CS 405 Project 3: Solar System Scene Graph

Sezer Enis Birgili

#31033

Overview

This project involved modifying an incomplete basic solar system simulation to implement drawing, lighting and an added mars planet object. The tasks required implementing a draw method for the scene graph, adding diffuse and specular lighting in the fragment shader, and extending the scene graph to include a Mars node to be seen in the model solar system. This report details the methodology and implementation for each task.

Task 1: Implementing Draw( ) in sceneNode.js

Objective

The goal was to implement the draw method in the sceneNode.js file. The unimplemented parts of the draw function were to be implemented to accurately display locations of objects and mirroring the transformation of the parent node onto the child nodes such that the transformations were propagated from the parent node to the child node. Particularly, the transformation of the Sun was reflected on Earth and the transformation of the Earth reflected on the Moon.

Implementation

For the transformation of nodes, each node's transformation matrix was gotten from the trs property in the sceneNode.js. From this property, the transformation matrix is available. The matrix contains the translation, rotation and scale values of the node. Every matrix belonging to that node is multiplied by the transformation matrix to get the transformed values.

For the drawing process, the transformed values are used as input for the draw function in meshDrawer. The same transformed values are also applied to the children of that node.

Code:

```
1. var transformMatrix = this.trs.getTransformationMatrix();
2.
3. var transformedMvp       = MatrixMult(mvp, transformMatrix);
4. var transformedModelView = MatrixMult(modelView, transformMatrix);
5. var transformedNormals   = MatrixMult(normalMatrix, transformMatrix);
6. var transformedModel     = MatrixMult(modelMatrix, transformMatrix);
7.
8. // Draw the MeshDrawer
9. if (this.meshDrawer) {
10.     this.meshDrawer.draw(transformedMvp, transformedModelView,
   transformedNormals, transformedModel);
11.     }
12.
13.for (const child of this.children) {
14.     child.draw(transformedMvp, transformedModelView, transformedNormals,
   transformedModel);
15.     }
```

Task 2: Adding Diffuse and Specular Lighting

Objective

The task required updating the fragment shader in meshDrawer.js to calculate diffuse and specular lighting in addition to the existing ambient lighting.

Implementation

For ambient lighting, a default value of 0.35 is set as the ambient light. The light is applied to the texture colors without any further calculation.

For diffuse lighting, the dot product between the light direction and the surface normal was calculated and the maximum value is taken between 0. Afterwards, diffuse light is added with ambient light for multiplication with texture colors.

For specular lighting, A reflection vector is calculated from the direction of the light and the normal. The reflection is used with view direction for the dot product between each other and then raised to the power of a shininess factor, called phongExp, to calculate the specular intensity. As with diffuse and ambient light, specular light is added as well and multiplied with texture colors.

Code:

This is the segment where lighting conditions are calculated in the fragment shader:

```
    vec3 viewDir = normalize(-fragPos); // View direction

    // Diffuse lighting
    diff = max(dot(normal, lightdir), 0.0);
    vec3 diffuseColor = diff * vec3(1.0, 1.0, 1.0);

    // Specular lighting
    vec3 reflectDir = reflect(-lightdir, normal);
    spec = pow(max(dot(reflectDir, viewDir), 0.0), phongExp);
    vec3 specularColor = spec * vec3(1.0, 1.0, 1.0);

    // Texture
    vec4 textureColor = texture2D(tex, vTexCoord);

    // Combine all components
    vec3 finalColor = (ambient + diffuseColor + specularColor) * textureColor.rgb;
    gl_FragColor = vec4(finalColor, textureColor.a);
```

Task 3: Adding Mars to the Scene Graph

Objective

Mars is added as a child of the Sun node, with specified transformations and texture mapping. Specifically, the mars object should clear these criteria:

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

Implementation

A new node instance is created in project3.html for Mars. The mesh is set to a default sphere model with the texture for the Mars object set. Some transformations are applied, the mars object is translated Mars by –6 units along the X-axis relative to the Sun. It is then scaled uniformly by 0.35 in all x,y,z values. Lastly, Mars object is rotated around its Z-axis at 1.5 times the Sun's rotation speed in renderLoop function.

Code:

The created Mars Instance:

```
    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);
```

Applied rotation in renderLoop( ):

```
    marsNode.trs.setRotation(0, 0, zRotation * 1.5);
```

Mars was successfully added to the scene graph as a child of the Sun, with the correct position, scale, rotation, and texture. The final scene displayed the Sun, Earth, Moon, and Mars.
