Laporan Tugas Besar 1
WebGL: 2D Primitive Elements

Dibuat sebagai Template Laporan Tugas Besar Grafika Komputer IF3260

Oleh:

Matthew Mahendra	13521007
Kenny Benaya Nathan	13521023
Ahmad Nadil	13521024

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

2024

Daftar Isi

Daftar Gambar

BAB 1

Pendahuluan

1.1 Deskripsi Tugas

Pada tugas ini, akan dibuat sebuah program computer aided-design (CAD) berbasis WebGL 1.0. Program dapat membuat dan menampilkan model 2D berupa bentuk persegi, persegi panjang, garis, dan persegi panjang. Model-model yang dibangun kemudian dapat dilakukan transformasi geometri maupun transformasi shader melalui perubahan warna pada model.

BAB 2

Implementasi Model dan Shader

2.1 Shaders

Terdapat 2 shader yang digunakan dalam program ini yaitu vertex shader dan fragment shader. Vertex shader digunakan untuk meletakkan setiap titik pada canvas dan fragment shader digunakan untuk mewarnai vertex dan fragment yang terbentuk.

Snippet shader yang digunakan adalah sebagai berikut,

```
const vs = ` // vertex shader
attribute vec2 position;
attribute vec3 color;
varying vec3 vertex_color;

void main() {
    gl_Position = vec4(position, 0.0, 1.0);
    vertex_color = color;
}`

const fs = ` // fragment shader
precision mediump float;
varying vec3 vertex_color;

void main(){
    gl_FragColor = vec4(vertex_color, 1);
}
`
```

Variabel position merupakan attribute vec2 untuk model 2D dan variabel color attribute vec3 untuk model warna menggunakan RGB yang dinormalisasi ke 0-1.

2.2 WebGL

WebGL digunakan untuk mendefinisikan cara menggambar titik-titik yang terdapat pada buffer. Setiap data titik yang dikirimkan mengandung 5 data informasi yaitu titik koordinat dan nilai RGB. Dengan demikian setiap titik didefinisikan sebagai Vertex: $< x, y, r, g, b >$.

Dengan demikian cara memproses setiap titik yang diterima harus dibagi menjadi 2 data pertama sebagai position dan 3 data terakhir sebagai color. Snippet penggunaan WebGL (dengan penyederhanaan dari implementasi akhir pada program) adalah sebagai berikut,

```
const canvas = document.getElementById("glCanvas")
const gl = canvas.getContext("webgl")

// Mendefinisikan Viewport dan background
gl.viewport(0, 0, canvas.width, canvas.height)
gl.clearColor(0.39, 0.39, 0.39, 0)
gl.clear(gl.COLOR_BUFFER_BIT)

// Membuat Vertex Shader
var vertexShader = createShader(gl, gl.VERTEX_SHADER, vs)
// Membuat Fragment Shader
var fragmentShader = createShader(gl, gl.FRAGMENT_SHADER, fs)

// Buat program dari vertex shader dan fragment shader
var program = createProgram(gl, vertexShader, fragmentShader)
gl.useProgram(program)

// Definisi Vertex Buffer
var vertexBuffer = gl.createBuffer()
gl.bindBuffer(gl.ARRAY_BUFFER, vertexBuffer)

// Binding attribute dari program
var positionAttributeLocation = gl.getAttribLocation(program, "position
    ")
var colorAttributeLocation = gl.getAttribLocation(program, 'color')

// Mendefinisikan membaca tiitk koordinat
gl.enableVertexAttribArray(positionAttributeLocation)
gl.vertexAttribPointer(positionAttributeLocation, 2, gl.FLOAT, false, 5
     * Float32Array.BYTES_PER_ELEMENT, 0)

// Mendefinisikan membaca nilai warna
gl.enableVertexAttribArray(colorAttributeLocation)
gl.vertexAttribPointer(colorAttributeLocation, 3, gl.FLOAT, false, 5 *
    Float32Array.BYTES_PER_ELEMENT, 2 * Float32Array.BYTES_PER_ELEMENT)

// Memasukkan nilai vertices ke dalam buffer
gl.bufferData(
  gl.ARRAY_BUFFER,
  new Float32Array(vertices),
  gl.STATIC_DRAW
)

// Cara membaca program, berdasarkan definisi cara membaca (dipotong
    setiap 5 titik). this.type salah satu dari: gl.POINTS, gl.LINE_STRIP
    , gl.LINE_LOOP, gl.LINES, gl.TRIANGLE_STRIP, gl.TRIANGLE_FAN, gl.
    TRIANGLES
gl.drawArrays(this.type, 0, this.vertices.length / 5)
```

2.3 Model Basis Shape

Model basis shape mengandung informasi cara melakukan rendering dan nilai default dari setiap model yang dibentuk serta vertex awal yang digunakan. Selain itu terdapat listener untuk setiap elemen pada HTML untuk mengubah nilai warna maupun melakukan

transformasi. Setiap model yang akan dibentuk diturunkan dari model basis ini.

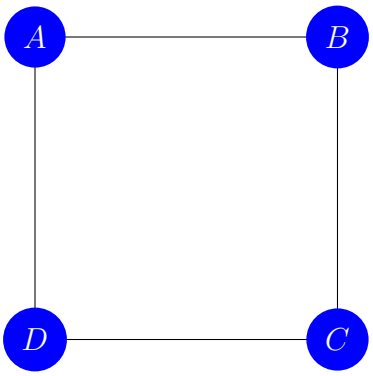
2.4 Model Garis

Model garis diwujudkan dengan mendefinisikan dua Vertex dan menggunakan metode menggambar gl.LINES. Selain itu, untuk membantu transformasi geometri, didefinisikan titik tengah dari suatu garis. Titik tengah didapatkan melalui persamaan 2.1. Model diwujudkan dalam sebuah kelas Line.

$$(x, y) = \frac{(x_1, y_1) + (x_2, y_2)}{2} \tag{2.1}$$

2.5 Model Persegi

Model persegi diwujudkan dengan mendefinisikan 4 Vertex dan menggunakan metode menggambar gl.TRIANGLE_FAN. Definisi dari keempat titik mengikuti arah jarum jam secara berurutan agar dihasilkan persegi yang sempurna saat penggambaran, yaitu pada titik A, B, C, dan D pada gambar 2.1.



Gambar 2.1: Persegi

Dalam pembentukan model, diberikan *constraint*, untuk memastikan bahwa sisi-sisi yang dibangun haruslah sama semua ukurannya. Definisi titik tengah didapatkan dengan mencari titik tengah antara diagonal A dan C menggunakan persamaan 2.1.

2.6 Model Persegi Panjang

Model persegi panjang dibangun dengan metode yang serupa dengan membuat model persegi panjang. Akan tetapi, dalam pembentukannya tidak menggunakan *constraint* agar sisinya sama, tetapi agar sisi yang paralel sama. Pencarian titik tengah model menggunakan perhitungan yang sama pula untuk titik sudut yang bersesuaian dengan gambar 2.1 (titik A dan C) dan menggunakan perhitungan 2.1.

2.7 Model Poligon

Setiap titik pada poligon akan didefinisikan terlebih dahulu lalu kemudian dimasukkan ke dalam vertex buffer. Titik tengah didefinisikan sebagai rata-rata dari seluruh titik

yang ada dengan perhitungan pada 2.2. Setiap vertex akan digambarkan menggunakan metode gl.TRIANGLE_FAN. Untuk membentuk poligon dari titik-titik yang dibentuk, digunakan perhitungan convex hull untuk menentukan luasan terkecil dari sejumlah titik yang dibentuk.

$$(x, y) = \frac{\sum_{i=1}^{n}(x_i, y_i)}{n} \tag{2.2}$$

BAB 3

Implementasi Transformasi Geometri

3.1 Dilatasi

Dilatasi adalah proses merubah sebuah objek untuk diperbesar atau diperkecil dengan faktor k. Untuk $k \geq 1$, dilakukan perbesaran dan $k < 1$, dilakukan pengecilan. Matriks transformasi untuk dilatasi pada objek dua dimensi dapat dilihat pada matrix 3.1.

$$M_D = \begin{bmatrix} k & 0 \\ 0 & k \end{bmatrix} \tag{3.1}$$

Sedangkan untuk perbesaran pada satu buah koordinat saja menggunakan matrix 3.2 dan 3.3.

$$M_{D_x} = \begin{bmatrix} k & 0 \\ 0 & 1 \end{bmatrix} \tag{3.2}$$

$$M_{D_y} = \begin{bmatrix} k & 0 \\ 0 & 1 \end{bmatrix} \tag{3.3}$$

3.2 Rotation

Suatu objek dapat dirotasi sejauh α derajat atau radian menggunakan matriks rotasi. Untuk objek dua dimensi digunakan matriks 3.4.

$$M_R = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix} \tag{3.4}$$

3.3 Shearing

Untuk menggeser suatu sumbu sejauh k_y dan k_x tetapi tetap mempertahankan arahnya, digunakan shearing. Pada objek dua dimensi, digunakan matriks 3.5.

$$M_S = \begin{bmatrix} 1 & k_y \\ k_x & 1 \end{bmatrix} \tag{3.5}$$

3.4 Translation

Translasi adalah proses memindahkan suatu titik ke titik yang lain sejauh jarak tertentu. Pada objek dua dimensi digunakan matriks 3.6.

$$T = \begin{bmatrix} T_x \\ T_y \end{bmatrix} \tag{3.6}$$

3.5 Transformasi Komposisi pada WebGL

Pada program yang dibuat, didefinisikan titik pusat objek sebagai $P = (x_p, y_p)$. Untuk itu transformasi dilakukan satu per satu dengan urutan rotasi, dilatasi, dilatasi pada x dan y, shearing, dan translasi. Transformasi akhir didapatkan dengan perhitungan 3.7.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = M_D M_{D_x} M_{D_y} M_S \left(M_R \begin{bmatrix} x - x_p \\ y - y_p \end{bmatrix} \right) + T + \begin{bmatrix} x_p \\ y_p \end{bmatrix} \tag{3.7}$$

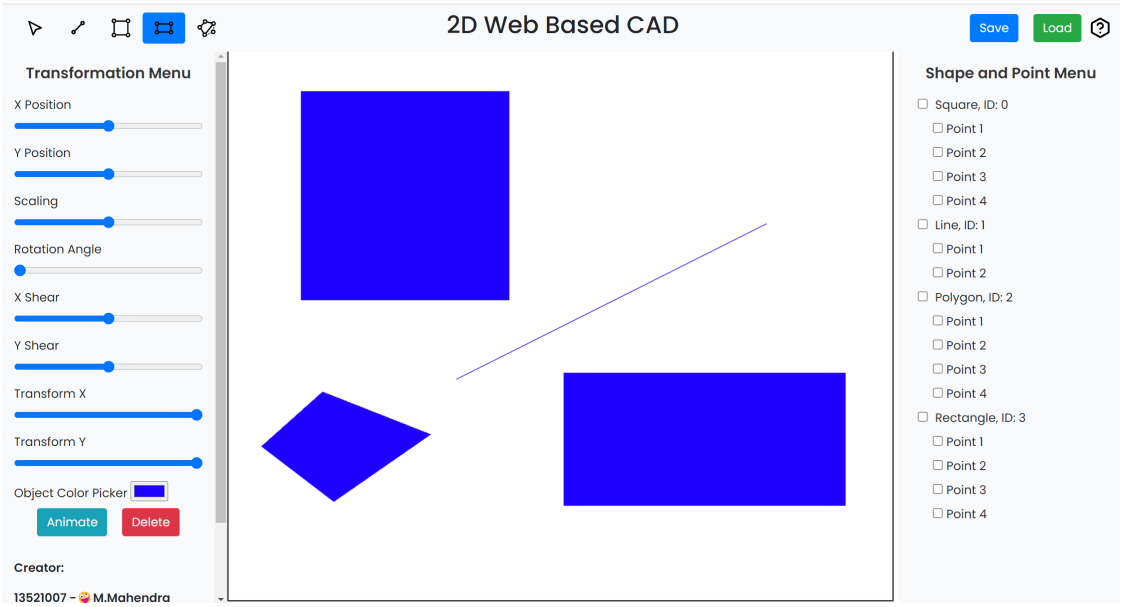
BAB 4

Program Akhir

4.1 Implementasi Website

4.1.1 Overview Implementasi

Sebuah website sederhana menggunakan HTML dan JS berbasis WebGL 1.0. HTML digunakan untuk membuat elemen canvas dan pengubah objek dan JS untuk mengatur rendering serta logic objek lainnya. Hasil implementasi dapat dilihat pada gambar 4.1.



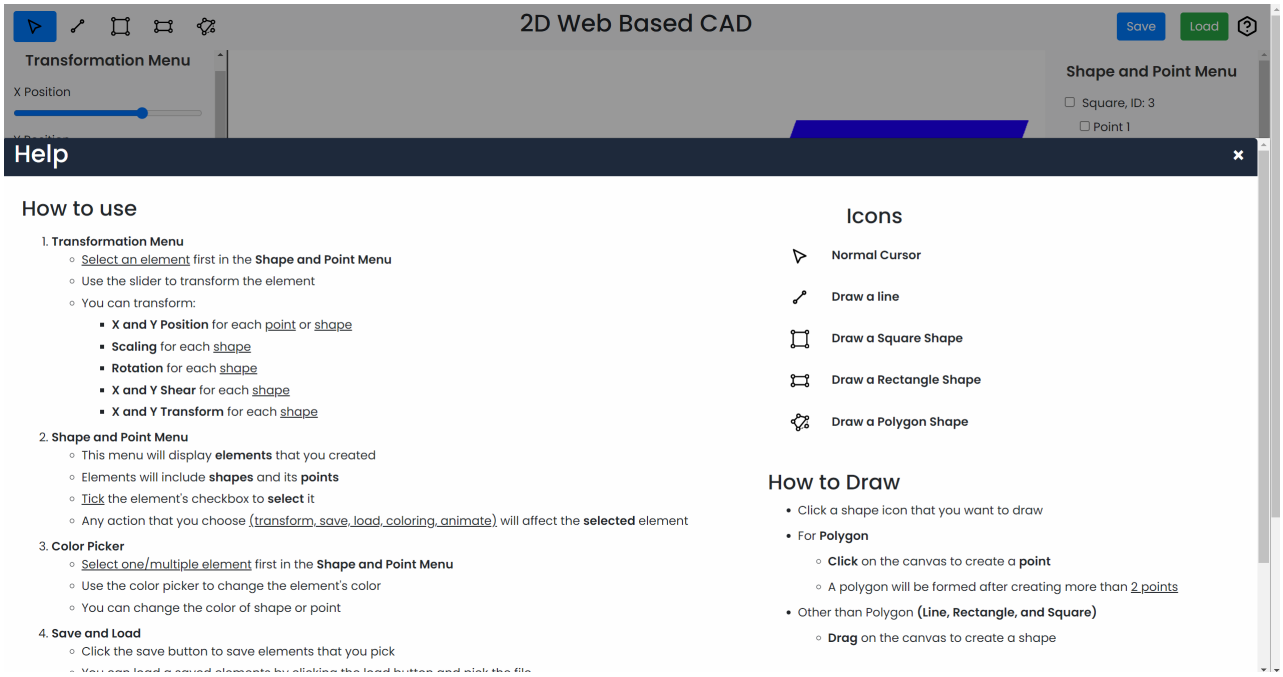
Gambar 4.1: Implementasi Website WebGL

Canvas dapat digambar dengan model-model yang berbeda sesuai dengan model pada bab sebelumnya. Titik dan atau model dapat dipilih untuk digeser secara mandiri atau melakukan perubahan terhadap warna setiap titik. Selain itu, dapat dilakukan penghapusan model secara keseluruhan maupun menghapus titik pada model.

Animasi diimplementasikan pada model dengan menambah derajat rotasi secara incremental dengan nilai 0.3. Agar sudut tidak melebihi 360°, maka dilakukan konversi dari sudut derajat menjadi nilai radian. Selain itu, model dapat disimpan sebagai file JSON yang menyimpan bentuk dasar dan nilai-nilai tiap titik akhirnya beserta nilai warna dari setiap titik.

Pengguna dapat melihat cara penggunaan program dengan menekan tombol *Help*.

Layar *Help* akan menunjukkan penjelasan *icon* pada menu *toolbar* di kiri atas, cara menggambar objek, transformasi objek, mengubah warna, menganimasi objek, menghapus objek atau titik, menyimpan, dan juga memuat objek dari penyimpanan lokal pengguna.



Gambar 4.2: Layar *Help* Pada Website

4.1.2 Pembuatan Model

Model dapat dibuat dengan memilih menu-menu yang tersedia di kiri atas. Terdapat model garis, persegi, persegi panjang, dan poligon secara berurutan. Untuk tiga model pertama, pembuatan model dapat dilakukan dengan melakukan click and drag pada canvas untuk membuat model. Untuk polygon, klik pada canvas posisi titik-titik yang akan digunakan untuk membuat poligon. Setelah selesai membuat model, klik logo cursor untuk menyelesaikan pembuatan model.

4.1.3 Transformasi Model

Untuk mentransformasi model, pilih model yang akan ditransformasi pada sisi kanan web. Anda dapat memilih keseluruhan model atau beberapa titik saja. Setelah memilih model, maka Anda dapat melakukan transformasi diantaranya adalah,

1. Translasi pada sumbu x atau sumbu y
2. Dilatasi Model
3. Memutar model
4. Shearing pada sumbu x atau sumbu y
5. Memindahkan posisi titik pada x atau y

Jika memilih pada titik dan melakukan pemindahan posisi titik, maka titik yang dipilih akan dikunci sehingga titik tersebut akan menjadi pusat dari pemindahan yang dilakukan.

Anda dapat melakukan animasi pada model yaitu animasi berputar dengan tetap memilih model secara keseluruhan (harus keseluruhan model) dan menekan tombol animate. Anda dapat melakukan animasi lebih dari satu model.

Anda dapat melakukan perubahan warna model secara keseluruhan atau pada beberapa titik saja. Pemilihan model atau titik sama seperti cara sebelumnya. Setelah itu, menggunakan color picker di sisi web, pilihlah warna yang diinginkan. Pemilihan warna pada titik dapat menghasilkan gradasi warna pada model.

Anda dapat menggabungkan dua model menjadi satu kesatuan. Pilihlah sekurang-kurangnya dua model. Setelahnya tekan tombol union untuk menggabungkan kedua model menjadi sebuah poligon yang baru.

4.1.4 Menyimpan dan Me-*load* Model

Model dapat disimpan dengan memilih model yang akan disimpan pada sisi kanan dan menekan tombol save. Model akan diunduh dalam bentuk json yang menyimpan bentuk dasar model beserta definisi titiknya. Untuk menggunakan kembali, tekan tombol load dan pilihlah model yang akan digunakan pada canvas.

Daftar Pustaka

[1] E. Angel and D. Shreiner, *Interactive Computer Graphics A Top-Down Approach with WebGL 7th ed.* Addison-Wesley, 2015.

[2] Greggmann, "Webgl fundamentals, available: https://webglfundamentals.org/."