Laporan Tugas Grafika Komputer – WebGL How It Works

Clarisa Natalia Edelin

13519213

Modul WebGL How It Works menjelaskan mengenai cara program WebGL dan GPU meggambarkan sebuah bentuk yang kita inginkan dari titik koordinat yang kita berikan di dalam program. Titik koordinat yang ada di dalam program kita akan dimasukkan ke dalam sebuah array yang kana diproses menjadi clip space vertices. Selanjutnya program akan mulai menggambarkan bentuk dalam pixel sesuai dengan titik koordinat yang tadi kita berikan.

Pada file HTML, kita mendeklarasikan sebuah vertex shader dan fragment shader.

```
<script  id="vertex-shader-2d" type="x-shader/x-vertex">
      attribute vec2 a_position;
      attribute vec4 a_color;

      uniform mat3 u_matrix;

      varying vec4 v_color;

      void main() {
        // Multiply the position by the matrix.
        gl_Position = vec4((u_matrix * vec3(a_position, 1)).xy,
      0, 1);

        // Copy the color from the attribute to the varying.
        v_color = a_color;
      }
</script>
```

Vertex shader digunakan untuk memberikan informasi mengenai vertex yang akan kita gunakan, seperti posisi dan warna.

```
<script  id="fragment-shader-2d" type="x-shader/x-fragment">
      precision mediump float;

      varying vec4 v_color;
```

```
      void main() {
        gl_FragColor = v_color;
      }
</script>
```

Fragment shader digunakan menentukan warna yang ada diantra titik yang telah ditentukan pada vertex shader, dimana hal tersebut ditentukan dengan adanya komponen warna pada bagian deklarasi fragment shader.

```
<canvas id="canvas"></canvas>
<div id="uiContainer">
        <div id="ui">
          <div id="x"></div>
          <div id="y"></div>
          <div id="angle"></div>
          <div id="scaleX"></div>
          <div id="scaleY"></div>
        </div>
</div>
```

Gambar yang dibuat oleh program kita akan dibuat pada tag canvas pada file HTML. Selain itu juga terdapat tombol – tombol yang digunakan untuk mengubah transformasi bentuk pada canvas.

File Javascript digunakan untuk menjalankan proses pembuatan bentuk ataupun perubahan yang akan dilakukan pada bentuk di canvas pada file HTML.

```
var canvas = document.querySelector("#canvas");
```

Selain itu, kita juga akan membuat variabel yang mengandung informasi mengenai posisi dan warna dari bentuk yang akan kita gambar.

```
var    positionLocation    =    gl.getAttribLocation(program,
"a_position");
  var    colorLocation    =    gl.getAttribLocation(program,
"a_color");
```

Untuk dapat menggunakan informasi mengenai posisi dan warna, kita perlu membuat sebuah buffer untuk informasi tersebut. Lalu kita menghubungkan variabel yang tadi kita buat dengan array buffer pada program.

```
var positionBuffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, positionBuffer);
setGeometry(gl);

var colorBuffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, colorBuffer);
setColors(gl);
```

Fungsi setGeometry merupakan sebuah fungsi dimana kita meletakkan informasi mengenai posisi titik yang akan membuat persegi panjang yang kita inginkan. Titik koordinat tersebut dimasukkan dalam sebuah array of 32 bit dan akan digambarkan dengan menggunakan static draw.

```
function setGeometry(gl) {
          gl.bufferData(
              gl.ARRAY_BUFFER,
              new Float32Array([
                  -150, -100,
                   150, -100,
                  -150,  100,
                   150, -100,
                  -150,  100,
                   150,  100]),
              gl.STATIC_DRAW);
        }
```

Sedagkan fungsi SetColor merupakan sebuah fungsi yang memberikan informasi warna apa yang kita inginkan untuk persegi panjang yang kita buat. Hampir sama dengan setGeometry, namun pada setColor kita memasukkan skala RGB untuk warna yang kita inginkan untuk setiap titik koordinat.

```
        function setColors(gl) {
          // Pick 2 random colors.
          var r1 = Math.random();
          var b1 = Math.random();
```

```
        var g1 = Math.random();
        var r2 = Math.random();
        var b2 = Math.random();
        var g2 = Math.random();

        gl.bufferData(
            gl.ARRAY_BUFFER,
            new Float32Array(
              [ r1, b1, g1, 1,
                r1, b1, g1, 1,
                r1, b1, g1, 1,
                r2, b2, g2, 1,
                r2, b2, g2, 1,
                r2, b2, g2, 1]),
            gl.STATIC_DRAW);
    }
```

Pada modul ini, kita membuat sebuah program yang dapat menggambarkan sebuah persegi panjang dan mengubah ukuran dan posisinya sesuai dengan input pengguna yang dapat mengubah posisi baik dari sumbu x, y, dan z, sudut kita melihat persegi panjang tersebut, dan skala ukuran dan persegi panjang yang kita buat. Hal pertama yang haru skita lakukan adalah membuat sebuah variabel terkait dengan translasi, sudut, dan skala awal persegi panjang yang akan ditampilkan.

```
        var translation = [200, 150];
        var angleInRadians = 0;
        var scale = [1, 1];
```

Untuk dapat mengubah posisi, sudut penglihatan, dan skala persegi panjang, kita memerlukan fungsi yang dapat menangani hal tersebut.

```
    function updatePosition(index) {
        return function(event, ui) {
          translation[index] = ui.value;
          drawScene();
        };
      }

      function updateAngle(event, ui) {
```

```
            var angleInDegrees = 360 - ui.value;
            angleInRadians = angleInDegrees * Math.PI / 180;
            drawScene();
          }

          function updateScale(index) {
            return function(event, ui) {
              scale[index] = ui.value;
              drawScene();
            };
          }
```

Fungsi dracScene merupakan sebuah fungsi yang digunakan untuk menggambar persegi panjang ke dalam canvas.

```
          function drawScene() {
            webglUtils.resizeCanvasToDisplaySize(gl.canvas);

            gl.viewport(0,         0,         gl.canvas.width,
      gl.canvas.height);

            gl.clearColor(0.5, 0.6, 0.6, 1.0);
            gl.clear(gl.COLOR_BUFFER_BIT);

            gl.useProgram(program);

            gl.enableVertexAttribArray(positionLocation);

            gl.bindBuffer(gl.ARRAY_BUFFER, positionBuffer);

            var size = 2;              // 2 components per
      iteration
            var type = gl.FLOAT;     // the data is 32bit
      floats
            var normalize = false; // don't normalize the
      data
            var stride = 0;        // 0 = move forward size *
      sizeof(type) each iteration to get the next position
            var offset = 0;         // start at the beginning
      of the buffer
            gl.vertexAttribPointer(
```

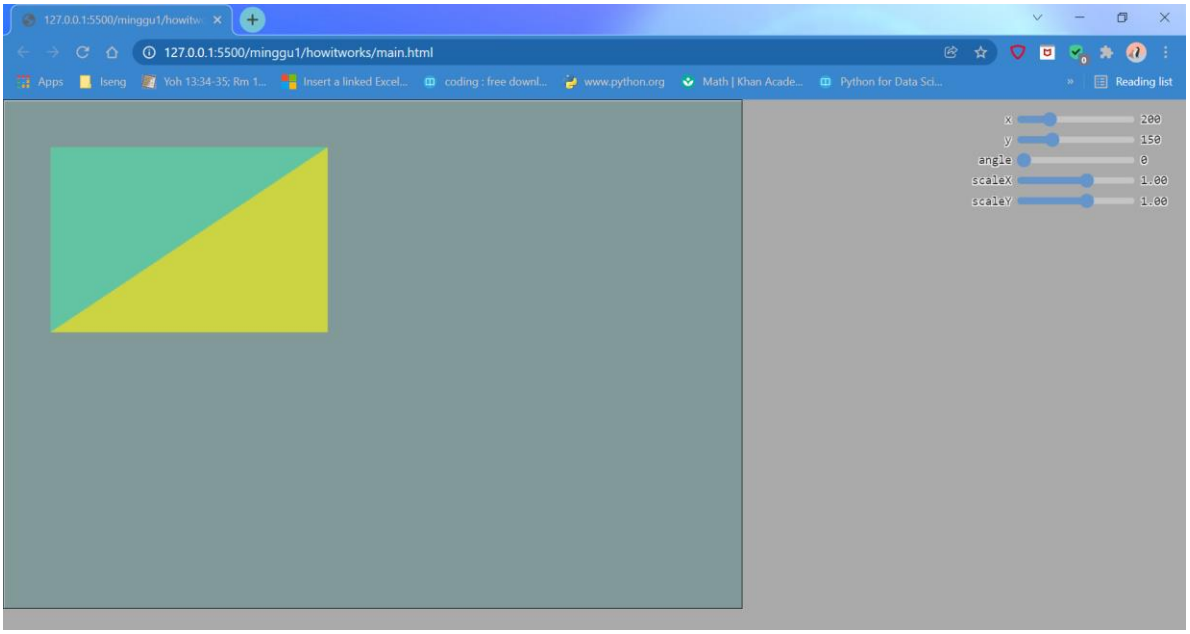
```
            positionLocation, size, type, normalize,
    stride, offset);

        gl.enableVertexAttribArray(colorLocation);

        gl.bindBuffer(gl.ARRAY_BUFFER, colorBuffer);

        var size = 4;              // 4 components per
    iteration
        var type = gl.FLOAT;     // the data is 32bit
    floats
        var normalize = false; // don't normalize the
    data
        var stride = 0;        // 0 = move forward size *
    sizeof(type) each iteration to get the next position
        var offset = 0;         // start at the beginning
    of the buffer
        gl.vertexAttribPointer(
            colorLocation, size, type, normalize, stride,
    offset);

        var matrix = m3.projection(gl.canvas.clientWidth,
    gl.canvas.clientHeight);
        matrix = m3.translate(matrix, translation[0],
    translation[1]);
        matrix = m3.rotate(matrix, angleInRadians);
        matrix = m3.scale(matrix, scale[0], scale[1]);

        gl.uniformMatrix3fv(matrixLocation,         false,
    matrix);

        var primitiveType = gl.TRIANGLES;
        var offset = 0;
        var count = 6;
        gl.drawArrays(primitiveType, offset, count);
      }
    }
```

Pada fungsi drawScene, kita perlu memberutahu WebGL cara untuk mengubah clip space menjadi pixel pada canvas, yaitu dengan menggunakan viewport. Setelah itu, kita akan ‘membersihkan’ canvas tempat bentuk akan

digambarkan dengan menggunakan clearColor untuk menentukan warna canvas saat bersih dan clear pada color buffer bit. Kita akan menggunakan useProgram untuk menggunakan Program yang telah kita buat sebelumnya dengan menggunakan useProgram. Bagian selanjutnya adalah bagian dimana program akan menggambarkan persegi panjang, dimulai dari menggambarkan titik sesuai dengan informasi yang kita sediakan sampai dengan pemberian warna pada bentuk yang dihasilkan oleh kumpulan titik tersebut. Namun, kita juga perlu memberikan informasi terkait bagaimana cara program mendapatkan data dari positionBuffer dan colorBuffer dengan membuat beberapa variabel seperti size, type, normalize, stride, dan offset, lalu menggunakan vertexAttribPointer untuk mengambil data yang diperlukan.

Untuk dapat mengubah baik ukuran, sudut penglihatan, dan skala persegi panjang yang kita buat, kita memerlukan matriks, dimana perubahan yang terjadi pada bentuk yang kita gambar dapat dilakukan dengan perkalian matriks transformasi. Kita membuat satu matriks untuk setiap kategori perubahan, yaitu translasi, rotasi, dan skala. Langkah terakhir adalah membuat program untuk menggambar persegi panjang tersebut dengan menggunakan drawArrays, dengan primitiveType berupa triangles, offset sebesar 0, dan count sebesar 6. Tidak lupa kita memanggil fungsi main untuk menjalankan program pada file Javascript.

Berikut ini merupakan foto hasil program dijalankan.



Link video Youtube : https://youtu.be/eyJJ4IArXNk

Link repository Git : https://github.com/ClarisaNatalia/WebGL-HowItWorks