

Laporan ETS WebGL



Disusun Oleh :
Aufa Nabil Amiri - 0721 17 4000 0029

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

1 File HTML

Setup Canvas

Listing 1: file index.html

```
1 <html lang="en">
2   <head>
3     <title>ETS</title>
4     <meta charset="utf-8" />
5   </head>
6   <body onload="startup();">
7     <canvas id="canvas" width="500" height="500"></canvas>
8
9     <div style="position: absolute; top: 550px; color: black;
10      ; z-index: 10">
11       Keyboard:
12       <ul>
13         <li>W untuk maju</li>
14         <li>S untuk mundur</li>
15         <li>D untuk rotate ke kanan</li>
16         <li>A untuk rotate ke kiri</li>
17         <li>panah atas untuk ke atas (max. 20)</li>
18         <li>panah bawah untuk ke bawah (min.0)</li>
19       </ul>
20     </div>
```

Melakukan inisiasi *canvas* pada html dengan cara melakukan `<canvas id="canvas"width="500"height="500"></canvas>`. Selain itu, juga dilakukan pemanggilan fungsi `startup()` pada saat file html sudah terload sepenuhnya.

Setup Vertex Shader

Listing 2: file vertexShader

```
1 <script id="vs-src" type="x-shader/x-vertex">
2   attribute vec3 aVertexPosition;
3   attribute vec4 aVertexColor;
4
5   uniform mat4 uModelViewMatrix;
6   uniform mat4 uProjectionMatrix;
7
8   varying vec4 vColor;
9
10  void main(void) {
11    gl_Position = uProjectionMatrix * uModelViewMatrix *
12      * vec4(aVertexPosition, 1.0);
13    vColor = aVertexColor;
```

```
13     }  
14 </script>
```

Nantinya, setiap vertex yang dikirimkan ke vertexShader akan dikalikan dengan `uProjectionMatrix` yang digunakan untuk memposisikan menentukan bagaimana behaviour "kamera" dalam scene. Selanjutnya, akan mengalami proses perkalian dengan `uModelViewMatrix` yang berfungsi untuk menentukan lokasi model dalam koordinat global.

Setup Fragment Shader

Listing 3: file fragmentShader

```
1 <script id="fs-src" type="x-shader/x-fragment">  
2     precision mediump float;  
3  
4     varying vec4 vColor;  
5  
6     void main(void) {  
7         gl_FragColor = vColor;  
8     }  
9 </script>
```

FragmentShader akan digunakan untuk menentukan warna model yang akan ditampilkan di layar.

Load Script yang Dibutuhkan

Listing 4: file Load Script

```
1 <script src="gl-matrix-min.js"></script>  
2 <script type="text/javascript" src="utils.js"></script>  
3 <script type="text/javascript" src="initShader.js"></script>  
4 <script type="text/javascript" src="index.js"></script>  
5 </body>  
6 </html>
```

Disini dilakukan beberapa load script - script yang akan digunakan dalam program nantinya. `gl-matrix-min.js` digunakan untuk mempermudah penghitungan matrix orde 3 dan 4 yang akan digunakan saat melakukan animasi. `utils.js` berisi beberapa fungsi penting seperti `loadWebglContext` yang digunakan untuk mendapatkan context webGL sebelum kita dapat menampilkan objek apapun. `initShader.js` digunakan untuk melakukan compile terhadap vertexShader dan fragmentShader. Dan `index.js` merupakan

an file utama yang paling penting karena berisi merupakan tempat fungsi `startup` berada

2 File Utils.js

Terdapat beberapa fungsi penting di file `utils.js` ini,

- **createGLContext**

Berfungsi untuk mendapatkan webGL context yang akan dipakai di seluruh bagian `index.js` nantinya.

- **getShaderfromDOM**

Mendapatkan ShaderSource baik itu adalah vertexShader maupun fragmentShader dari file `index.html` yang sudah dibuat sebelumnya.

- **createSphere**

Menghitung vertex yang akan digunakan untuk membentuk suatu model sphere.

Listing 5: fungsi createSphere

```
1 function createSphere(div, color) {
2   var positions = [];
3   for (var i = 0; i <= div; ++i) {
4     var ai = (i * Math.PI) / div;
5     var si = Math.sin(ai);
6     var ci = Math.cos(ai);
7     for (var j = 0; j <= div; ++j) {
8       var aj = (j * 2 * Math.PI) / div;
9       var sj = Math.sin(aj);
10      var cj = Math.cos(aj);
11      positions = positions.concat([si * sj, ci, si * ↵
          cj]);
12    }
13  }
14
15  var indices = [];
16  for (var i = 0; i < div; ++i) {
17    for (var j = 0; j < div; ++j) {
18      var p1 = i * (div + 1) + j;
19      var p2 = p1 + (div + 1);
20      indices = indices.concat([p1, p2, p1 + 1, p1 + 1, ↵
          p2, p2 + 1]);
21    }
22  }
```

```
23
24   var colors = [];
25   for (var i = 0; i != indices.length; i++) {
26       colors = colors.concat(color);
27   }
28
29   return {
30       vertexData: positions,
31       indices: indices,
32       colors: colors,
33   };
34 }
```

3 File initShader.js