Laporan Tugas 1

IF 3260 Grafika Komputer

Akeyla Pradia Naufal

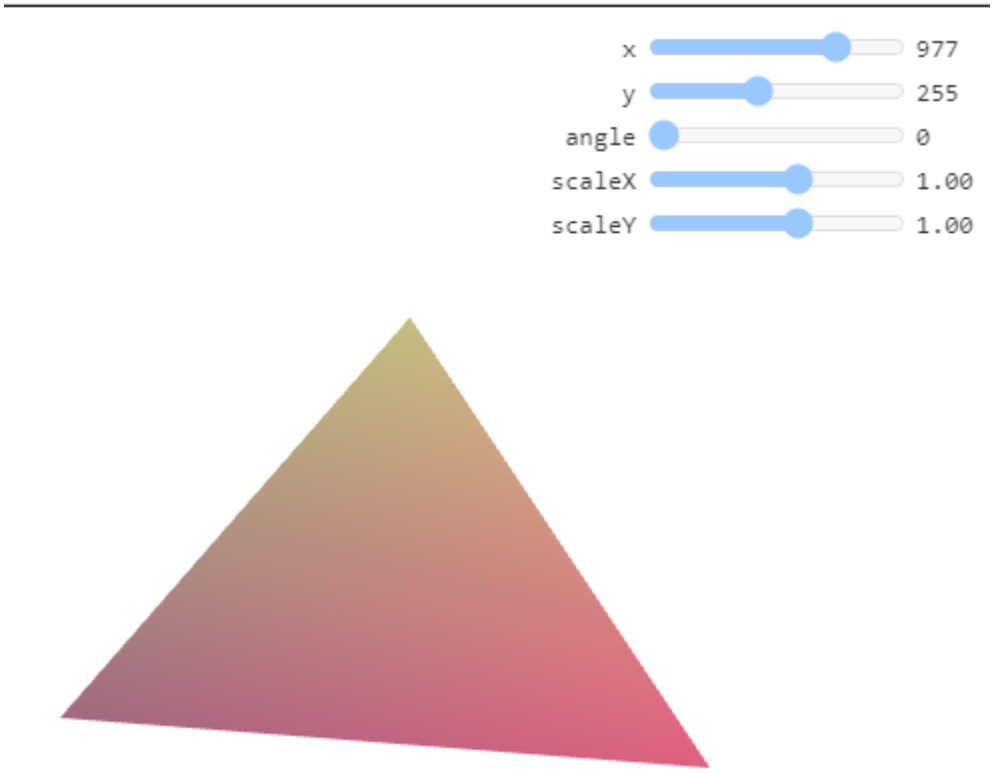
13519178

Bagian 1. *How It Works*

Secara garis besar, program bagian ini bertujuan untuk menampilkan segitiga dan persegi panjang di layar dengan pewarnaannya menggunakan *varying* yang berupa vec4 untuk memberikan efek gradasi.

Program 1

Program ini menampilkan sebuah segitiga dan beberapa menu slider yang dapat mengatur properti-properti dari segitiga tersebut seperti koordinat x, koordinat y, sudut rotasi, skala X, dan skala Y.



Bentuk bangun datar segitiga ini di awal diatur di fungsi `setGeometry(gl)`. Data mengenai koordinat x dan y dari titik-titik sudut segitiga dimasukkan ke dalam ARRAY_BUFFER, berupa float32, dan bersifat STATIC_DRAW yang berarti render gambar hanya dilakukan sekali.

```
function setGeometry(gl){
    gl.bufferData(
        gl.ARRAY_BUFFER,
        new Float32Array([
            0, -100,
            150, 125,
            -175, 100
        ]),
        gl.STATIC_DRAW
    );
}
```

Data mengenai posisi titik sudut segitiga ini diambil dari ARRAY_BUFFER dan dimasukkan ke dalam positionBuffer dengan aturan pembacaan yang ditunjukkan pada kode berikut:

```
// Tell the attribute how to get data out of position buffer (ARRAY_BUFFER)
var size = 2;       // 2 components per iteration
var type = gl.FLOAT; // the data is 32 bit float
var normalized = false; // dont 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 f the buffer
gl.vertexAttribPointer(positionAttributeLocation, size, type, normalized, stride, offset);
```

Menampilkan gambar tersebut di layar dilakukan oleh kode berikut. primitiveType menunjukkan bentuk yang dibentuk dari titik-titik yang ada dan count menunjukkan banyak titik yang ada.

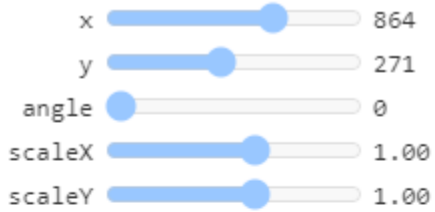
```
// Draw the geometry
var primitiveType = gl.TRIANGLES;
var offset = 0;
var count = 3;
gl.drawArrays(primitiveType, offset, count);
```

Untuk menampilkan tampilan segitiga setelah ditransformasikan, dilakukan perhitungan dengan menggunakan matriks berukuran 3 x 3 yang operasinya berasal dari modul m3 pada tutorial ini.

```
// Compute the matrix
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]);
```

Program 2

Program ini menampilkan dua buah segitiga yang membentuk sebuah persegi panjang dengan dua warna berbeda. Tetap ada slider seperti di program sebelumnya.



Mirip seperti sebelumnya, posisi awal persegi panjang ini dibangun di fungsi setGeometry(gl). Kali ini, terdapat enam pasangan koordinat yang menunjukkan bahwa persegi panjang ini dibangun dari dua buah segitiga, di mana dua pasang titiknya berhimpit.

```
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
    );
}
```

Akibatnya, tentu saja, variabel count di drawArrays dinaikkan menjadi 6.

```
// Draw the geometry
var primitiveType = gl.TRIANGLES;
var offset = 0;
var count = 6;
gl.drawArrays(primitiveType, offset, count);
```

Untuk mengatur warna, di kedua segitiga ini, pertama ditentukan warna untuk masing-masing segitiga terlebih dahulu dengan fungsi setColors(gl).

```
function setColors(gl){
    // Pick 2 random colors
    var r1 = Math.random();
    var g1 = Math.random();
    var b1 = Math.random();
    var r2 = Math.random();
    var g2 = Math.random();
    var b2 = Math.random();

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

Dan seperti halnya posisi, nilai warna ini disimpan di ARRAY_BUFFER dan akan dibaca dengan aturan atribut sebagai berikut. Size bernilai 4 karena terdapat empat komponen di warna: R, G, B, dan alpha.

```
// Tell the color attribute how to get data out of color buffer
var size = 4;               // 4 components per iteration
var type = gl.FLOAT;        // The data is 32 bit floats
var normalized = 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, normalized, stride, offset);
```

Program 3

Program 3 hampir sama dengan program 2. Perbedaan utama terletak pada warna segitiga yang bergradasi. Hal ini disebabkan adanya interpolasi oleh program untuk menyesuaikan warna di interior segitiga agar selaras dengan warna ketiga titik sudut segitiga. Hal ini dapat dicapai dengan mengeset warna ketiga titik segitiga secara acak satu sama lain, ditunjukkan pada potongan kode berikut.

```
function setColors(gl){
    // Make every vertex different colors
    gl.bufferData(
        gl.ARRAY_BUFFER,
        new Float32Array(
            [
                Math.random(), Math.random(), Math.random(), 1,
                Math.random(), Math.random(), Math.random(), 1,
                Math.random(), Math.random(), Math.random(), 1,
                Math.random(), Math.random(), Math.random(), 1,
                Math.random(), Math.random(), Math.random(), 1,
                Math.random(), Math.random(), Math.random(), 1
            ]
        ),
        gl.STATIC_DRAW
    );
}
```

Bagian 2. *Image Processing*

Program 1

Program ini menampilkan sebuah gambar. Penampilan gambar ini dapat dicapai dengan mengesetnya sebagai tekstur dari sebuah persegi panjang.

Pembuatan tekstur ini dilakukan dengan fungsi gl.createTexture. Pengaturan mengenai bagaimana tekstur tersebut ditampilkan terlihat pada potongan kode di bawah. Bagian gl.texParameteri yang memuat gl.TEXTURE_WRAP_S dan gl.TEXTURE_WRAP_T mengatur tampilan koordinat s dan t agar tekstur "ditarik" hingga ke ujung window apabila masih ada ukuran tekstur masih lebih kecil. Bagian yang memuat gl.TEXTURE_MIN_FILTER dan gl.TEXTURE_MAG_FILTER mengatur tampilan tekstur ketika tekstur diperbesar atau diperkecil. Di sini, digunakan gl.NEAREST untuk membuat piksel tekstur menjadi sama dengan piksel tekstur terdekatnya.

```
// Create a texture
var texture = gl.createTexture();
gl.bindTexture(gl.TEXTURE_2D, texture);

// Set the parameters so we can render any size image
gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_WRAP_S, gl.CLAMP_TO_EDGE);
gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_WRAP_T, gl.CLAMP_TO_EDGE);
gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.NEAREST);
gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.NEAREST);

// Upload the image into the texture
gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_BYTE, image);
```

Seperti halnya warna, pengambilan data tekstur diatur dengan fungsi gl.vertexAttribPointer.

```
// Tell the texcoord attribute how to get data out of texcoordBuffer
var size = 2;
var type = gl.FLOAT;
var normalized = false;
var stride = 0;
var offset = 0;
gl.vertexAttribPointer(texcoordLocation, size, type, normalized, stride, offset);
```

Program 2

Program ini hampir sama seperti program sebelumnya. Perbedaan hanya pada warna yang berubah dari merah menjadi biru. Hal ini diatur dengan menambahkan .bgra pada tekstur yang diatur dengan bahasa GLSL yang ada di file HTML.

```
void main(){
    gl_FragColor = texture2D(u_image, v_texCoord).bgra;
}
```

Program 3

Pada program ini, gambar ditampilkan secara buram dengan mengatur tampilan semua piksel yang ada menjadi rata-rata piksel yang sebaris dan bersebelahan langsung dengan piksel tersebut.

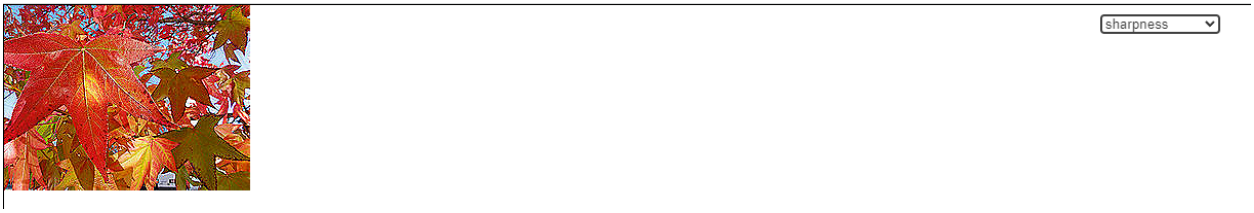
Hal ini dapat dicapai dengan mengatur gl_FragColor sebagai vektor yang isi pikselnya merupakan rata-rata dari piksel sebelah kiri dan kanannya serta dari piksel semula.

```
void main(){
    // Compute 1 pixel in texture coordinates
    vec2 onePixel = vec2(1.0, 1.0) / u_textureSize;

    // Average the left, middle, and right pixels
    gl_FragColor = (
        texture2D(u_image, v_texCoord) +
        texture2D(u_image, v_texCoord + vec2(onePixel.x, 0.0)) +
        texture2D(u_image, v_texCoord + vec2(-onePixel.x, 0.0))
    ) / 3.0;
```

Program 4

Program 4 menambahkan lebih banyak pilihan cara memburamkan gambar. Pengguna kini dapat memilih cara memburamkan gambar tersebut. Hal ini dapat dicapai dengan mengeset matriks yang berguna sebagai kernel konvolusi dari gambar.



Beberapa matriks yang dipakai ditunjukkan pada potongan kode berikut.

```
// Define several convolution kernels
var kernels = {
  normal: [
    0, 0, 0,
    0, 1, 0,
    0, 0, 0
  ],
  gaussianBlur: [
    0.045, 0.122, 0.045,
    0.122, 0.332, 0.122,
    0.045, 0.122, 0.045
  ],
  gaussianBlur2: [
    1, 2, 1,
    2, 4, 2,
    1, 2, 1
  ],
  gaussianBlur3: [
    0, 1, 0,
    1, 1, 1,
    0, 1, 0
  ],
  unsharpen: [
    -1, -1, -1,
    -1,  9, -1,
    -1, -1, -1
  ],
  sharpness: [
     0,-1, 0,
    -1, 5,-1,
     0,-1, 0
  ],
```

Perhitungan dilakukan dengan menggunakan fungsi berikut.

```
function computeKernelWeight(kernel) {
  var weight = kernel.reduce(function(prev, curr) {
      return prev + curr;
  });
  return weight <= 0 ? 1 : weight;
}
```

Link

Link github:

https://github.com/anakpindahan/Tutorial-WebGL

Link youtube:

https://youtu.be/0h_4rsiw9iI