CS405 Report

Student Name-Surname: Ahmethan Özcan
Student ID: 28483

TASK 1:

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

        // Get the transformation matrix from this node's TRS object
        var transformationMatrix = this.trs.getTransformationMatrix();

        // Apply the transformation matrix to the passed matrices
        var transformedMvp = MatrixMult(mvp, transformationMatrix);
        var transformedModelView = MatrixMult(modelView,
transformationMatrix);
        var transformedNormals = MatrixMult(normalMatrix,
transformationMatrix);
        var transformedModel = MatrixMult(modelMatrix,
transformationMatrix);

        // Draw the MeshDrawer with updated matrices
        if (this.meshDrawer) {
            this.meshDrawer.draw(transformedMvp, transformedModelView,
transformedNormals, transformedModel);
        }

        // Recursively call draw on children nodes
        for (var i = 0; i < this.children.length; i++) {
            this.children[i].draw(transformedMvp, transformedModelView,
transformedNormals, transformedModel);
        }
    }
```

- var transformationMatrix = this.trs.getTransformationMatrix(): Retrieves the node-specific transformation matrix combining translation, rotation, and scaling.

- var transformedMvp = MatrixMult(mvp, transformationMatrix): Applies node transformations to the Model-View-Projection (MVP) matrix.
- var transformedModelView = MatrixMult(modelView, transformationMatrix): Applies transformations to the Model-View matrix, affecting object position and orientation.
- var transformedNormals = MatrixMult(normalMatrix, transformationMatrix): Transforms normal vectors, ensuring correct lighting calculations under transformation.
- var transformedModel = MatrixMult(modelMatrix, transformationMatrix): Applies transformations to the Model matrix, affecting object's world space positioning.
- this.meshDrawer.draw(transformedMvp, transformedModelView, transformedNormals, transformedModel): Renders the mesh with the transformed matrices if the node has a drawable mesh.
- for (var i = 0; i < this.children.length; i++) { ... }: Recursively applies the same drawing process to all child nodes, ensuring hierarchical transformations.

TASK 2:

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

    float diffIntensity = max(dot(normal, lightdir), 0.0);
    vec3 viewDir = normalize(-vPosition); // Assuming the camera is at the
origin
    vec3 reflectDir = reflect(lightdir, normal);
    float specIntensity = pow(max(dot(viewDir, reflectDir), 0.0),
phongExp);

    // Combine the lighting components
    diff = diffIntensity;
    spec = specIntensity;



    // PLEASE DO NOT CHANGE ANYTHING BELOW !!!

////////////////////////////////////////////////////////////////////////
///
```

- float diffIntensity = max(dot(normal, lightdir), 0.0): Calculates diffuse lighting by taking the dot product of the surface normal and light direction, clamping the result to a minimum of zero.

- vec3 viewDir = normalize(-vPosition`: Determines the view direction from the camera to the fragment, assuming the camera is at the origin.
- vec3 reflectDir = reflect(lightdir, normal): Computes the reflection direction of light on the surface, based on the light direction and surface normal.
- float specIntensity = pow(max(dot(viewDir, reflectDir), 0.0), phongExp): Calculates specular lighting by raising the angle between the view and reflection directions to the power of `phongExp`, ensuring non-negative values.
- diff = diffIntensity: Assigns the calculated diffuse intensity to the `diff` variable.
- spec = specIntensity: Assigns the calculated specular intensity to the `spec` variable.

TASK 3:

```
/**
                 * @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
                 *
                 */
                marsMeshDrawer.setMesh(sphereBuffers.positionBuffer,
sphereBuffers.texCoordBuffer, sphereBuffers.normalBuffer);
                setTextureImg(marsMeshDrawer,
"https://i.imgur.com/Mwsa16j.jpeg");
                marsTrs = new TRS();
                marsTrs.setTranslation(-6, 0, 0);
                marsTrs.setScale(0.35, 0.35, 0.35);
                marsNode = new SceneNode(marsMeshDrawer, marsTrs,
sunNode);
```

- marsMeshDrawer.setMesh(sphereBuffers.positionBuffer, sphereBuffers.texCoordBuffer, sphereBuffers.normalBuffer): Sets up Mars with a sphere mesh using position, texture coordinate, and normal buffers.
- setTextureImg(marsMeshDrawer, "https://i.imgur.com/Mwsa16j.jpeg"): Applies the Mars texture from the provided URL to the sphere mesh.
- marsTrs = new TRS(): Initializes a new TRS (Translation, Rotation, Scale) object for Mars.
- marsTrs.setTranslation(-6, 0, 0): Positions Mars 6 units left of the Sun on the X-axis.
- marsTrs.setScale(0.35, 0.35, 0.35): Scales Mars to 35% of its original size in all dimensions.

- marsNode = new SceneNode(marsMeshDrawer, marsTrs, sunNode): Creates a scene node for Mars, making it a child of the Sun node in the scene graph.

```
/**
                 *@task3 : add rotation to mars on z-axis.
                    the rotation should be 1.5 * zRotation
                */
                sunNode.draw(mvp, modelViewMatrix, normalMatrix,
modelMatrix);
                requestAnimationFrame(renderLoop);
```

- sunNode.draw(mvp, modelViewMatrix, normalMatrix, modelMatrix): Calls the `draw` method on the Sun node, which recursively renders the Sun and all of its child nodes, including Mars, with the specified transformation matrices.
- requestAnimationFrame(renderLoop): Requests the browser to call the `renderLoop` function before the next repaint, ensuring smooth animations and rendering updates, including the rotation of Mars.