Laporan EAS 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

```
<html lang="en">
  <head>
    <title>ETS</title>
    <meta charset="utf-8" />
  </head>
  <body onload="startup();">
    <canvas id="canvas" width="500" height="500"></canvas>
```

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

```
    <script id="vs-src" type="x-shader/x-vertex">
      attribute vec3 aVertexPosition;
      attribute vec3 aVertexNormal;

      uniform mat4 uModelViewMatrix;
      uniform mat4 uProjectionMatrix;
      uniform mat4 uNormalViewMatrix;
      uniform vec4 uLightPosition;

      varying vec3 L, N, E;

      void main(void) {
        vec3 pos = (uModelViewMatrix * vec4(aVertexPosition,1.0)).xyz;
        vec3 lightPos = (uNormalViewMatrix * uLightPosition).xyz;

        L = normalize(lightPos - pos);
        N = normalize((uModelViewMatrix * vec4(aVertexNormal,1.0)).xyz);
        E = -normalize(pos);

        gl_Position = uProjectionMatrix * uModelViewMatrix * vec4(aVertexPosition, 1.0);
      }
    </script>
```

mirip seperti assignment sebelumnya, terdapat projectionMatrix, dan ModelViewMatrix. kedua matrix tersebut digunakan untuk melakukan rotasi kepada objek. Variabel `pos` digunakan untuk menyimpan posisi koordinat objek tanpa perspektif kamera dan `lightPost` adalah posisi sumber cahaya dalam kamera.

Variabel `L` adalah vektor arah cahaya, yang perlu dilakukan adalah melakukan normalisasi hasil dari pengurangan posisi cahaya dengan posisi objek sehingga cahaya menyinari objek tersebut. `N` adalah vektor normal objek, vektor normal ini ikut berubah posisinya mengikuti vertex objek. `E` adalah arah kamera. Karena kamera dianggap berada pada titik (0,0), arah kamera ini hanyalah normalisasi dari koordinat objek.

Setup Fragment Shader

Listing 3: file fragmentShader

```
<script id="fs-src" type="x-shader/x-fragment">
  precision mediump float;
  varying vec3 L, N, E;

  uniform vec4 ambientProduct;
  uniform vec4 diffuseProduct;
  uniform vec4 specularProduct;
  uniform float shininess;

  void main()
  {
    vec4 diffuse = max(dot(L, N), 0.0) * diffuseProduct;
    vec3 H = normalize(L+E);
    vec4 specular = pow(max(dot(N, H), 0.0), shininess) ↩
       * specularProduct;

    if (dot(L, N) < 0.0)
      specular = vec4(0.0, 0.0, 0.0, 1.0);

    vec4 fColor = ambientProduct + diffuse + specular;
    fColor.a = 1.0;

    gl_FragColor = fColor;
  }
</script>
```

Disini fragmentShader mengatur warna dengan melihat posisi dan arah cahaya dan sifat material. `Diffuse` adalah warna utama yang akan ditampilkan oleh objek saat terkena cahaya. Nilai minimal dari `diffuse` adalah 0

karena apabila nilai tersebut kurang dari 0, berarti cahaya berada di belakang bidang, yang mana juga sama - sama berarti gelap.

Untuk melakukan penghitungan specular, karena implementasi yang dipilih adalah Phong shader dengan variasi Blinn, terdapat tambahan penghitungan untuk mencari jarak tengah antara vektor arah kamera dengan vektor arah cahaya. Variabel `H` digunakan untuk menyimpan jarak tengah tersebut. Specular sendiri dihitung dengan cara melakukan normalisasi antara vektor normal dengan `H` dan dipangkatkan dengan banyaknya `shininess` yang dibutuhkan.

Untuk memudahkan penghitungan di GPU, yang diinputkan bukan merupakan besaran sifat - sifat Light dan Materialnya, tapi sudah merupakan hasil perkalian dari kedua sifat tersebut.

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 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.

- **createPyramid**

  Menghitung vertex, indices, dan normal yang akan digunakan untuk membentuk suatu model pyramid.

Listing 5: fungsi createPyramid

```
function createPyramid() {
  //prettier-ignore
  const positions = [

  // Bottom face
  -1.0, -1.0, -1.0,
   1.0, -1.0, -1.0,
   1.0, -1.0,  1.0,
  -1.0, -1.0,  1.0,

   0.0,  1.5,  0.0
  ]

  //prettier-ignore
  const indices = [
    0,  1,  2,      0,  2,  3,    // front
    0,  1,  4,      2,  3,  4,    // back
    0,  3,  4,      1,  2,  4,   // top
  ];

  var normal = [];

  for (i = 0; i <= indices.length - 3; i += 3) {
    a = indices[i] * 3;
    b = indices[i + 1] * 3;
    c = indices[i + 2] * 3;

    p1 = vec3.fromValues(positions[a], positions[a + 1], positions[a + 2]);
    p2 = vec3.fromValues(positions[b], positions[b + 1], positions[b + 2]);
    p3 = vec3.fromValues(positions[c], positions[c + 1], positions[c + 2]);

    t1 = vec3.create();
    t2 = vec3.create();
    result = vec3.create();

```

```
36      vec3.subtract(t1, p2, p1);
37      vec3.subtract(t2, p3, p1);
38
39      vec3.cross(result, t1, t2);
40      vec3.normalize(result, result);
41
42      normal = normal.concat([result[0], result[1], ↩
            result[2]]);
43    }
44
45    return {
46      vertexData: positions,
47      indices: indices,
48      normal: normal,
49    };
```

3 File initShader.js

File ini digunakan untuk melakukan load terhadap shader dari ShaderSource menjadi WebGLProgram. fungsi `loadShader` merupakan fungsi untuk melakukan compile shaderSource untuk kemudian agar bisa dilakukan `gl.attachShader` dalam webGL. Berikut adalah salah satu fungsi yang ada di file `initShader.js`.

Listing 6: fungsi setupShader

```
1  function setupShaders(gl, vertexSource, fragmentSource) {
2    var vertextShader = loadShader(gl, gl.VERTEX_SHADER, ↩
         vertexSource);
3    var fragmentShader = loadShader(gl, gl.FRAGMENT_SHADER, ↩
         fragmentSource);
4
5    shaderProgram = gl.createProgram();
6    gl.attachShader(shaderProgram, vertextShader);
7    gl.attachShader(shaderProgram, fragmentShader);
8    gl.linkProgram(shaderProgram);
9
10   if (!gl.getProgramParameter(shaderProgram, gl.LINK_STATUS)↩
         ) {
11     console.error("failed to setup shaders");
12   }
13
14   return shaderProgram;
15 }
```

4 File index.js

File ini adalah backbone utama dari project WebGL ini, berisi konfigurasi viewport webgl, object modelling, object animation dan banyak lagi.

object anim

Saya memutuskan untuk menggunakan objek anim untuk menghindari penggunaan global variabel dalam script.

Listing 7: objek anim

```
1
2  var anim = {
3    camLocation: [4, 0, 0],
4    povLoc: [0, 0, 0],
5
6    render: function () {
7      anim.gl.viewport(0, 0, anim.canvas.width, anim.canvas.↩
         height);
8      anim.gl.clear(anim.gl.COLOR_BUFFER_BIT | anim.gl.↩
         DEPTH_BUFFER_BIT);
9
10     drawAttr(
11       anim.gl,
12       new Float32Array(objectArray.object.vertexData),
13       3,
14       anim.programInfo.attr.vertexPostition
15     );
16
17     drawAttr(
18       anim.gl,
19       new Float32Array(objectArray.object.normal),
20       // objectArray.object.normal,
21       3,
22       anim.programInfo.attr.normal
23     );
24
25     drawIndices(anim.gl, objectArray.object.indices);
26
27     tempTranslate = animate(
28       anim.gl,
29       anim.canvas,
30       anim.programInfo.uniform,
31       objectArray.object.indices.length,
32       objectArray.currentRotation,
33       anim.camLocation,
34       anim.povLoc
```

```
35      );
36
37      objectArray.currentRotation =
38        objectArray.currentRotation + objectArray.↩
            deltaRotation;
39
40      window.requestAnimationFrame(this.render);
41    },
42  };
```

Seperti bisa dilihat di potongan kode 7, terdapat beberapa variabel penting yang disimpan dalam objek tersebut, seperti :

- **camLocation**, berisi koordinat posisi kamera, akan digunakan saat memasukkan value lookAt
- **povLoc**, titik yang "dilihat" oleh kamera
- **gl**, menyimpan referensi WebGLRenderingContext
- **canvas**, berisi HTMLCanvasElement
- **render**, sebuah fungsi untuk melakukan setup webGL attribute (avertexPosition dan aVertexNormal), setup indices yang digunakan saat proses draw, mengatur rotasi dan memanggil fungsi `animate`.

fungsi Animate

Dalam project ini, terdapat beberapa step dalam melakukan animasi. Yang pertama adalah mengatur `projectionMatrix`, dilanjutkan dengan mengatur `modelViewMatrix`, disambung dengan menghitung product hasil pencahayaan dan terakhir melakukan draw model ke canvas.

Listing 8: mengatur projectionMatrix

```
1 //projection View Matrix
2 const fieldOfView = (45 * Math.PI) / 180; //45 degree angle
3 const aspect = canvas.clientWidth / canvas.clientHeight;
4 const projectionMatrix = mat4.create();
5 mat4.perspective(projectionMatrix, fieldOfView, aspect, 0.1,↩
    100.0);
```

Potongan kode 8 digunakan untuk mengatur bentuk projectionMatrix dari koordinat dunia menjadi koordinat dalam viewport. Disini saya hanya merubah dari default projection, yaitu orthographic menjadi perspective.

Listing 9: mengatur modelViewMatrix

```
//ModelView Matrix
  var modelViewMatrix = mat4.create();

  mat4.lookAt(modelViewMatrix, camLocation, POVLoc, [0, 1, 0]);

  var normalViewMatrix = mat4.clone(modelViewMatrix);

  mat4.rotate(modelViewMatrix, modelViewMatrix, rotate, [0, 1, 0]);
```

Potongan kode 9 digunakan untuk menghitung modelViewMatrix yang digunakan untuk mengubah dari koordinat lokal menjadi koordinat dunia. `normalViewMatrix` dibutuhkan karena dalam penghitungan, posisi cahaya tidak ikut berputar mengikuti objek.

Listing 10: mengatur pencahayaan

```
//lighting...
  var ambientProduct = vec4.create();
  vec4.multiply(ambientProduct, lightAmbient, materialAmbient);

  var diffuseProduct = vec4.create();
  vec4.multiply(diffuseProduct, lightDiffuse, materialDiffuse);

  var specularProduct = vec4.create();
  vec4.multiply(specularProduct, lightSpecular, materialSpecular);
```

potongan kode 10 digunakan agar penghitungan product, tidak menggunakan GPU tapi masih berada dalam CPU, diharapkan dengan melakukan penghitungan di CPU, dapat lebih meningkatkan FPS dalam render.

Listing 11: mengatur posisi sumber cahaya

```
  if (isFlip === true) {
    lightPosition[0] += 0.05;
  } else {
    lightPosition[0] -= 0.05;
  }

  if (lightPosition[0] * lightPosition[0] > 4) {
    isFlip = !isFlip;
  }
```

Kode 11 dibuat untuk memenuhi salah satu kriteria yang harus ada dalam projek, yaitu sumber cahaya yang bergerak. Disini saya hanya melakukan

pergerakan simpel dari kanan ke kiri dan sebaliknya (namun karena `up` pada `lookAt` berada pada axis Y, maka cahaya tampak bergerak dari atas ke bawah)

Listing 12: draw

```
1   drawMatUniform(gl, uniformLocation.projection, ↩
        projectionMatrix);
2   drawMatUniform(gl, uniformLocation.modelView, ↩
        modelViewMatrix);
3   drawMatUniform(gl, uniformLocation.normalView, ↩
        normalViewMatrix);
4
5   drawVecUniform(gl, uniformLocation.lightPos, lightPosition↩
        );
6   drawVecUniform(gl, uniformLocation.ambientProduct, ↩
        ambientProduct);
7   drawVecUniform(gl, uniformLocation.diffuseProduct, ↩
        diffuseProduct);
8   drawVecUniform(gl, uniformLocation.specularProduct, ↩
        specularProduct);
9
10  gl.uniform1f(uniformLocation.shininess, materialShininess)↩
        ;
11
12  gl.drawElements(gl.TRIANGLES, vertexLength, gl.↩
        UNSIGNED_SHORT, 0);
13
14 return modelViewMatrix;
```

Terakhir tinggal melakukan setup uniform ke webGL dan melakukan `drawElement` untuk melakukan render model ke viewport.

Fungsi Startup

Adalah fungsi yang pertama kali dipanggil saat dokumen html selesai di load oleh browser,

Listing 13: mengatur eventListener

```
1   /** .. */
2
3   gl.clearColor(102 / 255, 153 / 255, 1.0, 1.0);
4   gl.enable(gl.DEPTH_TEST);
5   gl.clear(gl.COLOR_BUFFER_BIT);
6   gl.viewport(0, 0, gl.viewportWidth, gl.viewportHeight);
```

Selain itu, juga terdapat fungsi webGL untuk mengatur warna background dan viewport.

URL Demo dan Source Code

Seluruh source code dalam project ini dapat diakses dalam repository github saya, yaitu `https://github.com/chillytaka/webGL/tree/master/eas`, sedangkan untuk demo, dapat diakses di `https://chillytaka.github.io/webGL/eas/index.html`. Selain itu, juga terdapat daftar dari projek - projek webGL yang lain yang dapat diakses di `https://chillytaka.github.io/webGL/`