERTEX_ATTRIBS.
WebGL warning: enableVertexAttribArray: -1 is not
a valid `index`. This value probably comes from a
getAttribLocation() call, where this return value
-1 means that the passed name didn't correspond to
an active attribute in the specified program.
WebGL warning: vertexAttribI?Pointer: `index`
(4294967295) must be < MAX_VERTEX_ATTRIBS.
WebGL warning: enableVertexAttribArray: -1 is not
a valid `index`. This value probably comes from a
getAttribLocation() call, where this return value
-1 means that the passed name didn't correspond to
an active attribute in the specified program.
WebGL warning: vertexAttribI?Pointer: `index`
(4294967295) must be < MAX_VERTEX_ATTRIBS.
WebGL warning: enableVertexAttribArray: -1 is not
a valid `index`. This value probably comes from a
getAttribLocation() call, where this return value
-1 means that the passed name didn't correspond to
an active attribute in the specified program.
WebGL warning: vertexAttribI?Pointer: `index`
(4294967295) must be < MAX_VERTEX_ATTRIBS.
WebGL warning: enableVertexAttribArray: -1 is not
a valid `index`. This value probably comes from a
getAttribLocation() call, where this return value
-1 means that the passed name didn't correspond to
an active attribute in the specified program.
After reporting 32, no further warnings will be
reported for this WebGL context.
```

In this output, you can see the Sun, the Earth, and the Moon. However, light implementation has not been properly implemented. We will handle the light in task 2.

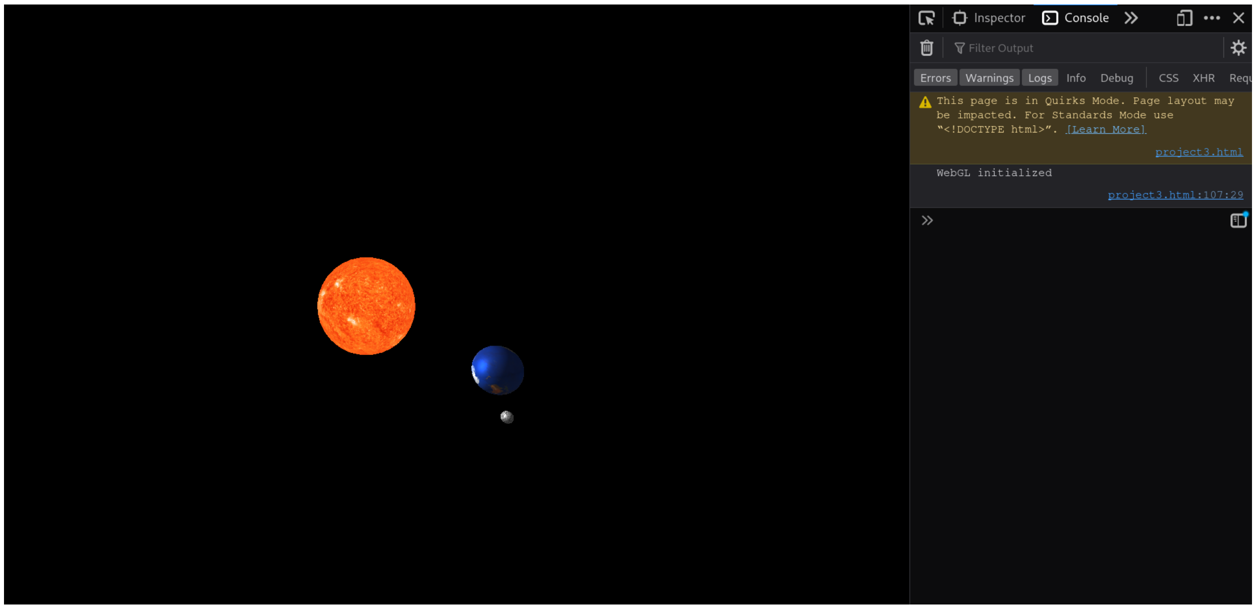
Task 2

In this task, we will update the fragment shader inside of the **meshDrawer.js** file. The current implementation only supports ambient lighting. You should update the fragment shader below to calculate the diffuse and specular lighting properly.



```
/**
 * @Task2 : Update the fragment shader for diffuse and specular lighting.
 *
 */
const meshFS = `
precision mediump float;

varying vec3 vNormal;
varying vec3 vPosition;
varying vec2 vTexCoord;
varying vec3 fragPos;

uniform sampler2D tex;
uniform bool isLightSource;

void main()
{
    vec3 normal = normalize(vNormal); // Normalize the normal
    vec3 lightPos = vec3(0.0, 0.0, 5.0); // Position of the light source
    vec3 lightdir = normalize(lightPos - fragPos); // Normalize the light direction

    float ambient = 0.35;
    float diff = 0.0;
    float spec = 0.0;
    float phongExp = 8.0;

    ///////////////////////////////////////////////////////////////////////////////
    // PLEASE DO NOT CHANGE ANYTHING ABOVE !!!
    // Calculate the diffuse and specular lighting below.



    // PLEASE DO NOT CHANGE ANYTHING BELOW !!!
    ///////////////////////////////////////////////////////////////////////////////

    if (isLightSource) {
        gl_FragColor = texture2D(tex, vTexCoord) * vec4(1.0, 1.0, 1.0, 1.0);
    } else {
        gl_FragColor =  texture2D(tex, vTexCoord) * ( ambient + diff + spec ); // Set the fragment color
    }
}
`;
```

If you calculated the light properly, you should get the following output:



```
This page is in Quirks Mode. Page layout may
be impacted. For Standards Mode use
"<!DOCTYPE html>". [Learn More]
                                  project3.html
WebGL initialized
                          project3.html:107:29
```

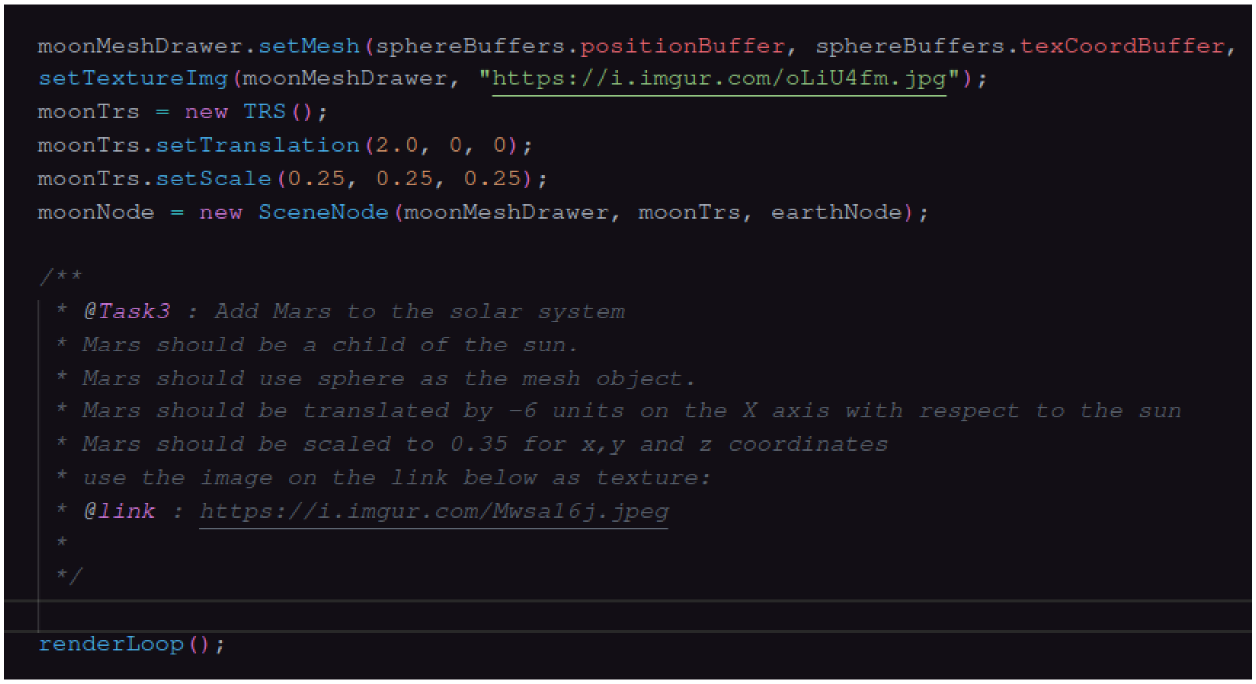
Task 3

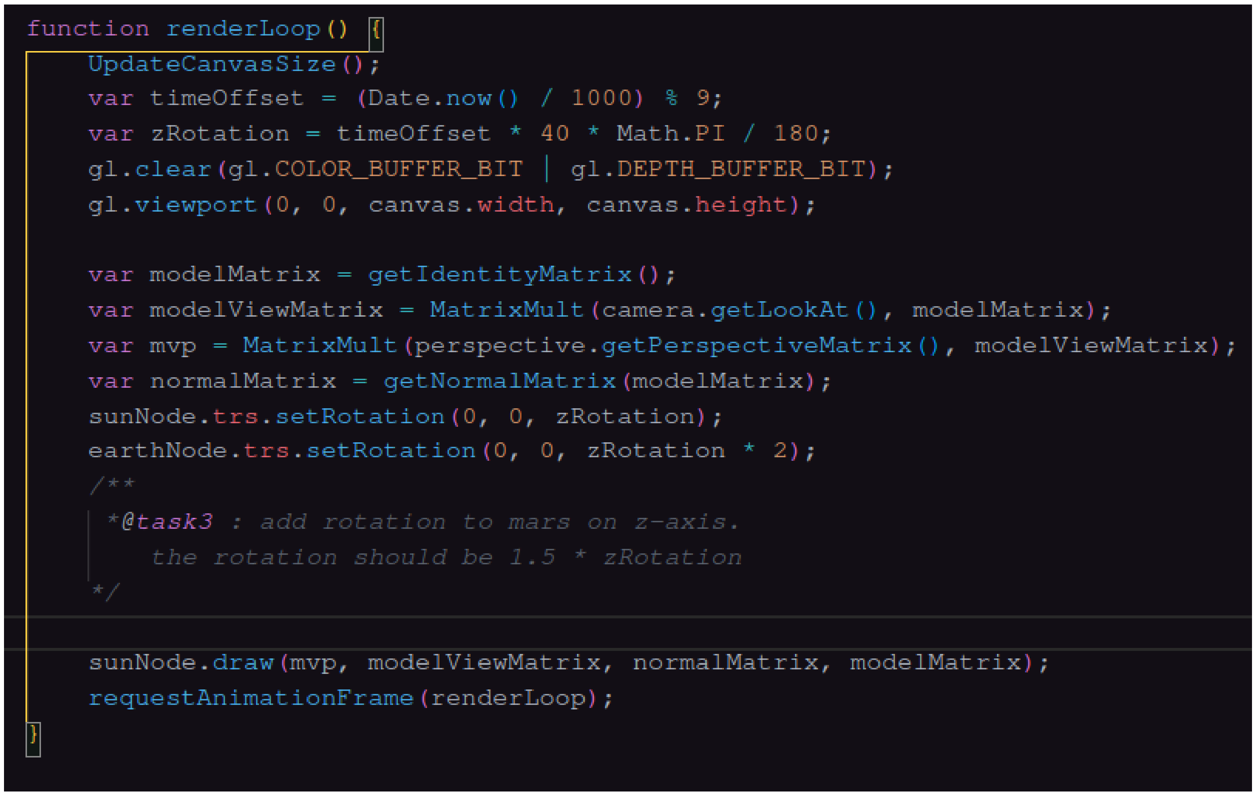
In the third task, you will add Mars planet to the solar system. To add Mars to the solar system, you should draw Mars in the scene and add it to the scene graph as the child of the Sun node. You can find the details of the Mars node below:

- Mars should be a child of the sun.
- Mars should use the "sphere" as the mesh object.
- Mars should be translated by -6 units on the X-axis with respect to the sun
- Mars should be scaled to 0.35 for x,y, and z coordinates
- Mars should be rotated around its z-axis 1.5 times the sun's rotation (check renderLoop method inside the project3.html)
- use the image on the link below as texture:

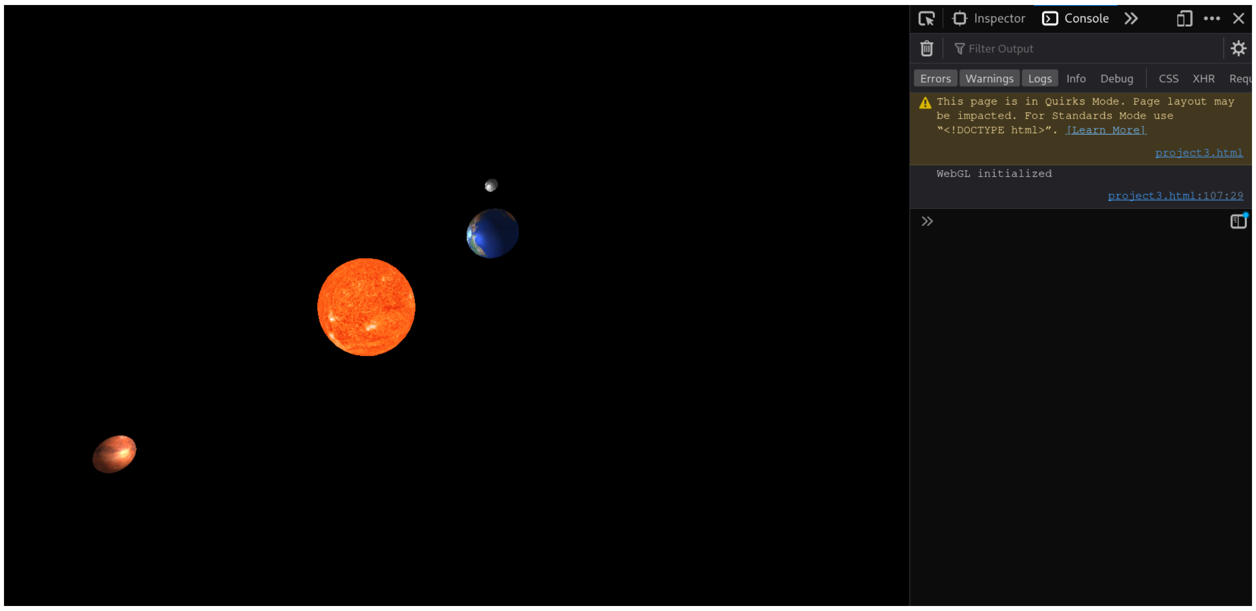
Mars texture link: https://i.imgur.com/Mwsa16j.jpeg

To add Mars to your scene, you should change the respected parts below:


```
moonMeshDrawer.setMesh(sphereBuffers.positionBuffer, sphereBuffers.texCoordBuffer,
setTextureImg(moonMeshDrawer, "https://i.imgur.com/oLiU4fm.jpg");
moonTrs = new TRS();
moonTrs.setTranslation(2.0, 0, 0);
moonTrs.setScale(0.25, 0.25, 0.25);
moonNode = new SceneNode(moonMeshDrawer, moonTrs, earthNode);

/**
 * @Task3 : Add Mars to the solar system
 * Mars should be a child of the sun.
 * Mars should use sphere as the mesh object.
 * Mars should be translated by -6 units on the X axis with respect to the sun
 * Mars should be scaled to 0.35 for x,y and z coordinates
 * use the image on the link below as texture:
 * @link : https://i.imgur.com/Mwsa16j.jpeg
 *
 */

renderLoop();
```


```
function renderLoop() {
    UpdateCanvasSize();
    var timeOffset = (Date.now() / 1000) % 9;
    var zRotation = timeOffset * 40 * Math.PI / 180;
    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
    gl.viewport(0, 0, canvas.width, canvas.height);

    var modelMatrix = getIdentityMatrix();
    var modelViewMatrix = MatrixMult(camera.getLookAt(), modelMatrix);
    var mvp = MatrixMult(perspective.getPerspectiveMatrix(), modelViewMatrix);
    var normalMatrix = getNormalMatrix(modelMatrix);
    sunNode.trs.setRotation(0, 0, zRotation);
    earthNode.trs.setRotation(0, 0, zRotation * 2);
    /**
     *@task3 : add rotation to mars on z-axis.
        the rotation should be 1.5 * zRotation
     */

    sunNode.draw(mvp, modelViewMatrix, normalMatrix, modelMatrix);
    requestAnimationFrame(renderLoop);
}
```

If you did everything correctly, this should be what your scene looks like in the end:



```
This page is in Quirks Mode. Page layout may
be impacted. For Standards Mode use
"<!DOCTYPE html>". [Learn More]
                                  project3.html
WebGL initialized
                           project3.html:107:29
```

Report

Additionally, we expect you to write a report that clearly explains your methodology. **Any submission without the report will not be graded!**

Submission Guidelines

You should upload your work **to both GitHub and SuCourse.**
GitHub: Uploading the codes only is sufficient. Once you upload your code to GitHub, write the repository link to a txt file named github-link.txt and include that file with your submission to SuCourse.
SuCourse: You should zip all of your work (report, the code(s), and github-link.txt file) and upload it to SuCourse.

Important: Plagiarism will not be tolerated!