Laporan Tugas WebGL 1

Disusun Oleh :
Aufa Nabil Amiri - 0721 17 4000 0029

Teknik Komputer
Fakultas Teknologi Elektro dan Informatika Cerdas Institut Teknologi Sepuluh Nopember

1 File HTML

```
<body onload="startup();">
  <canvas id="canvas" width="500" height="500" />

  <script type="text/javascript" src="initShader.js"></script>
  <script type="text/javascript" src="index.js"></script>
```

Gambar 1: file html yang digunakan

seperti yang bisa dilihat dalam gambar 1, file html ini akan berfungsi untuk mengatur ukuran dari canvas yang akan digunakan. Canvas sendiri adalah tempat dimana webgl bisa melakukan manipulasi piksel secara bebas

selain itu, juga dilakukan import *script initShader.js* dan *index.js*. *initShader.js* digunakan untuk melakukan inisiasi Vertex Shader dan Fragment Shader. sedangkan *index.js* berisi program utama.

2 initShader.js

```
function loadShader(gl, type, shaderSource) {
  const shader = gl.createShader(type);

  gl.shaderSource(shader, shaderSource);
  gl.compileShader(shader);
      You, 3 days ago • initial test
  // if something happened, delete the shader and return null
  if (!gl.getShaderParameter(shader, gl.COMPILE_STATUS)) {
    console.error(`ERROR ${gl.getShaderInfoLog(shader)}`);
    gl.deleteShader(shader);
    return null;
  }

  return shader;
}
```

Gambar 2: fungsi *loadShader* dalam *initShader.js*

fungsi dalam gambar 2 digunakan untuk melakukan kompilasi terhadap shader yang digunakan. Hasil dari kompilasi akan digunakan untuk mengatur bagaimana sebuah vertex akan ditampilkan di dalam canvas. Apabila dalam

proses kompilasi terjadi error, maka shader yang sudah dikompilasi tadi akan dihapus dan fungsi akan melakukan return null.

```
function setupShaders(gl, vertexSource, fragmentSource) {
  var vertextShader = loadShader(gl, gl.VERTEX_SHADER, vertexSource);
  var fragmentShader = loadShader(gl, gl.FRAGMENT_SHADER, fragmentSource);

  shaderProgram = gl.createProgram();
  gl.attachShader(shaderProgram, vertextShader);
  gl.attachShader(shaderProgram, fragmentShader);
  gl.linkProgram(shaderProgram);

  if (!gl.getProgramParameter(shaderProgram, gl.LINK_STATUS)) {
    console.error("failed to setup shaders");
  }

  return shaderProgram;
}
```

Gambar 3: fungsi setupShader dalam initShader.js

setiap program webgl akan memerlukan vertex shader dan fragment shader. program di gambar 3 digunakan untuk menggabungkan antara vertex shader dan fragment shader dan dengan menggunakan *linkProgram* menghubungkan antara program dengan shader yang sudah digabung tadi.

3 index.js

```
var vertextShaderSource = `
attribute vec3 aVertexPosition;
attribute vec3 color;
varying vec3 vColor;

void main() {
    gl_Position = vec4(aVertexPosition, 1.0);
    vColor = color;
}
`;
```

Gambar 4: vertexShader

yang pertama kali harus di set dalam program webgl adalah vertex shader dan fragment shader. dalam vertex Shader, harus terdapat fungsi main, yaitu

fungsi yang pertama kali akan dipanggil oleh webgl. Selain itu, juga harus terdapat assign value ke gl_Position yang akan digunakan oleh webgl untuk mengetahui posisi piksel yang akan digambar.

hal lain yang harus dilihat adalah *attribute* dan *varying* yang terdapat dalam program vertex. *attribute* digunakan untuk menghubungkan antara program (dalam hal ini index.js) dengan shader. sedangkan *varying* digunakan untuk menghubungkan antara vertex Shader dengan fragment Shader. Terdapat satu lagi jenis variabel yang tidak dipakai dalam shader ini adalah *uniform*.

untuk saat ini, dalam vertex Shader kita hanya melakukan assign gl_Position dan vColor.

```
var fragmentShaderSource = `
precision mediump float;
    You, 4 days ago • initial commit
varying vec3 vColor;

void main() {
    gl_FragColor = vec4(vColor, 1.0);
}
`;
```

Gambar 5: fragmentShader

fragment shader berguna untuk melakukan set bagaimana warna dari suatu vertex. dalam gambar 5, kita hanya mengambil value yang di set di vertex Shader dan memasukkannya ke dalam gl_FragColor. gl_FragColor digunakan oleh webgl dan harus ada dalam fragment Shader.

```
function createGLContext(canvas) {
  var names = ["webgl", "experimental-webgl"];
  var context = null;

  // checking if the browser support WebGL or not
  for (var i = 0; i < names.length; i++) {
    try {
      context = canvas.getContext(names[i]);
    } catch (e) {
      console.error(e);
    }

    if (context) {      You, 4 days ago • initial com
      break;
    }
  }

  if (context) {
    context.viewportWidth = canvas.width;
    context.viewportHeight = canvas.height;
  } else {
    console.error("failed to craete WEBGL context");
  }

  return context;
}
```

Gambar 6: fungsi createContext