Nama : Muhammad Dehan Al Kautsar
NIM : 13519200
Hands-On IF3260 Grafika Komputer - Image Processing

Pada tutorial yang terdapat di laman web https://webglfundamentals.org/webgl/lessons/webgl-image-processing.html, laman tersebut menjelaskan bagaimana kita dapat menampilkan gambar yang ingin kita tampilkan pada canvas browser dan mengedit gestur serta tekstur dari gambar yang kita tampilkan.

Pada laporan kali ini, saya akan melakukan *highlight* apa yang menjadi hal penting yang ditampilkan pada tutorial pada laman tersebut. Hal yang saya dapatkan akan dibahas di bawah paragraf ini.

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

uniform vec2 u_resolution;

varying vec2 v_texCoord;

void main() {
   // convert the rectangle from pixels to 0.0 to 1.0
   vec2 zeroToOne = a_position / u_resolution;

   // convert from 0->1 to 0->2
   vec2 zeroToTwo = zeroToOne * 2.0;

   // convert from 0->2 to -1->+1 (clipspace)
   vec2 clipSpace = zeroToTwo - 1.0;

   gl_Position = vec4(clipSpace * vec2(1, -1), 0, 1);

   // pass the texCoord to the fragment shader
   // The GPU will interpolate this value between points.
   v_texCoord = a_texCoord;
}
</script>
<!-- fragment shader -->
<script  id="fragment-shader-2d" type="x-shader/x-fragment">
precision mediump float;

// our texture
uniform sampler2D u_image;
uniform vec2 u_textureSize;
uniform float u_kernel[9];
uniform float u_kernelWeight;

// the texCoords passed in from the vertex shader.
varying vec2 v_texCoord;

void main() {
   vec2 onePixel = vec2(1.0, 1.0) / u_textureSize;
   vec4 colorSum =
       texture2D(u_image, v_texCoord + onePixel * vec2(-1, -1)) * u_kernel[0] +
       texture2D(u_image, v_texCoord + onePixel * vec2( 0, -1)) * u_kernel[1] +
       texture2D(u_image, v_texCoord + onePixel * vec2( 1, -1)) * u_kernel[2] +
       texture2D(u_image, v_texCoord + onePixel * vec2(-1,  0)) * u_kernel[3] +
       texture2D(u_image, v_texCoord + onePixel * vec2( 0,  0)) * u_kernel[4] +
       texture2D(u_image, v_texCoord + onePixel * vec2( 1,  0)) * u_kernel[5] +
       texture2D(u_image, v_texCoord + onePixel * vec2(-1,  1)) * u_kernel[6] +
       texture2D(u_image, v_texCoord + onePixel * vec2( 0,  1)) * u_kernel[7] +
       texture2D(u_image, v_texCoord + onePixel * vec2( 1,  1)) * u_kernel[8] ;

   gl_FragColor = vec4((colorSum / u_kernelWeight).rgb, 1);
}
</script>
```

Script pada HTML di atas merupakan *remake* dari apa yang telah dikerjakan pada tutorial sebelumnya: “How It Works”. Perbedaan yang ada dari kedua tutorial tersebut adalah dengan ditambahkannya variabel v_texCoord yang akan di-passing ke fragment shader di bawahnya. Kemudian

untuk melihat warna dari tekstur image yang kita ingin tampilkan, pada script fragment shader digunakan fungsi texture2D().

```
function main() {
  var image = new Image();
  image.src = "https://webglfundamentals.org/webgl/resources/leaves.jpg";  //
  image.crossOrigin = "anonymous";      You, 3 days ago • feat: image proces
  image.onload = function() {
    render(image);
  };
}
```

Hal yang selanjutnya dilakukan adalah melakukan loading image dari gambar apa yang ingin ditampilkan. Namun pada tutorial tidak ada crossOrigin sehingga saat ditampilkan pada browser ia masih error. Ketika ditambahkan barulah program dapat menampilkan gambar.

```
// look up where the vertex data needs to go.
var positionLocation = gl.getAttribLocation(program, "a_position");
var texcoordLocation = gl.getAttribLocation(program, "a_texCoord");

// Create a buffer to put three 2d clip space points in
var positionBuffer = gl.createBuffer();
// Bind it to ARRAY_BUFFER (think of it as ARRAY_BUFFER = positionBuffer)
gl.bindBuffer(gl.ARRAY_BUFFER, positionBuffer);
// Set a rectangle the same size as the image.
setRectangle( gl, 0, 0, image.width, image.height);

// provide texture coordinates for the rectangle.
var texcoordBuffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, texcoordBuffer);
gl.bufferData(gl.ARRAY_BUFFER, new Float32Array([
    0.0,  0.0,
    1.0,  0.0,
    0.0,  1.0,
    0.0,  1.0,
    1.0,  0.0,
    1.0,  1.0,
]), gl.STATIC_DRAW);

// 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);
```

Kemudian pada fungsi render(), fungsi yang dipakai adalah fungsi yang sebelumnya juga telah pernah diimplementasikan pada tutorial sebelumnya. Namun diperlukan penambahan pemanggilan beberapa prosedur seperti setRectangle() untuk melakukan setting ukuran yang ada di canvas agar sama

dengan image. texParameteri() untuk setting parameter sehingga kita dapat melakukan proses rendering untuk semua ukuran yang mungkin, dan texImage2D() untuk melakukan upload image ke dalam tekstur.

```
// lookup uniforms
var resolutionLocation = gl.getUniformLocation(program, "u_resolution");
var textureSizeLocation = gl.getUniformLocation(program, "u_textureSize");
var kernelLocation = gl.getUniformLocation(program, "u_kernel[0]");
var kernelWeightLocation = gl.getUniformLocation(program, "u_kernelWeight");

// 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
  ],
  sharpen: [
     -1, -1, -1,
     -1, 16, -1,
     -1, -1, -1
  ],
  edgeDetect: [
     -0.125, -0.125, -0.125,
     -0.125,  1,     -0.125,
     -0.125, -0.125, -0.125
  ],
  edgeDetect2: [
     -1, -1, -1,
     -1,  8, -1,
     -1, -1, -1
  ],
```

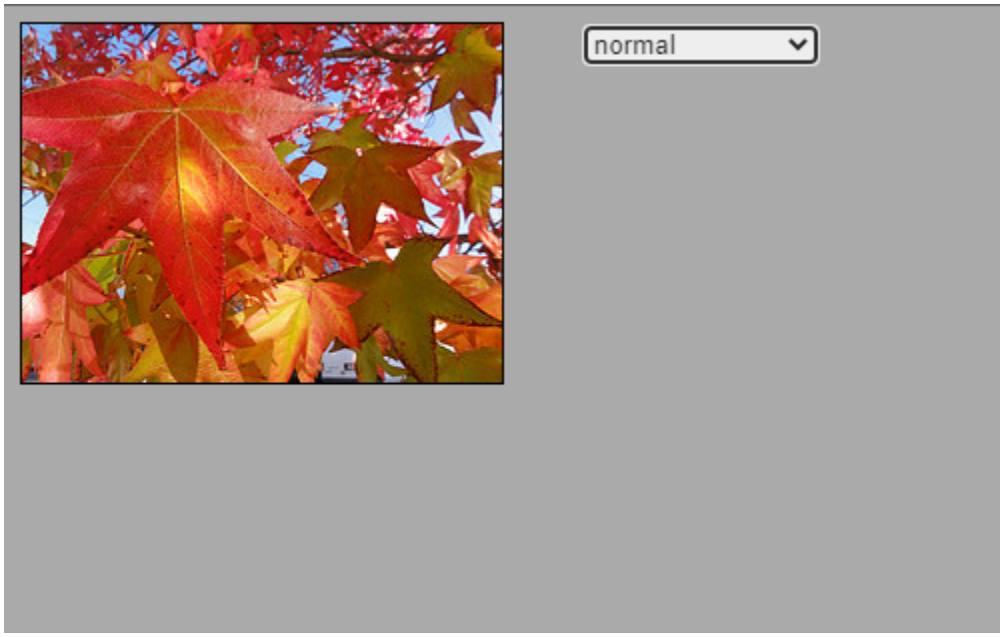
```
edgeDetect3: [
   -5, 0, 0,
    0, 0, 0,
    0, 0, 5
],
edgeDetect4: [
   -1, -1, -1,
    0,  0,  0,
    1,  1,  1
],
edgeDetect5: [
   -1, -1, -1,
    2,  2,  2,
   -1, -1, -1
],
edgeDetect6: [
   -5, -5, -5,
   -5, 39, -5,
   -5, -5, -5
],
sobelHorizontal: [
    1,  2,  1,
    0,  0,  0,
   -1, -2, -1
],
sobelVertical: [
    1,  0, -1,
    2,  0, -2,
    1,  0, -1
],
previtHorizontal: [
    1,  1,  1,
    0,  0,  0,
   -1, -1, -1
],
previtVertical: [
    1,  0, -1,
    1,  0, -1,
    1,  0, -1
],
boxBlur: [
    0.111, 0.111, 0.111,
    0.111, 0.111, 0.111,
    0.111, 0.111, 0.111
],
triangleBlur: [
    0.0625, 0.125, 0.0625,
    0.125,  0.25,  0.125,
    0.0625, 0.125, 0.0625
],
emboss: [
   -2, -1,  0,
   -1,  1,  1,
    0,  1,  2
]
;
```

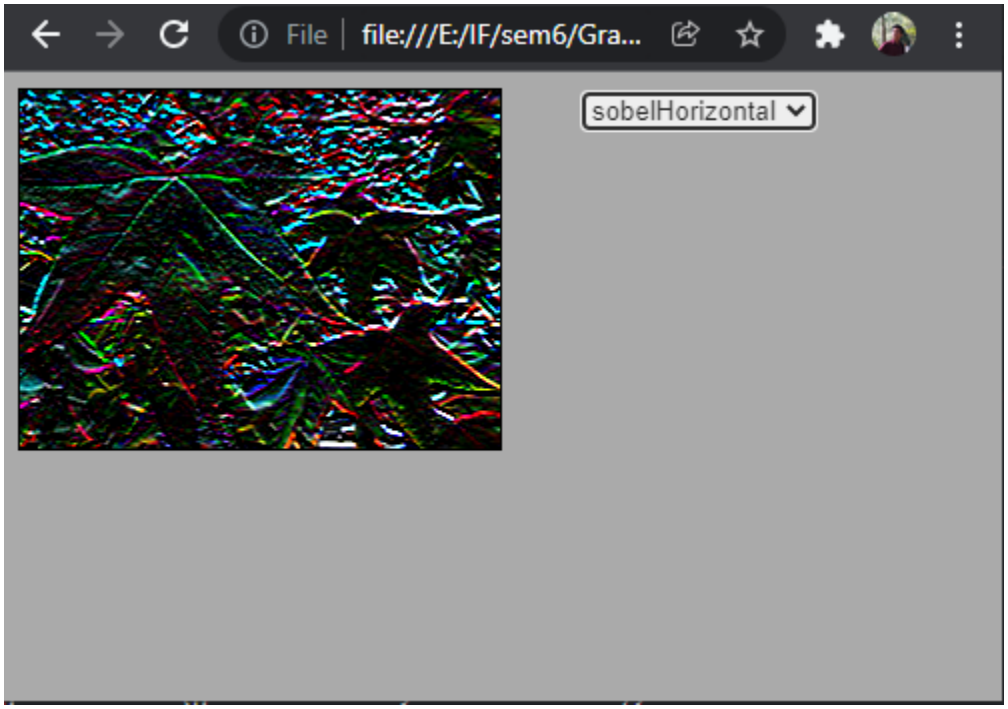
Kemudian, untuk membuat program lebih menarik lagi, kita dapat menambahkan efek dengan melakukan setting sehingga pada akhirnya program akan menampilkan gambar dengan filter tertentu. Untuk code dapat dilihat di bawah ini, pada dasarnya program akan meng-compute seberapa beratnya sebuah kernel/filter untuk kemudian ditampilkan bersamaan dengan gambar tersebut.

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

```
// set the kernel and it's weight
gl.uniform1fv(kernelLocation, kernels[name]);
gl.uniform1f(kernelWeightLocation, computeKernelWeight(kernels[name]));
```

Hasil Program





Link

Link video laporan Youtube: https://youtu.be/sjbuhrohSw4
Link video laporan repository github:
https://github.com/dehanalkautsar/Tutorial-WebGl/tree/main/Image%20Processing