CS405 Project 3

Alp Civan 28408

Task 1

The draw function for the SceneNode class renders the node and its children in the scene graph. It takes four matrices from the parent node: mvp, modelView, normalMatrix, and modelMatrix. It does the following:

- It gets the node’s local transformation matrix from its trs object.
- It multiplies the local matrix with the parent’s matrices, and gets the node’s global matrices.
- It draws the node’s mesh with the global matrices, if it has a meshDrawer object.
- It draws the children nodes with the same global matrices, by calling their draw functions.

The draw function lets us make a tree of nodes, where each node has its own transformations and also gets the parent’s transformations. This helps us make complex scenes with related objects.

```
  draw(mvp, modelView, normalMatrix, modelMatrix) {
    // Get the node's transformation matrix and multiply it with the parent's matrices
    var transformationMatrix = this.trs.getTransformationMatrix();
    [mvp, modelView, normalMatrix, modelMatrix] = [
      mvp,
      modelView,
      normalMatrix,
      modelMatrix,
    ].map((matrix) => MatrixMult(matrix, transformationMatrix));

    // Draw the MeshDrawer if it exists
    this.meshDrawer &&
      this.meshDrawer.draw(mvp, modelView, normalMatrix, modelMatrix);

    // Draw the children nodes using the same matrices
    this.children.forEach((child) =>
      child.draw(mvp, modelView, normalMatrix, modelMatrix)
    );
  }
}
```

(parameter) matrix: any

Task 2

The fragment shader for diffuse and specular lighting calculates the lighting components for each fragment based on the normal vector, the light direction vector and the view direction vector. It combines the ambient, diffuse, and specular components to get the final color of the fragment. If the fragment belongs to a light source, it simply uses the texture color without any lighting calculations.

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

float diffuseCoefficient = max(dot(normal, lightdir), 0.0);

vec3 viewDirection = normalize(-vPosition);
vec3 reflectedLightDirection = reflect(lightdir, normal);
float specularCoefficient = pow(max(dot(viewDirection, reflectedLightDirection), 0.0), phongExp);

float diffuse = diffuseCoefficient;
float specular = specularCoefficient;

diff = diffuse;
spec = specular;



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

Task 3

I started by defining a new MeshDrawer and giving it the name mesh_mars for this task. Then, I added the necessary sphereBuffers to the mesh_mars. After that, I established the translation and scale of a new TRS. I then used this Mars and mesh_mars to construct a new marsNode, which I linked to the sunNode. I then used setRotation to set its rotation. These are all satisfying these rules down below:

- 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

```
    earthNode.trs.setRotation(0, 0, zRotation * 2);

    marsNode.trs.setRotation(0, 0, zRotation * 1.5);
    sunNode.draw(mvp, modelViewMatrix, normalMatrix, modelMatrix);
    requestAnimationFrame(renderLoop);
```

```
    earthTrs = new TRS();
    earthTrs.setTranslation(3, 0, 0);
    earthTrs.setScale(0.5, 0.5, 0.5);
    earthNode = new SceneNode(earthMeshDrawer, earthTrs, sunNode);

    moonMeshDrawer.setMesh(sphereBuffers.positionBuffer, sphereBuffers.texC
    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);

    mesh_mars.setMesh(sphereBuffers.positionBuffer, sphereBuffers.texCoordB

    setTextureImg(mesh_mars, "https://i.imgur.com/Mwsa16j.jpeg");

    var Mars = new TRS();

    Mars.setTranslation(-6, 0, 0);
    Mars.setScale(0.35, 0.35, 0.35);
    marsNode = new SceneNode(mesh_mars, Mars ,sunNode);

    renderLoop();
};
```