BBM414 Computer Graphics Lab.
Assignment 2 - Report

Berk Bubuş
21945939
Department of Computer Engineering
Hacettepe University
Ankara, Turkey
b21945939@cs.hacettepe.edu.tr

Overview

In this assignment, we have 2 parts. The first part is about color interpolation. The second part has 6 steps about transformation, rotation, and shader animations.

1 Part 1

In Part 1, I changed the triangle function a bit and it worked as I wanted. The function determines which color is chosen and pushes the right color for that point.

```
function triangle( a, b, c, color )
{

    // add colors and vertices for one triangle

    var baseColors = [
        vec3(1.0, 0.0, 0.0),
        vec3(0.0, 1.0, 0.0),
        vec3(0.0, 0.0, 1.0),
        vec3(0.0, 0.0, 0.0)
    ];
    if(color===0){
        colors.push( baseColors[0] );
        points.push( a );
        colors.push( baseColors[1] );
        points.push( b );
        colors.push( baseColors[2] );
        points.push( c );
    }
    else if(color===1){
        colors.push( baseColors[1] );
        points.push( a );
        colors.push( baseColors[2] );
        points.push( b );
        colors.push( baseColors[3] );
        points.push( c );
    }
    else if(color===2){
        colors.push( baseColors[2] );
        points.push( a );
```

```
        colors.push( baseColors[3] );
        points.push( b );
        colors.push( baseColors[0] );
        points.push( c );
    }

}
```

2 Part 2

In part 2, I had written my code step by step. I will explain all functions but as a summary, The first step was about just changing the colors of the snowflake of the previous assignment. About the second step, I had not written it on the vector shader but after the next steps I had to change that version and I had coded it to the vector shader. On the third part, I had not made this part on vector shader as the second part but I have not changed this one. The fourth one was about rolling back all the variables of transformation and rotation. It was the easiest part. The fifth and the sixth part is on shaders.

```
var gl;
var uTheta = 0;
var absTheta;
var thetaSpeed = 1;
var swingMode = false;
var colorSwingMode = false;
var swingDirection = 1;
var basePositionX = 0.0;
var basePositionY = 0.0;
var angle = 0;
var colorSwingColor = [0.5, 0.8, 0.9,1.0];
const numOfComponents = 2;
const normalize =false;
const stride = 0;
const offset = 0;
var level = 4;
```

There are lots of variables that I used lots of part on my project. uTheta is for rotation. absTheta is for changing color according to uTheta. thetaSpeed is the changing speed of the angle. swingMode is for swinging animation. swingDirection is for changing of angle which is negative or positive. basePositionX and basePositionY are for transformation. angle is the base angle and it is constant. colorSwingColor changes if we press '3'.

```
window.onload = function init() {

    const canvas = document.querySelector("#glCanvas");
    gl = canvas.getContext("webgl2");
    document.addEventListener("keydown", function (event) {
        if (event.key === "ArrowRight") {
            basePositionX += 0.01;
        } else if (event.key === "ArrowLeft") {
            basePositionX -= 0.01;
        } else if (event.key === "ArrowUp") {
            basePositionY += 0.01;
        } else if (event.key === "ArrowDown") {
            basePositionY -= 0.01;
        } else if (event.key === "-") {
            angle += 1;
        } else if (event.key === "+") {
            angle -= 1;
```

```
        } else if (event.key === "1") {
            basePositionX = 0.0;
            basePositionY = 0.0;
            angle = 0;
            uTheta = 0;
        } else if (event.key === "2") {
            swingMode = true;
        } else if (event.key === "3") {
            colorSwingMode = true;
        }
    });
    if(!gl){
        alert("Unable to initialize WebGL2. Your browser or machine
        may not support it.");
        return;
    }
    draw();
}
```

This init() function works on load and event listeners remain work. There are all the key management. And it calls draw() at the end.

```
function draw(){
    gl.clearColor(0.9,0.9,0.9,1.0);
    gl.clear(gl.COLOR_BUFFER_BIT);

    const program1 = initShaderProgram(gl, vsSource, fsSource1);
    const program2 = initShaderProgram(gl, vsSource, fsSource2);

    var thetaLoc1 = gl.getUniformLocation(program1, "uTheta");
    var thetaLoc2 = gl.getUniformLocation(program2, "uTheta");
    var posXLoc1 = gl.getUniformLocation(program1, "posX");
    var posXLoc2 = gl.getUniformLocation(program2, "posX");
    var posYLoc1 = gl.getUniformLocation(program1, "posY");
    var posYLoc2 = gl.getUniformLocation(program2, "posY");
    var colorLoc = gl.getUniformLocation(program1, "absTheta");

    var positions = makeSnowflake(level,-0.15,-0.075,0.15,angle);
    const buffer = initBuffer(gl, positions);
    gl.useProgram(program1);
    gl.uniform1f(posXLoc1, basePositionX);
    gl.uniform1f(posYLoc1, basePositionY);
    gl.uniform1f(thetaLoc1, uTheta);
    if(colorSwingMode){
        gl.uniform1f(colorLoc, Math.abs(uTheta));
    }
    else{
        gl.uniform1f(colorLoc, 0);
    }

    gl.enableVertexAttribArray(gl.getAttribLocation(program1, "aPosition"));
    gl.bindBuffer(gl.ARRAY_BUFFER, buffer.position);
    gl.vertexAttribPointer(gl.getAttribLocation(program1, "aPosition"),
    numOfComponents,gl.FLOAT,normalize,stride,offset);
    gl.drawArrays(gl.TRIANGLES, offset, positions.length/2);

    var positions2 = makeSnowflake(level,-0.10,-0.05,0.10,angle);
    const buffer2 = initBuffer(gl, positions2);
    gl.useProgram(program2);
    gl.uniform1f(posXLoc2, basePositionX);
    gl.uniform1f(posYLoc2, basePositionY);
    gl.uniform1f(thetaLoc2, uTheta);
    gl.enableVertexAttribArray(gl.getAttribLocation(program2, "aPosition"));
```

```
    gl.bindBuffer(gl.ARRAY_BUFFER, buffer2.position);
    gl.vertexAttribPointer(gl.getAttribLocation(program2, "aPosition"),
    numOfComponents,gl.FLOAT,normalize,stride,offset);
    gl.drawArrays(gl.TRIANGLES, offset, positions2.length/2);

    if(swingMode || colorSwingMode){
        if(swingMode){
            document.addEventListener("keydown", function (event) {
                if (event.key !== "2") {
                    swingMode = false;
                    uTheta = 0;
                }
            });
        }
        else{
            document.addEventListener("keydown", function (event) {
                if (event.key !== "3") {
                    colorSwingMode = false;
                    uTheta = 0;
                }
            });
        }
        if(uTheta >= 80){
            swingDirection = -1;
        }
        else if(uTheta <= -80){
            swingDirection = 1;
        }
        uTheta += swingDirection*thetaSpeed;
    }
    requestAnimationFrame(draw);
}
```

Lots of the parts of this function from the previous assignment. There are some uniforms about the rotation and coloring. It controls swingMode and colorSwingMode and manages swingDirection and uTheta. And it also adds new event listeners to close swing modes. At last, it calls requestAnimationFrame(draw) to loop itself.

```
const vsSource = `#version 300 es
    #define PI 3.1415926538
    in vec4 aPosition;

    uniform float uTheta;
    uniform float posX;
    uniform float posY;

    void main()
    {
        float angle = uTheta*PI/180.0;
        float c = cos(angle);
        float s = sin(angle);

        mat4 rz = mat4(c, s, 0.0, 0.0,
                       -s,  c, 0.0, 0.0,
                       0.0,  0.0, 1.0, 0.0,
                       posX,  posY, 0.0, 1.0);


        gl_Position = rz * aPosition;

        gl_Position.z = -gl_Position.z;
    }
`;
```

There are uniform variables for transformation and rotation. In main(), we have a matrix that is 4x4, it has got needed transformations and rotations in it.

```
const fsSource1 =  `#version 300 es
    precision mediump float;
    uniform float absTheta;
    out vec4 fColor;
    void main(){
        fColor = vec4(0.5-absTheta/250.0, 0.8
        -absTheta/154.0, 0.9-absTheta/138.0,1.0);
    }
`;

const fsSource2 =  `#version 300 es
    precision mediump float;
    out vec4 fColor;
    void main(){
        fColor = vec4(0.9,0.9,0.9,1.0);
    }
`;
```

I have 2 fsSource on my project. fsSource1 is for the blue snowflake. fsSource2 is for the white one. fsSource1 also has got color-changing parts. I calculated the necessary float numbers for creating the right color.