Yavuz Mehmet Oğuz Şimşek
27920

CS405 Project3 Report

TASK1:

I implemented the draw function by determining the transformation matrix. Since we already have the components translation, rotation and scale, I created a member function called ‘createTransformationFunction’ which basically takes the transformation parameters from this.trs and return combined transformation matrix. Here is the function:

```
createTransformationMatrix(translation, rotation, scale) {
    // Create the translation matrix
    var translateMat = [
        1, 0, 0, translation[0],
        0, 1, 0, translation[1],
        0, 0, 1, translation[2],
        0, 0, 0, 1
    ];

    // Assuming rotation is in radians and around the z-axis
    // For simplicity, only a rotation around the z-axis is considered here
    var angle = rotation[2]; // Assuming rotation[2] is the angle in radians
    var cosA = Math.cos(angle);
    var sinA = Math.sin(angle);
    var rotateMat = [
        cosA, -sinA, 0, 0,
        sinA,  cosA, 0, 0,
        0,       0, 1, 0,
        0,       0, 0, 1
    ];

    // Create the scale matrix
    var scaleMat = [
        scale[0],       0,       0, 0,
              0, scale[1],       0, 0,
              0,       0, scale[2], 0,
              0,       0,       0, 1
    ];

    // Combine the transformations by multiplying the matrices
    var transformMat = this.MatrixMult(rotateMat, this.MatrixMult(scaleMat, translateMat));
    return transformMat;
}
```

And then, I updated the mvp, modelview, normal, model matrices to their transformed formats as follows:

```
var incompleteMatrix = this.createTransformationMatrix(this.trs['translation'],this.trs['rotation'],this.trs['scale']);
var transformationMatrix = this.transposeMatrix(incompleteMatrix);

var transformedMvp = this.MatrixMult(mvp, transformationMatrix);
var transformedModelView = this.MatrixMult(modelView, transformationMatrix);
var transformedNormals = this.MatrixMult(normalMatrix, transformationMatrix);
var transformedModel = this.MatrixMult(modelMatrix, transformationMatrix);
```

While doing that, I took the transpose of the transformation matrix and then multiplied them. Corresponding matrix multiplication and transposematrix function as follows:

Yavuz Mehmet Oğuz Şimşek
27920

```
MatrixMult( A, B )
{
    var C = [];
    for ( var i=0; i<4; ++i ) {
        for ( var j=0; j<4; ++j ) {
            var v = 0;
            for ( var k=0; k<4; ++k ) {
                v += A[j+4*k] * B[k+4*i];
            }
            C.push(v);
        }
    }
    return C;
}

transposeMatrix(matrix) {
    var result = new Array(16);
    for (var row = 0; row < 4; row++) {
        for (var col = 0; col < 4; col++) {
            result[col * 4 + row] = matrix[row * 4 + col];
        }
    }
    return result;
}
```

Finally, I passed the updated variables to meshDrawer draw function and also created a loop for its children which enables its children to be get transformed as well. Final implementation as follows:

```
draw(mvp, modelView, normalMatrix, modelMatrix) {

    var incompleteMatrix = this.createTransformationMatrix(this.trs['translation'],this.trs['rotation'],this.trs['scale']);
    var transformationMatrix = this.transposeMatrix(incompleteMatrix);

    var transformedMvp = this.MatrixMult(mvp, transformationMatrix);
    var transformedModelView = this.MatrixMult(modelView, transformationMatrix);
    var transformedNormals = this.MatrixMult(normalMatrix, transformationMatrix);
    var transformedModel = this.MatrixMult(modelMatrix, transformationMatrix);

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

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

TASK2:

I calculated diffuse value by taking dot product of light direction and normal.
In order to find specular value, I took the dot product of view direction and reflection direction by finding the result value's power of given phong value.
Afterall, resulting implementation as follows:

Yavuz Mehmet Oğuz Şimşek
27920

```
////////////////////////////////////////////////////////////////////////
diff = max(dot(normal, lightdir), 0.0);

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

TASK3:

In order to add mars, Firstly

- I created a new MeshDrawer object, `marsMeshDrawer = new MeshDrawer();`
- Then I configured the Mesh Drawer by establishing buffers, texture and transformations,

```
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);
```

- And finally, I created the mars Scene Node as a child of sun and established the Z-axis rotation,

```
marsNode = new SceneNode(marsMeshDrawer, marsTrs, sunNode);

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

After these steps the view is completed according to the our instructions. Here is the final view: