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

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

    <div style="position: absolute; top: 550px; color: black; z-index: 10">
      Keyboard:
      <ul>
        <li>W untuk maju</li>
        <li>S untuk mundul</li>
        <li>D untuk rotate ke kanan</li>
        <li>A untuk rotate ke kiri</li>
        <li>panah atas untuk ke atas (max. 20)</li>
        <li>panah bawah untuk ke bawah (min.0)</li>
      </ul>
    </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

```
    <script id="vs-src" type="x-shader/x-vertex">
      attribute vec3 aVertexPosition;
      attribute vec4 aVertexColor;

      uniform mat4 uModelViewMatrix;
      uniform mat4 uProjectionMatrix;

      varying vec4 vColor;

      void main(void) {
          gl_Position = uProjectionMatrix * uModelViewMatrix
              * vec4(aVertexPosition, 1.0);
          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` merupak-

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

```
function createSphere(div, color) {
  var positions = [];
  for (var i = 0; i <= div; ++i) {
    var ai = (i * Math.PI) / div;
    var si = Math.sin(ai);
    var ci = Math.cos(ai);
    for (var j = 0; j <= div; ++j) {
      var aj = (j * 2 * Math.PI) / div;
      var sj = Math.sin(aj);
      var cj = Math.cos(aj);
      positions = positions.concat([si * sj, ci, si * ↩
          cj]);
    }
  }

  var indices = [];
  for (var i = 0; i < div; ++i) {
    for (var j = 0; j < div; ++j) {
      var p1 = i * (div + 1) + j;
      var p2 = p1 + (div + 1);
      indices = indices.concat([p1, p2, p1 + 1, p1 + 1,↩
          p2, p2 + 1]);
    }
  }
```

```
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

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.

data planet

Untuk memudahkan dalam pengembangan, saya memutuskan untuk menyatukan konfigurasi planet - planet yang akan di render dalam scene. Dengan cara memisahkan antara konfigurasi planet dengan cara menggambar nantinya akan memudahkan saat proses debugging.

Listing 7: list data planet

```
1  {
2    color: [1, 0.97, 0.19, 1.0],
3    translation: [0, 0, 0],
4    rotation: 0.01,
5    scale: 1,
6    hasChild: true,
7    parentId: -1,
8  }
```

Potongan kode 7 adalah salah satu data planet yang digunakan, untuk keterangannya adalah :

- **color**, menyimpan value warna planet
- **translation**, menyimpan lokasi koordinat lokal planet
- **rotation**, kecepatan rotasi dan kecepatan planet memutari parentnya
- **scale**, ukuran planet, hal ini karena fungsi `createSphere` di potongan kode 5 hanya membuat sphere dengan ukuran radius 1.0.
- **hasChild**, sebuah boolean value untuk menunjukkan apakah suatu planet memiliki child dalam hierarki
- **parentId**, menyimpan id parent yang akan digunakan dalam hierarki

object anim

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

Listing 8: objek anim

```
var anim = {
  prevModelView: [],
  camLocation: [0, -10, 0],
  povLoc: [0, 0, 0],

  render: function () {
    anim.gl.viewport(0, 0, anim.canvas.width, anim.canvas.height);
    anim.gl.clear(anim.gl.COLOR_BUFFER_BIT | anim.gl.DEPTH_BUFFER_BIT);

    //reset prevModelVIew
    anim.prevModelView = [];

    for (var i = 0; i != objectArray.length; i++) {
      drawAttr(
        anim.gl,
        new Float32Array(objectArray[i].object.vertexData),
        3,
        anim.programInfo.attr.vertexPostition
      );

      drawAttr(
        anim.gl,
        new Float32Array(objectArray[i].object.colors),
        4,
        anim.programInfo.attr.color
      );

      drawIndices(anim.gl, objectArray[i].object.indices);

      tempTranslate = animate(
        anim.gl,
        anim.canvas,
        anim.programInfo.uniform,
        objectArray[i].object.indices.length,
        objectArray[i].currentRotation,
        objectArray[i].scale,
        objectArray[i].translation,
        anim.prevModelView[objectArray[i].parentId],
        anim.camLocation,
        anim.povLoc
      );

      if (objectArray[i].hasChild) {
        anim.prevModelView.push(tempTranslate);
      }

```

```
47        objectArray[i].currentRotation =
48          objectArray[i].currentRotation + objectArray[i].↩
              deltaRotation;
49      }
50
51      window.requestAnimationFrame(this.render);
52    },
53  };
```

Seperti bisa dilihat di potongan kode 8, terdapat beberapa variabel penting yang disimpan dalam objek tersebut, seperti :

- **prevModelView**, menyimpan modelView parent yang nantinya akan dimasukkan ke dalam penghitungan modelViewMatrix para child
- **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 aVertexColor), setup indices yang digunakan saat proses draw, memasukkan modelView parent ke dalam prevModelParent, 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` dan terakhir melakukan draw model ke canvas.

Listing 9: 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 9 digunakan untuk mengatur bentuk projectionMatrix dari koordinat dunia menjadi koordinat dalam viewport. Disini saya hanya merubah dari default projection, yaitu orthographic menjadi perspective.

Listing 10: mengatur modelViewMatrix

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

if (prevModelView != undefined) {
  modelViewMatrix = mat4.clone(prevModelView);
} else {
  mat4.lookAt(modelViewMatrix, camLocation, POVLoc, [0, 0, 1]);
}

mat4.scale(modelViewMatrix, modelViewMatrix, [scale, scale, scale]);
mat4.rotate(modelViewMatrix, modelViewMatrix, rotate, [0, 0,1]);
mat4.translate(
  modelViewMatrix, // destination matrix
  modelViewMatrix, // matrix to translate
  translate
);
```

Potongan kode 10 digunakan untuk menghitung modelViewMatrix yang digunakan untuk mengubah dari koordinat lokal menjadi koordinat dunia.

`mat4.lookAt` hanya di apply pada model yang paling atas dalam hierarki, yaitu planet matahari, sedangkan sisa planet akan menggunakan hasil penghitungan modelViewMatrix dari matahari sehingga pada akhirnya ikut terpengaruh oleh `lookAt`.

Untuk transformasi yang dilakukan terhadap model, yang pertama adalah melakukan scale agar memiliki scale yang sesuai dengan data yang sudah di set sebelumnya, dilanjutkan dengan melakukan rotasi. Hasil dari rotasi tersebut akan dilakukan translasi sehingga planet tampak memutari koordinat pusat, yaitu [0,0]

Listing 11: draw

```
drawUniform(gl, uniformLocation.projection, projectionMatrix);
drawUniform(gl, uniformLocation.modelView, modelViewMatrix);

gl.drawElements(gl.TRIANGLES, vertexLength, gl.UNSIGNED_SHORT, 0);

return modelViewMatrix;
```

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

fungsi onKeyUp

fungsi ini digunakan untuk memproses event keypress oleh user sehingga menambah interaktifitas dalam model - model planet.

Listing 12: fungsi onKeyUp

```
function onKeyUp(event) {
  const defaultJump = 5;

  if (event.keyCode == 38 && anim.camLocation[2] <= 20) {
    anim.camLocation[2] += defaultJump;
  }
  if (event.keyCode == 40 && anim.camLocation[2] >= 0) {
    anim.camLocation[2] -= defaultJump;
  }

  if (event.keyCode == 87 && anim.camLocation[1] <= 20) {
    anim.camLocation[1] += defaultJump;
  }
  if (event.keyCode == 83 && anim.camLocation[1] >= -30) {
    anim.camLocation[1] -= defaultJump;
  }

  if (event.keyCode == 68) {
    anim.povLoc[0] += defaultJump / 2;
  }
  if (event.keyCode == 65) {
    anim.povLoc[0] -= defaultJump / 2;
  }
}
```

Beberapa tombol yang dapat digunakan adalah :

- **keycode 38 atau panah atas,**

  Digunakan menambah ukuran Z axis pada lokasi kamera, sehingga kamera menjadi diatas model planet. Sengaja dibatasi karena apabila value Z melewati titik tertentu, objek menjadi tampak aneh sebelum akhirnya menghilang karena terlalu kecil.

- **keycode 40 atau panah bawah,**

  sama seperti diatas, digunakan untuk mengurangi ukuran Z akis pada lokasi kamera.

- **keycode 87 atau "w",**

  Digunakan untuk membuat kamera maju dengan cara menambah value Y axis pada lokasi kamera. Pembatasan value dilakukan agar kamera planet - planet masih dapat terlihat.

- **keycode 83 atau "s",**

  Sama seperti diatas, digunakan untuk membuat kamera mundur dengan cara mengurangi value Y axis pada lokasi kamera.

- **keycode 68 atau "d",**

  Digunakan untuk merubah koordinat yang dilihat oleh kamera kearah kanan. Memberikan efek rotasi pada kamera.

- **keycode 65 atau "a",**

  Digunakan untuk merubah koordinat yang dilihat oleh kamera kearah kiri. Memberikan efek rotasi pada kamera.

Fungsi Startup

Adalah fungsi yang pertama kali dipanggil saat dokumen html selesai di load oleh browser, Salah satu yang penting adalah melakukan setup agar program dapat menangkap keyboard press.

Listing 13: mengatur eventListener

```
1  document.addEventListener("keydown", onKeyUp, true);
2
3  /** .. */
4
5  gl.clearColor(102 / 255, 153 / 255, 1.0, 1.0);
6  gl.enable(gl.DEPTH_TEST);
7  gl.clear(gl.COLOR_BUFFER_BIT);
8  gl.viewport(0, 0, gl.viewportWidth, gl.viewportHeight);
```

Selain itu, juga terdapat fungsi webGL untuk mengatur warna background dan viewport. Tidak kalah penting adalah menyalakan fitur Hidden Surface Removal agar objek yang tertutupi oleh objek lain akan menjadi tidak tampak dengan cara `gl.enable(gl.DEPTH_TEST);`.

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/ets`, sedangkan untuk demo, dapat diakses di `https://chillytaka.github.io/webGL/ets/index.html`. Selain itu, juga terdapat daftar dari projek - projek webGL yang lain yang dapat diakses di `https://chillytaka.github.io/webGL/`