CSE 3114 / CSE 3219

COMPUTER GRAPHICS

SPRING 2023

Homework #3 Report

Uğurcan Çırak – 190315051

Emine Aydın – 190315053

Submission Date: 17 May 2023

Program Output

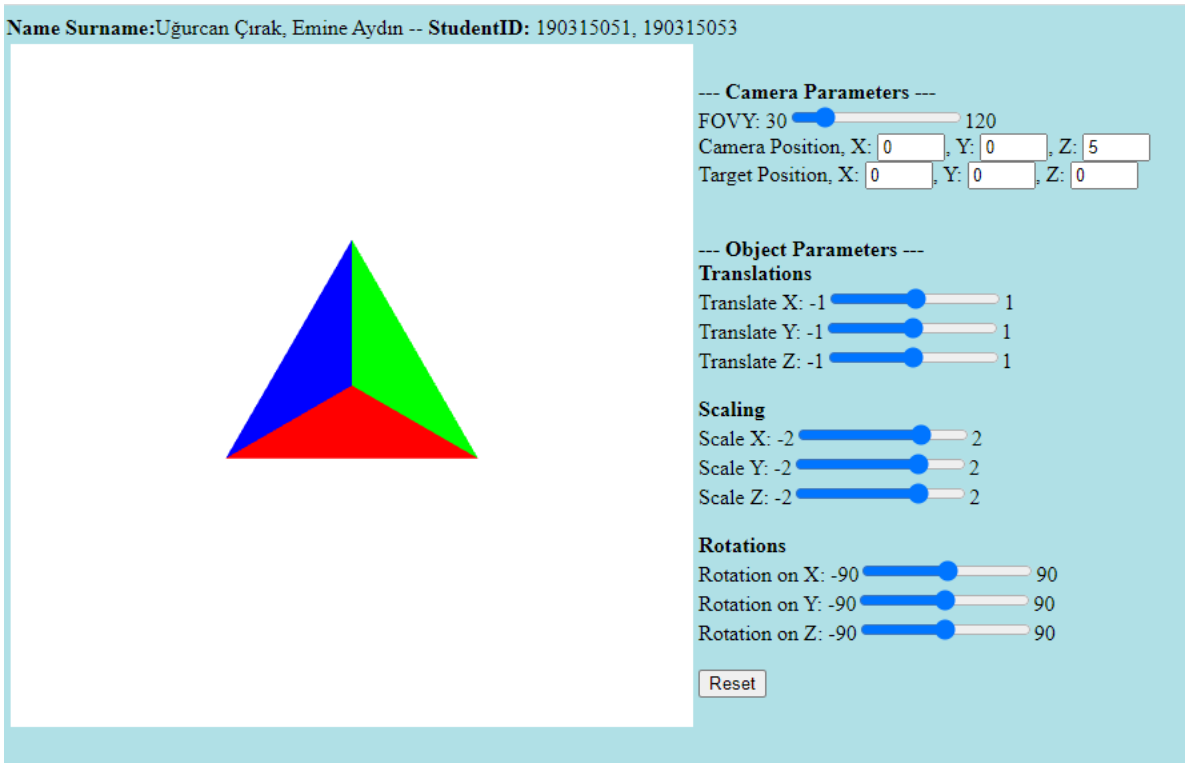

Figure1: *Output-1*

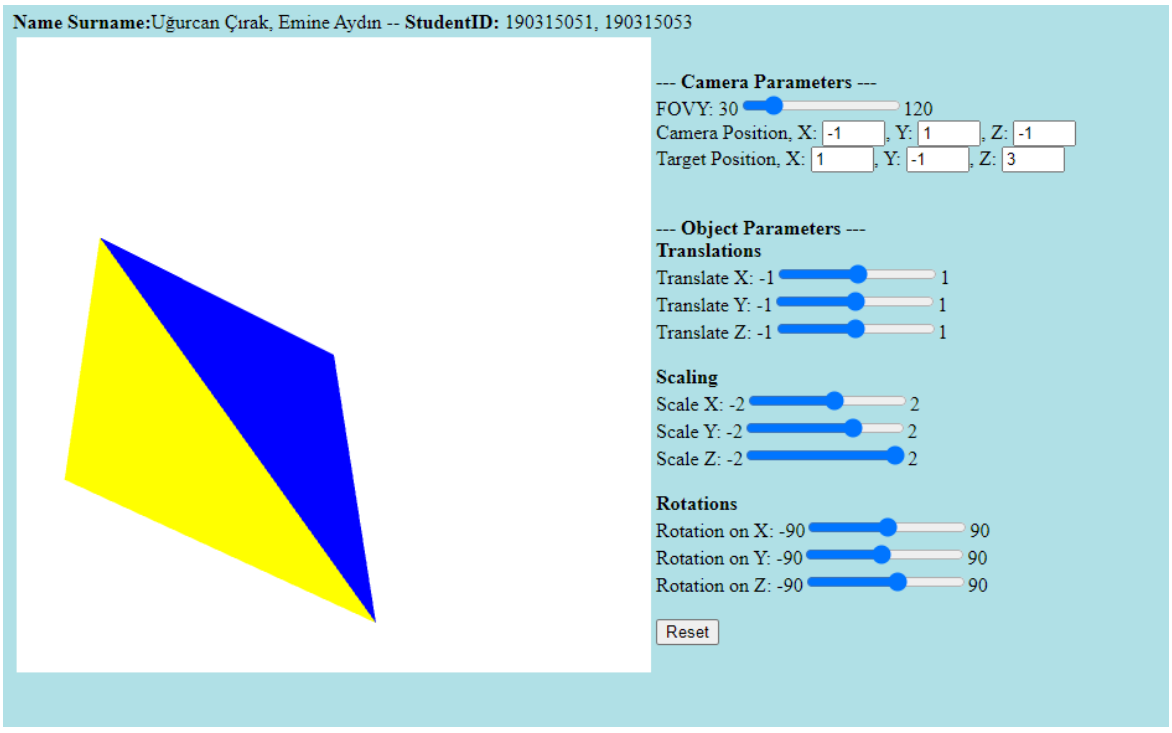

Figure2: *Output-2*

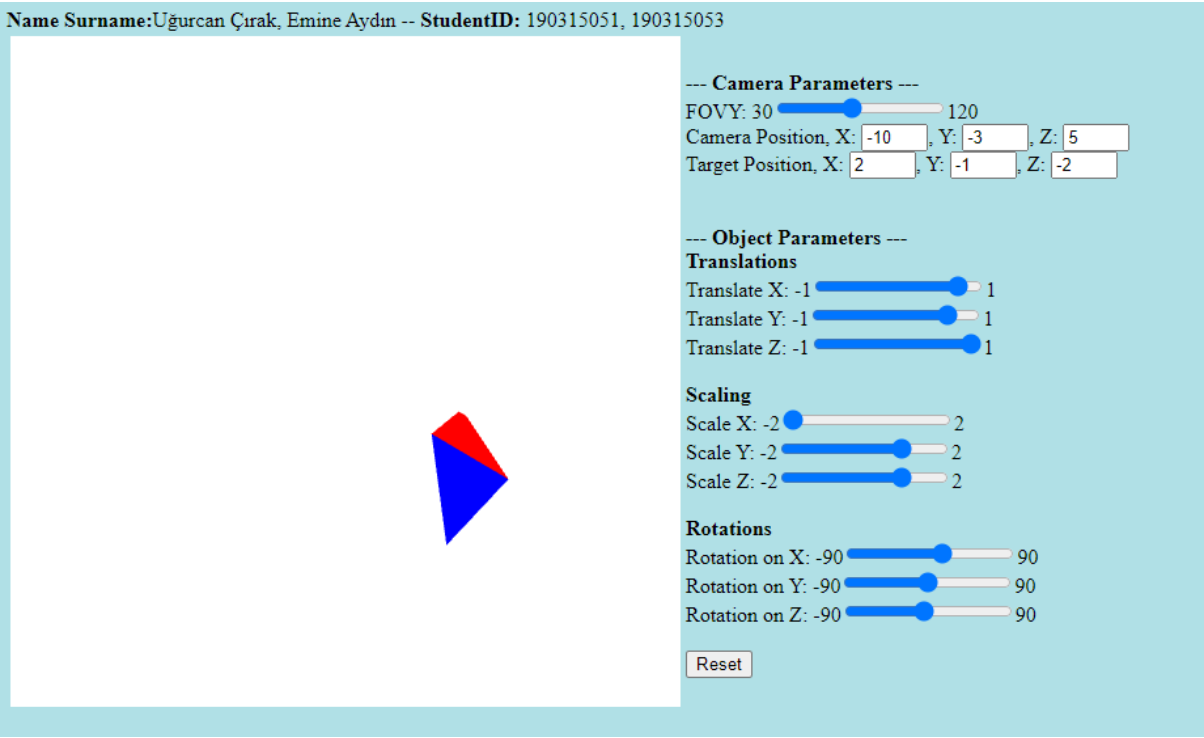

Figure3: *Output-3*

Reflections

In general, we did not experience any difficulties. We had some difficulties writing the modelview and projection matrices, but we learned how to do it. We learned how to write the ***lookAt()*** *and* ***perspective()*** *functions.*

Source Code

HTML Code:

```
<!DOCTYPE html>

<html>

<script id="vertex-shader" type="x-shader/x-vertex">

attribute  vec4 vPosition;

attribute  vec4 vColor;

varying vec4 fColor;
```

```
uniform mat4 modelViewMatrix;

uniform mat4 projectionMatrix;

void main()

{

   gl_Position = projectionMatrix*modelViewMatrix*vPosition;

   fColor = vColor;

}

</script>

<script id="fragment-shader" type="x-shader/x-fragment">

precision mediump float;

varying vec4 fColor;

void main()

{

   gl_FragColor = fColor;

}

</script>

<script type="text/javascript" src="../Common/webgl-utils.js"></script>

<script type="text/javascript" src="../Common/initShaders.js"></script>

<script type="text/javascript" src="../Common/MV.js"></script>

<script type="text/javascript" src="hw3.js"></script>

<body style="background-color:powderblue;">

<div>

<b>Name Surname:</b>Uğurcan Çırak, Emine Aydın -- <b>StudentID:</b> 190315051, 190315053

</div>

<table>
```

```
<tr>
        <td>
        <canvas id="gl-canvas" width="512" height="512">
                Oops ... your browser doesn't support the HTML5 canvas element
        </canvas>
        </td>
        <td>
                <div> <strong>--- Camera Parameters --- </strong></div>
                <div>
                FOVY: 30<input id="fovy" type="range"
                 min="30" max="120" step="5.0" value="45" />120
                </div>
                <div>
                Camera Position, X: <input id="camX" type="number " size="2"
                 value="0" />, Y: <input id="camY" type="number " size="2"
                 value="0" />, Z: <input id="camZ" type="number " size="2"
                 value="5" />
                </div>
                <div>
                Target Position, X: <input id="tarX" type="number " size="2"
                 value="0" />, Y: <input id="tarY" type="number " size="2"
                 value="0" />, Z: <input id="tarZ" type="number " size="2"
                 value="0" />
                </div>
                <br> <br>
                <div> <strong>--- Object Parameters ---</strong></div>
```

```
<div> <strong>Translations</strong> </div>
<div>
Translate X: -1<input id="posX" type="range"
 min="-1" max="1" step="0.1" value="0" />1
</div>
<div>
Translate Y: -1<input id="posY" type="range"
 min="-1" max="1" step="0.1" value="0" />1
</div>
<div>
Translate Z: -1<input id="posZ" type="range"
 min="-1" max="1" step="0.1" value="0" />1
</div>
 <br>
<div> <strong>Scaling</strong> </div>
<div>
Scale X: -2<input id="scaleX" type="range"
 min="-2" max="2" step="0.1" value="1" />2
</div>
<div>
Scale Y: -2<input id="scaleY" type="range"
 min="-2" max="2" step="0.1" value="1" />2
</div>
<div>
Scale Z: -2<input id="scaleZ" type="range"
 min="-2" max="2" step="0.1" value="1" />2
```

```
                    </div>
                    <br>
                    <div> <strong>Rotations</strong> </div>
                    <div>
                    Rotation on X: -90<input id="rotX" type="range"
                     min="-90" max="90" step="5.0" value="0" />90
                    </div>
                    <div>
                    Rotation on Y: -90<input id="rotY" type="range"
                     min="-90" max="90" step="5.0" value="0" />90
                    </div>
                    <div>
                    Rotation on Z: -90<input id="rotZ" type="range"
                     min="-90" max="90" step="5.0" value="0" />90
                    </div>
                    <br>
                    <div>
                    <button id = "ResetButton">Reset</button>
                    </div>
                </td>
            </tr>
    </table>
    <div>
    </body>
    </html>
```

JavaScript Code:

```
var canvas;

var gl;

//initial object transformations

var rotX = rotY = rotZ = 0;

var posX = posY = posZ = 0;

var scaleX = scaleY = scaleZ = 1;

//4 vertices to define tetrahedron corners

var vertices = [

    vec3(  0.0000,  0.0000, 1.0000 ),

        vec3(  0.0000,  0.9428, -0.3333 ),

        vec3( -0.8165, -0.4714, -0.3333 ),

        vec3(  0.8165, -0.4714, -0.3333 )

];


//colors of each tetrahedron corner

var vertexColors = [

    vec4( 0.0, 0.0, 1.0, 1.0 ),  // blue

    vec4( 1.0, 0.0, 0.0, 1.0 ),  // red

    vec4( 1.0, 1.0, 0.0, 1.0 ),  // yellow

    vec4( 0.0, 1.0, 0.0, 1.0 ),  // green

];


//initial camera and view parameters

var near = 0.3; //near clipping plane

var far = 11.0; //far clipping plane
```

```
var eyeX = 0; //camera position x
var eyeY = 0; //camera position y
var eyeZ = 5; //camera position z
var tarX = tarY = tarZ = 0; //camera target (at) position x, y, z
var fovy = 45.0;  // Field-of-view in Y direction angle (in degrees)
const up = vec3(0.0, 1.0, 0.0); //camera up vector
var modelViewMatrix, projectionMatrix;
var modelViewMatrixLoc, projectionMatrixLoc;
var points = [];
var colors = [];
//function that generates tetrahedron geometry
function tetrahedron()
{
   points.push(vertices[0]);
   colors.push(vertexColors[0]);
   points.push(vertices[1]);
   colors.push(vertexColors[0]);
   points.push(vertices[2]);
   colors.push(vertexColors[0]);

   points.push(vertices[3]);
   colors.push(vertexColors[1]);
   points.push(vertices[0]);
   colors.push(vertexColors[1]);
   points.push(vertices[2]);
   colors.push(vertexColors[1]);
```

```
    points.push(vertices[1]);
    colors.push(vertexColors[2]);
    points.push(vertices[2]);
    colors.push(vertexColors[2]);
    points.push(vertices[3]);
    colors.push(vertexColors[2]);

    points.push(vertices[3]);
    colors.push(vertexColors[3]);
    points.push(vertices[1]);
    colors.push(vertexColors[3]);
    points.push(vertices[0]);
    colors.push(vertexColors[3]);
}
window.onload = function init() {
    canvas = document.getElementById( "gl-canvas" );
    gl = WebGLUtils.setupWebGL( canvas );
    if ( !gl ) { alert( "WebGL isn't available" ); }
    gl.viewport( 0, 0, canvas.width, canvas.height );
    aspect =  canvas.width/canvas.height;
    gl.clearColor( 1.0, 1.0, 1.0, 1.0 );
    gl.enable(gl.DEPTH_TEST); //enable depth test for occlusion handling
        tetrahedron();//compute geometry
    //  Load shaders
    var program = initShaders( gl, "vertex-shader", "fragment-shader" );
```

```
gl.useProgram( program );
        //initialize attribute buffers
        var vBuffer = gl.createBuffer();
gl.bindBuffer( gl.ARRAY_BUFFER, vBuffer);
gl.bufferData( gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW );
var vPosition = gl.getAttribLocation( program, "vPosition" );
gl.vertexAttribPointer( vPosition, 3, gl.FLOAT, false, 0, 0 );
gl.enableVertexAttribArray( vPosition );
var cBuffer = gl.createBuffer();
gl.bindBuffer( gl.ARRAY_BUFFER, cBuffer);
gl.bufferData( gl.ARRAY_BUFFER, flatten(colors), gl.STATIC_DRAW );
        var vColor = gl.getAttribLocation( program, "vColor" );
gl.vertexAttribPointer( vColor, 4, gl.FLOAT, false, 0, 0 );
gl.enableVertexAttribArray( vColor);
        // get uniform matrix locations
modelViewMatrixLoc = gl.getUniformLocation( program, "modelViewMatrix" );
projectionMatrixLoc = gl.getUniformLocation( program, "projectionMatrix" );
// sliders for viewing parameters
document.getElementById("fovy").oninput = function(event) {
   //TODO:handle input here
   fovy = event.target.value;
};
        document.getElementById("tarX").onchange = function(event) {
   //TODO:handle input here
   // Update the target position x component
tarX = event.target.value;
```

```
// Recalculate the eye vector
eye = vec3(eyeX, eyeY, eyeZ);
// Recalculate the modelViewMatrix
modelViewMatrix = lookAt(eye, vec3(tarX, tarY, tarZ), up);
};
document.getElementById("tarY").onchange = function(event) {
   //TODO:handle input here
   // Update the target position y component
tarY = event.target.value;
// Recalculate the eye vector
eye = vec3(eyeX, eyeY, eyeZ);
// Recalculate the modelViewMatrix
modelViewMatrix = lookAt(eye, vec3(tarX, tarY, tarZ), up);
};
document.getElementById("tarZ").onchange = function(event) {
   //TODO:handle input here
   // Update the target position z component
tarZ = event.target.value;
// Recalculate the eye vector
eye = vec3(eyeX, eyeY, eyeZ);
// Recalculate the modelViewMatrix
modelViewMatrix = lookAt(eye, vec3(tarX, tarY, tarZ), up);
};
document.getElementById("camX").onchange = function(event) {
   //TODO:handle input here
   // Update the camera position x component
```

```
eyeX = event.target.value;
// Recalculate the eye vector
eye = vec3(eyeX, eyeY, eyeZ);
// Recalculate the modelViewMatrix
modelViewMatrix = lookAt(eye, vec3(tarX, tarY, tarZ), up);
};
document.getElementById("camY").onchange = function(event) {
   //TODO:handle input here
    // Update the camera position y component
eyeY = event.target.value;
// Recalculate the eye vector
eye = vec3(eyeX, eyeY, eyeZ);
// Recalculate the modelViewMatrix
modelViewMatrix = lookAt(eye, vec3(tarX, tarY, tarZ), up);
};
document.getElementById("camZ").onchange = function(event) {
   //TODO:handle input here
    // Update the camera position z component
eyeZ = event.target.value;
// Recalculate the eye vector
eye = vec3(eyeX, eyeY, eyeZ);
// Recalculate the modelViewMatrix
modelViewMatrix = lookAt(eye, vec3(tarX, tarY, tarZ), up);
};
        //sliders for object parameters
        document.getElementById("rotX").oninput = function(event) {
```

```
    //TODO:handle input here
    rotX = event.target.value;
};
document.getElementById("rotY").oninput = function(event) {
    //TODO:handle input here
    rotY = event.target.value;
};
document.getElementById("rotZ").oninput = function(event) {
    //TODO:handle input here
    rotZ = event.target.value;
};
document.getElementById("posX").oninput = function(event) {
    //TODO:handle input here
    posX = event.target.value;
};
document.getElementById("posY").oninput = function(event) {
    //TODO:handle input here
    posY = event.target.value;
};
document.getElementById("posZ").oninput = function(event) {
    //TODO:handle input here
    posZ = event.target.value;
};
        document.getElementById("scaleX").oninput = function(event) {
    //TODO:handle input here
    scaleX = event.target.value;
```

```
};
document.getElementById("scaleY").oninput = function(event) {
   //TODO:handle input here
   scaleY = event.target.value;
};
document.getElementById("scaleZ").oninput = function(event) {
   //TODO:handle input here
   scaleZ = event.target.value;
};
        //reset button callback
        document.getElementById("ResetButton").addEventListener("click", function(){
                //TODO:handle input here
fovy = 45.0;
eyeX = 0;
eyeY = 0;
eyeZ = 5;
tarX = tarY = tarZ = 0;

// Reset object parameters
posX = posY = posZ = 0;
scaleX = scaleY = scaleZ = 1;
rotX = rotY = rotZ = 0;

// Update the UI sliders to their initial values
document.getElementById("fovy").value = fovy;
document.getElementById("tarX").value = tarX;
```

```
    document.getElementById("tarY").value = tarY;
    document.getElementById("tarZ").value = tarZ;
    document.getElementById("camX").value = eyeX;
    document.getElementById("camY").value = eyeY;
    document.getElementById("camZ").value = eyeZ;
    document.getElementById("rotX").value = rotX;
    document.getElementById("rotY").value = rotY;
    document.getElementById("rotZ").value = rotZ;
    document.getElementById("posX").value = posX;
    document.getElementById("posY").value = posY;
    document.getElementById("posZ").value = posZ;
    document.getElementById("scaleX").value = scaleX;
    document.getElementById("scaleY").value = scaleY;
    document.getElementById("scaleZ").value = scaleZ;
    });
    render();
}
var render = function() {
    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
    // Calculate projection matrix (constant)
    projectionMatrix = perspective(fovy, aspect, near, far);
    // Calculate modelview matrix
    var translateMatrix = translate(posX, posY, posZ);
    var rotateMatrix = rotate(rotX, vec3(1, 0, 0));
    rotateMatrix = mult(rotateMatrix, rotate(rotY, vec3(0, 1, 0)));
    rotateMatrix = mult(rotateMatrix, rotate(rotZ, vec3(0, 0, 1)));
```

```
    var scaleMatrix = scalem(scaleX, scaleY, scaleZ);
    var modelViewMatrix = mult(
        lookAt(vec3(eyeX, eyeY, eyeZ), vec3(tarX, tarY, tarZ), up),
        mult(translateMatrix, mult(rotateMatrix, scaleMatrix))
    );
    gl.uniformMatrix4fv(modelViewMatrixLoc, false, flatten(modelViewMatrix));
    gl.uniformMatrix4fv(projectionMatrixLoc, false, flatten(projectionMatrix));
    // Draw the geometry
    gl.drawArrays(gl.TRIANGLES, 0, points.length);
    requestAnimationFrame(render);
}
```