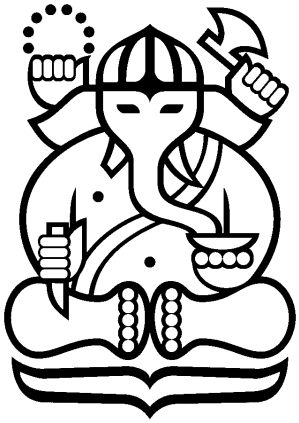
IF3260 Grafika Komputer

WEBGL PART 1:
2D PRIMITIVE ELEMENTS

Laporan Tugas Besar 1

Disusun untuk memenuhi tugas mata kuliah IF3260 Grafika Komputer
pada Semester 2 (dua) Tahun Akademik 2023/2024



Oleh

Chiquita Ahsanunnisa	13521129
Rava Maulana Azzikri	13521149
Hanif Muhammad Zhafran	13521157

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG

2024

DAFTAR ISI

BAB I
DESKRIPSI PROGRAM

WebGL (*Web Graphics Library*) merupakan suatu API JavaScript untuk rasterisasi. WebGL dapat menggambarkan grafika 2D maupun 3D berdasarkan data dan program yang ditulis oleh pengembang di dalam browser web yang kompatibel dengan WebGL. WebGL dijalankan di GPU dari sebuah komputer. Oleh karena itu, program yang dituliskan di WebGL nantinya akan dikirimkan ke GPU untuk dijalankan.

Program yang akan dijalankan berbentuk sepasang fungsi, yaitu *vertex shader* dan *fragment shader* yang ditulskan dalam bahasa C/C++. *Vertex shader* berfungsi untuk menghitung posisi dari seluruh *vertex* yang ada. Posisi dari tiap *vertex* akan digunakan oleh WebGL untuk rasterasi berbagai macam bentuk primitif yang tersedia (titik, garis, segitiga). Rasterisasi dari tiap primitif akan dibantu oleh *fragment shader*, dimana *fragment shader* akan memberikan warna untuk setiap pixel dari komponen primitif yang sedang digambar. Data warna akan terlebih dahulu dipasangkan dengan tiap *vertex*, kemudian warna pada pixel lain dalam komponen primitif akan diinterpolasi berdasarkan warna tiap *vertex*.

Fungsi-fungsi yang tersedia pada WebGL API akan digunakan untuk mengisi data yang diperlukan oleh *fragment shader* dan *vertex shader*. Terdapat 4 cara untuk mengisi data yang dibutuhkan oleh *shader*, yaitu:

1. Attributes dan Buffers
   Buffers merupakan array yang berisi data biner yang akan dikirimkan ke GPU (biasanya berisi posisi, vektor normal, warna, dsb). Attributes digunakan untuk memperjelas bagaimana data akan diambil dari buffer untuk diolah.
2. Uniforms
   Uniform merupakan variabel global yang ditetapkan sebelum shader dieksekusi.
3. Textures
   Textures merupakan array data yang dapat diakses secara random pada shader, berbeda dengan buffer yang tidak bisa diakses secara random.
4. Varyings
   Varyings merupakan salah satu cara vertex shader mengirimkan data ke fragment shader. Nilai yang dikirimkan melalui varying akan diinterpolasi saat mengeksekusi fragment shader.

Pada tugas ini, WebGL akan dimanfaatkan untuk membuat suatu aplikasi berbasis web yang dapat digunakan untuk menggambar grafika 2D dengan fitur-fitur sebagai berikut:

- Menggambar garis dengan cara drag and drop
- Mengubah panjang garis dengan cara memasukkan angka ke dalam input box
- Menggambar persegi panjang dengan cara drag and drop
- Mengubah panjang dan lebar persegi panjang dengan cara memasukkan angka ke dalam input box
- Menggambar persegi dengan cara drag and drop
- Mengubah panjang sisi persegi dengan cara memasukkan angka ke dalam input box
- Menggambar polygon dengan cara menambahkan *vertex* satu per satu tiap klik dari pengguna. Polygon yang digambar akan selalu merupakan convex hull dari titik-titiknya.
- Menambah dan menghapus vertex dari suatu polygon
- Menggerakkan vertex pada suatu bangun dengan cara drag and drop. Bentuk dari bangun akan mengalami dilatasi sesuai dengan posisi vertex yang dipindahkan
- Menggerakkan (translasi) bangun dengan cara drag and drop
- Mengubah warna dari salah satu atau semua vertex pada suatu bangun
- Memilih bangun/vertex dengan cara memencet bangun/vertex pada koordinat yang sesuai
- Memilih bangun/vertex melalui dropdown
- Menyimpan bangun dalam bentuk file JSON
- Memuat bangun dari file JSON
- Animasi transformasi dilatasi pada seluruh bangun yang ada dalam kanvas

BAB II
HASIL IMPLEMENTASI

2.1 Struktur Program

Program yang dibangun merupakan sebuah web yang *interface*-nya dibangun dengan menggunakan HTML dan CSS. Logika program dibangun menggunakan bahasa pemrograman TypeScript, yang di-*compile* menggunakan Webpack, tanpa menggunakan *framework* apapun (*vanilla*). Selain itu, karena menggunakan WebGL, web juga dibangun dengan *script* OpenGL Shading Language (GLSL) yang tertulis pada *script* HTML.

Program dibangun dengan menggunakan paradigma pemrograman berorientasi objek dan fungsional. Program berekstensi `.ts` yang diawali huruf kapital berisi kelas, sedangkan program berekstensi `.ts` yang diawali huruf nonkapital berisi modul. Berikut merupakan struktur program yang dibangun.

```
tugas-besar-grafkom-1-hanifbaik
│
├── .vscode
│   └── settings.json
│
├── doc
│   └── IF3260_Laporan Tubes 1_13521129.pdf
│
├── public
│   ├── styles
│   │   └── index.css
│   │
│   └── index.html
│
├── src
│   ├── models
│   │   ├── Line.ts
│   │   ├── Polygon.ts
│   │   ├── Rectangle.ts
│   │   └── Square.ts
│   │
│   ├── primitives
│   │   ├── Model.ts
│   │   └── Vertex.ts
│   │
│   └── utils
│       ├── program.ts
│       └── shader.ts
│
├── test
│   ├── line-test.json
```

```
│   ├── polygon-test.json
│   ├── rectangle-test.json
│   └── square-test.json
│
├── .eslintrc.cjs
├── .gitignore
├── package-lock.json
├── package.json
├── README.md
├── tsconfig.json
└── webpack.config.js
```

2.2 Detail Program

2.2.1 Kelas Vertex

Kelas Vertex merepresentasikan sebuah titik sudut (*vertex*) dalam gambar dua dimensi. Kelas ini mempunyai atribut koordinat dua dimensi beserta warna *red-green-blue* (RGB) untuk titik sudut yang direpresentasikan.

Kelas Vertex diimplementasikan pada *file* `src/primitives/Vertex.ts`. Berikut merupakan detail implementasi kelas Vertex.

Tabel 2.2.1.1 Atribut Kelas Vertex

Atribut	**Keterangan**
`coord`	Melambangkan koordinat dua dimensi (x dan y) dari titik.
`color`	Melambangkan warna RGB dari titik.
`count`	Atribut statik yang melambangkan total jumlah objek Vertex yang telah dibangun.

Tabel 2.2.1.2 Metode Kelas Vertex

Metode	**Keterangan**
`constructor`	Menerima parameter koordinat dua dimensi dan warna yang bersifat opsional. Membangun objek Vertex dari kedua parameter tersebut. Jika parameter warna tidak diisi, titik akan mempunyai warna *random*.
`getSqDistTo`	Menerima parameter objek Vertex lain. Menghitung jarak *euclidean* dari sebuah titik ke tiitk lain. Metode ini digunakan sebagai metode pembantu untuk menghitung *convex hull*.
`isEq`	Menerima parameter objek Vertex lain. Mengembalikan nilai kesamaan antara kedua titik. Kedua titik dianggap sama jika koordinatnya sama.

2.2.2 Kelas Model

Kelas Model merepresentasikan sebuah model dalam ruang dua dimensi. Kelas ini merupakan sebuah kelas abstrak yang nantinya akan diturunkan menjadi kelas model lain sesuai dengan bentuknya. Hal ini dilakukan agar program dapat memanfaatkan *polymorphism* pada konsep pemrograman berorientasi objek sehingga program lebih modular.

Kelas Model diimplementasikan pada *file* `src/primitives/Model.ts`. Berikut merupakan detail implementasi kelas Model.

Tabel 2.2.2.1 Atribut Kelas Model

Atribut	**Keterangan**
`id`	Melambangkan identitas atau nomor model.
`vertexList`	Berisi daftar seluruh objek Vertex yang membangun model. Daftar objek Vertex inilah yang akan di-*flatten* lalu dikirim ke *shader*. Urutan objek Vertex melambangkan urutan penggambaran yang akan dilakukan oleh *shader*.
`transformMat`	Merupakan matriks transformasi dari objek. Matriks ini diinisialisasi nilainya dengan matriks identitas.
`rightmostX`	Melambangkan bagian x dari koordinat titik yang paling kanan dari model.
`leftmostX`	Melambangkan bagian x dari koordinat titik yang paling kiri dari model.
`topmostY`	Melambangkan bagian y dari koordinat titik yang paling atas dari model.
`bottommostY`	Melambangkan bagian y dari koordinat titik yang paling bawah dari model.

Tabel 2.2.2.2 Metode Kelas Model

Metode	**Keterangan**
`constructor`	Menerima parameter id dan membangun objek Model dari id tersebut.
`getPosArray`	Mengembalikan larik dari koordinat setiap objek pada atribut vertexList yang sudah di-*flatten*. Larik inilah yang nantinya akan dikirim ke *vertex shader*.
`getColorArray`	Mengembalikan larik dari warna setiap objek pada atribut vertexList yang sudah di-*flatten*. Larik inilah yang nantinya akan dikirim ke *fragment shader*.
`getTransformMatArray`	Mengembalikan matriks transformasi model yang sudah di-*flatten*. Larik inilah yang nantinya akan dikirim ke *vertex shader* untuk mentransformasikan *vertex*-*vertex*.
`getDrawMethod`	Metode abstrak yang mengembalikan metode gambar WebGL.
`addVertex`	Menerima parameter objek Vertex. Secara *default*, berfungsi untuk menambahkan objek Vertex baru ke dalam vertexList. Namun, metode ini dapat di-*override* oleh

	kelas turunan untuk menyesuaikan penambahan titik dengan model yang digunakan.
`getRightmostX`	*Getter* untuk atribut rightmostX.
`getLeftmostX`	*Getter* untuk atribut leftmostX.
`getTopmostY`	*Getter* untuk atribut topmostY.
`getBottommostY`	*Getter* untuk atribut bottommostY.
`setVertexList`	*Setter* untuk atribut vertexList. Meng-*assign* atribut rightmostX, leftmostX, topmostY, dan bottomostY menjadi nilai yang baru sesuai dengan vertexList yang baru.
`translate`	Menerima parameter tx dan ty. Mengalikan atribut transformMat dengan matriks translasi yang dibangun dari tx dan ty.
`resetTranslate`	Mentranslasikan semua titik ke *clip space* dan mengaplikasikan matriks transformasi dari hasil translasi ke semua titik.
`scale`	Menerima parameter sx dan sy. Mengalikan atribut transformMat dengan matriks dilatasi yang dibangun dari sx dan sy.
`resetScale`	Mentranslasikan semua titik ke *clip space* dan mengaplikasikan matriks transformasi dari hasil dilatasi ke semua titik.
`render`	Me-*render* model ke kanvas HTML dengan cara mengirim *vertex*, warna, dan matriks transformasi ke *shader*.

2.2.3 Kelas Line

Kelas ini merupakan kelas turunan dari kelas Model. Kelas ini merepresentasikan sebuah garis dalam ruang dua dimensi.

Kelas Line diimplementasikan pada *file* `src/models/Line.ts`. Berikut merupakan detail implementasi kelas Line.

Tabel 2.2.3.1 Atribut Kelas Line

Atribut	**Keterangan**
`vertexRef`	Titik yang menjadi acuan saat menggambar garis dengan cara men-*drag* kanvas.
`length`	Melambangkan panjang dari garis.
`count`	Atribut statik yang melambangkan total jumlah objek Line yang telah dibangun.

Tabel 2.2.3.2 Metode Kelas Line

Metode	**Keterangan**

`constructor`	Menerima parameter vertexRef (sebuah objek Vertex) yang bersifat opsional. Membangun objek Line dari parameter tersebut. Jika parameter vertexRef tidak diisi, vertexRef akan diinisialisasi dengan titik (0, 0).
`fromObject`	Metode statik yang menerima parameter objek TypeScript (JavaScript) yang akan diubah menjadi objek Line.
`addVertex`	Metode kelas Model yang di-*override*. Menerima parameter objek Vertex. Menambahkan objek Vertex baru ke dalam vertexList.
`getDrawMethod`	Metode kelas Model yang di-*override*. Mengembalikan metode menggambar objek Line, yaitu gl.LINE.
`getVertexRef`	*Getter* untuk atribut vertexRef.
`updateVerticesWhenDrawing`	Metode untuk memperbarui vertexList saat menggambar garis dengan cara men-*drag* kanvas.

2.2.4 Kelas Square

Kelas ini merupakan kelas turunan dari kelas Model. Kelas ini merepresentasikan sebuah persegi dalam ruang dua dimensi.

Kelas Square diimplementasikan pada *file* `src/models/Square.ts`. Berikut merupakan detail implementasi kelas Square.

Tabel 2.2.4.1 Atribut Kelas Square

Atribut	**Keterangan**
`vertexRef`	Titik yang menjadi acuan saat menggambar persegi dengan cara men-*drag* kanvas.
`size`	Melambangkan panjang sisi dari persegi.
`count`	Atribut statik yang melambangkan total jumlah objek Square yang telah dibangun.

Tabel 2.2.4.2 Metode Kelas Square

Metode	**Keterangan**
`constructor`	Menerima parameter vertexRef (sebuah objek Vertex) yang bersifat opsional. Membangun objek Square dari parameter tersebut. Jika parameter vertexRef tidak diisi, vertexRef akan diinisialisasi dengan titik (0, 0).
`fromObject`	Metode statik yang menerima parameter objek TypeScript (JavaScript) yang akan diubah menjadi objek Square.
`addVertex`	Metode kelas Model yang di-*override*. Menerima parameter objek Vertex. Menambahkan objek Vertex baru ke dalam vertexList.
`getDrawMethod`	Metode kelas Model yang di-*override*. Mengembalikan metode menggambar

	objek Square, yaitu gl.TRIANGLE_FAN. Metode penggambaran ini dipilih karena jumlah *vertex* yang harus dikirim dan digambar lebih sedikit dibanding metode lainnya.
`getVertexRef`	*Getter* untuk atribut vertexRef.
`setVertexRef`	*Setter* untuk koordinat atribut vertexRef.
`updateVerticesWhenDrawing`	Metode untuk memperbarui vertexList saat menggambar persegi dengan cara men-*drag* kanvas.

2.2.5 Kelas Rectangle

Kelas ini merupakan kelas turunan dari kelas Model. Kelas ini merepresentasikan sebuah persegi panjang dalam ruang dua dimensi.

Kelas Rectangle diimplementasikan pada *file* `src/models/Rectangle.ts`. Berikut merupakan detail implementasi kelas Rectangle.

Tabel 2.2.5.1 Atribut Kelas Rectangle

Atribut	**Keterangan**
`vertexRef`	Titik yang menjadi acuan saat menggambar persegi panjang dengan cara men-*drag* kanvas.
`width`	Melambangkan lebar dari persegi panjang.
`height`	Melambangkan tinggi dari persegi panjang.
`count`	Atribut statik yang melambangkan total jumlah objek Rectangle yang telah dibangun.

Tabel 2.2.5.2 Metode Kelas Rectangle

Metode	**Keterangan**
`constructor`	Menerima parameter vertexRef (sebuah objek Vertex) yang bersifat opsional. Membangun objek Rectangle dari parameter tersebut. Jika parameter vertexRef tidak diisi, vertexRef akan diinisialisasi dengan titik (0, 0).
`fromObject`	Metode statik yang menerima parameter objek TypeScript (JavaScript) yang akan diubah menjadi objek Rectangle.
`addVertex`	Metode kelas Model yang di-*override*. Menerima parameter objek Vertex. Menambahkan objek Vertex baru ke dalam vertexList.
`getDrawMethod`	Metode kelas Model yang di-*override*. Mengembalikan metode menggambar objek Rectangle, yaitu gl.TRIANGLE_FAN. Metode penggambaran ini dipilih karena jumlah *vertex* yang harus dikirim dan digambar lebih sedikit dibanding metode lainnya.

`getWidth`	*Getter* untuk atribut width.
`getHeight`	*Getter* untuk atribut height.
`getVertexRef`	*Getter* untuk atribut vertexRef.
`setVertexRef`	*Setter* untuk koordinat atribut vertexRef.
`getRelativeWidth`	Menghitung lebar relatif dari objek Rectangle.
`getRelativeHeight`	Menghitung tinggi relatif dari objek Rectangle.
`restoreVertexRef`	Mengeset vertexRef ke titik sudut yang paling kiri atas dari persegi panjang.
`updateVerticesWhenDrawing`	Metode untuk memperbarui vertexList saat menggambar persegi panjang dengan cara men-*drag* kanvas.

2.2.6 Kelas Polygon

Kelas ini merupakan kelas turunan dari kelas Model. Kelas ini merepresentasikan sebuah poligon dalam ruang dua dimensi.

Kelas Polygon diimplementasikan pada *file* `src/models/Polygon.ts`. Berikut merupakan detail implementasi kelas Polygon.

Tabel 2.2.5.1 Atribut Kelas Polygon

Atribut	**Keterangan**
`polarRef`	Titik yang menjadi acuan koordinat polar relatif saat menghitung *convex hull*.
`count`	Atribut statik yang melambangkan total jumlah objek Polygon yang telah dibangun.

Tabel 2.2.5.2 Metode Kelas Polygon

Metode	**Keterangan**
`constructor`	Mengkonstruksi objek Polygon. Mempunyai parameter opsional yaitu polygon (sebuah objek Polygon) yang jika diisi akan mengkonstruksi objek Polygon dari parameter tersebut.
`fromObject`	Metode statik yang menerima parameter objek TypeScript (JavaScript) yang akan diubah menjadi objek Polygon.
`addVertex`	Metode kelas Model yang di-*override*. Menerima parameter objek Vertex. Menambahkan objek Vertex baru ke dalam vertexList sesuai dengan aturan *convex hull* untuk poligon.
`deleteVertex`	Menerima parameter objek Vertex. Menghapus objek Vertex dari vertexList sesuai dengan aturan *convex hull* untuk poligon.
`getDrawMethod`	Metode kelas Model yang di-*override*. Mengembalikan metode menggambar objek

	Polygon. Jika poligon masih berupa garis, metode yang digunakan yaitu gl.LINE. Selain itu, metode yang digunakan adalah gl.TRIANGLE_FAN. Metode penggambaran ini dipilih karena jumlah *vertex* yang harus dikirim dan digambar lebih sedikit dibanding metode lainnya.
`getOrientation`	Menentukan orientasi pergerakan dari sebuah titik ke titik lain relatif terhadap polarRef. Orientasi dapat berupa kolinear, searah jarum jam, maupun berlawanan arah jarum jam.
`cmpPolar`	Membandingkan urutan koordinat polar relatif yang digunakan untuk proses pengurutan pada algoritma Graham Scan.
`convexHull`	Menyesuaikan vertexList agar sesuai dengan aturan *convex hull* untuk poligon. Algoritma *convex hull* yang digunakan yaitu algoritma Graham Scan.

2.2.7 Modul Shader

Modul Shader berisi fungsi utilitas untuk membuat *shader*, baik *vertex shader* maupun *fragment shader*, yang nantinya akan digunakan di program utama. Modul ini diimplementasikan pada *file* `src/utils/shader.ts`. Berikut merupakan detail implementasi modul Shader.

Tabel 2.2.7.1 Fungsi Modul Shader

Fungsi	**Keterangan**
`createShader`	Fungsi yang menerima parameter konteks *rendering* WebGL, tipe (*vertex shader* atau *fragment shader*), dan sumber kode *shader* lalu membuat *shader* berdasarkan parameter tersebut.

2.2.8 Modul Program

Modul Program berisi fungsi utilitas untuk membuat program WebGL yang terdiri atas *vertex shader* dan *fragment shader*. Modul ini diimplementasikan pada *file* `src/utils/program.ts`. Berikut merupakan detail implementasi modul Program.

Tabel 2.2.8.1 Fungsi Modul Program

Fungsi	**Keterangan**
`createProgram`	Fungsi yang menerima parameter konteks *rendering* WebGL, *vertex shader*, dan *fragment shader* lalu membuat program WebGL berdasarkan parameter tersebut.

2.2.9 WebGL Shader

WebGL *shader* yang digunakan terdiri dari *vertex shader* dan *fragment shader*. Keduanya ditulis dalam *script* terpisah dalam bahasa OpenGL Shading Language (GLSL) yang tertulis dalam *script* HTML. Kedua *shader* ini diimplementasikan pada *file* `public/index.html`.

2.2.10 Program Utama

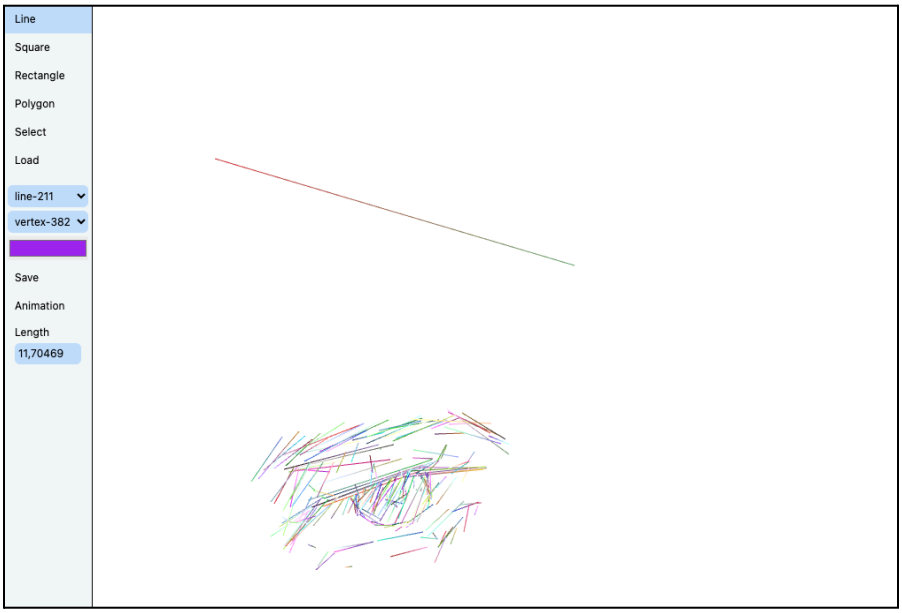
Program utama menangani seluruh logika interaksi program dengan pengguna, mulai dari membuat model, mengkustomisasi model, hingga menampilkannya pada kanvas. Program utama ini diimplementasikan pada *file* `src/index.ts`. *Interface* web yang dikaitkan dengan program utama diimplementasikan pada *file* `public/index.html`.

BAB III

MANUAL

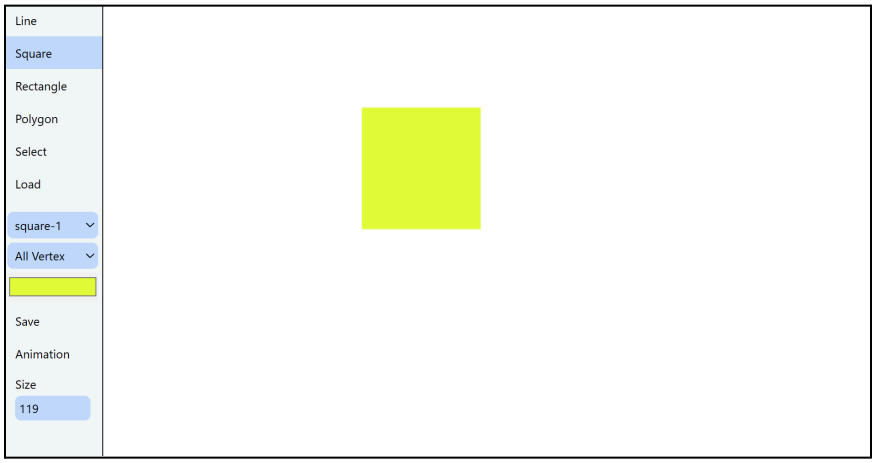
3.1 Membuat Garis

1. Tekan opsi ‘Line’ pada sidebar.
2. Drag cursor pada canvas untuk membuat garis.



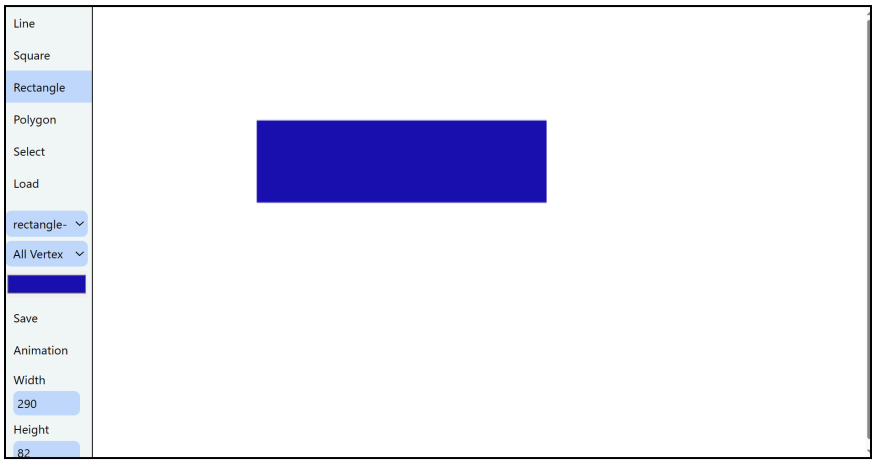
3.2 Membuat Persegi

1. Tekan opsi ‘Square’ pada sidebar.
2. Drag cursor pada canvas untuk membuat persegi.



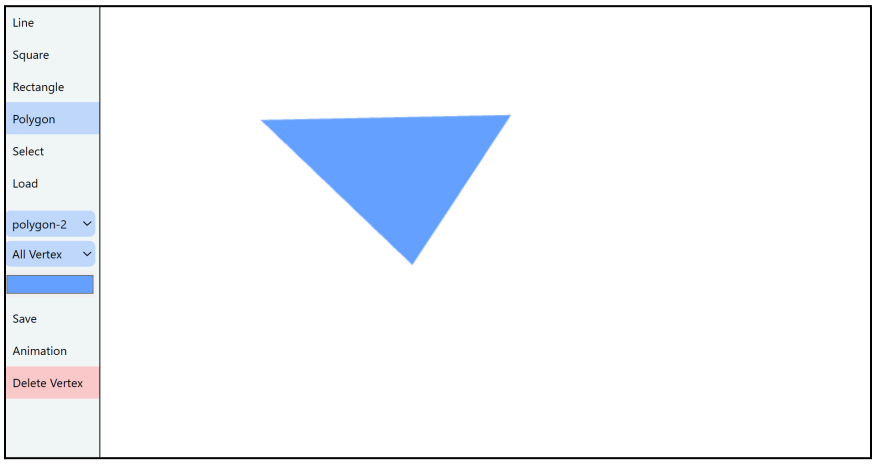
3.3 Membuat Persegi Panjang

1. Tekan opsi ‘Rectangle pada sidebar.
2. Drag cursor pada canvas untuk membuat persegi panjang.



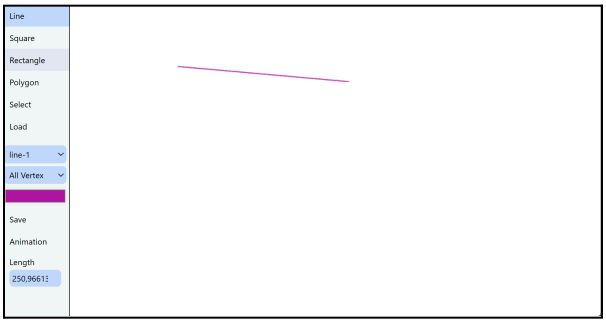
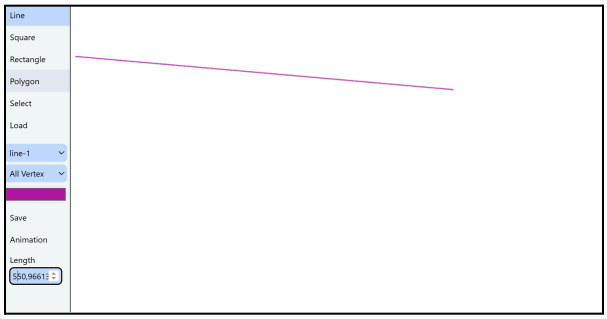
3.4 Membuat Poligon

1. Tekan opsi ‘Polygon’ pada sidebar.
2. Tekan cursor pada canvas untuk menambahkan titik sudut.
3. Ulangi langkah kedua untuk menambahkan jumlah titik sudut pada poligon.
4. Tekan kembali opsi ‘Polygon’ pada sidebar untuk menyimpan poligon pada canvas.



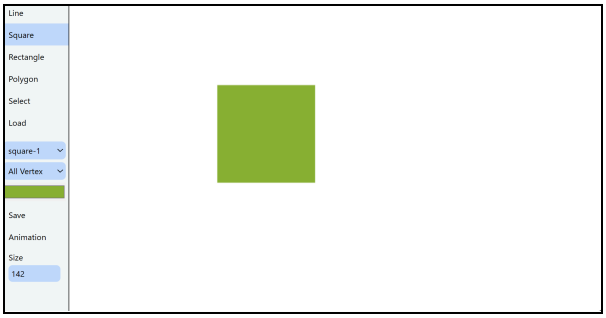
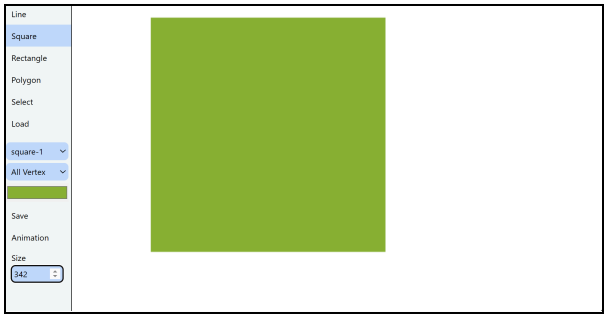
3.5 Mengubah Panjang Garis

1. Pilih garis yang panjangnya ingin diubah menggunakan dropdown pada sidebar.
2. Gunakan input ‘Length’ pada sidebar untuk mengubah panjang garis.

Sebelum	Sesudah
	

3.6 Mengubah Panjang Sisi Persegi

1. Pilih persegi yang panjang sisinya ingin diubah menggunakan dropdown pada sidebar.
2. Gunakan input ‘Size’ pada sidebar untuk mengubah panjang sisi persegi.

Sebelum	Sesudah
	

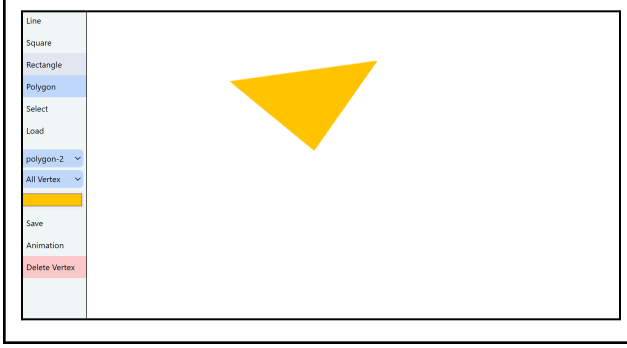
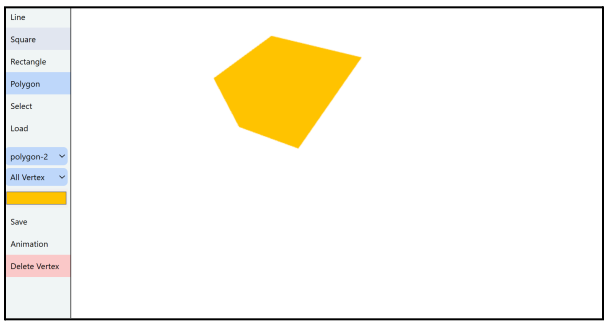
3.7 Mengubah Lebar dan/atau Tinggi Persegi Panjang

1. Pilih persegi panjang yang lebar dan/atau tingginya ingin diubah menggunakan dropdown pada sidebar.
2. Gunakan input ‘Width’ dan ‘Height’ pada sidebar untuk mengubah lebar dan/atau tinggi persegi panjang.

Sebelum	Sesudah

3.8 Menambahkan Titik Sudut Poligon

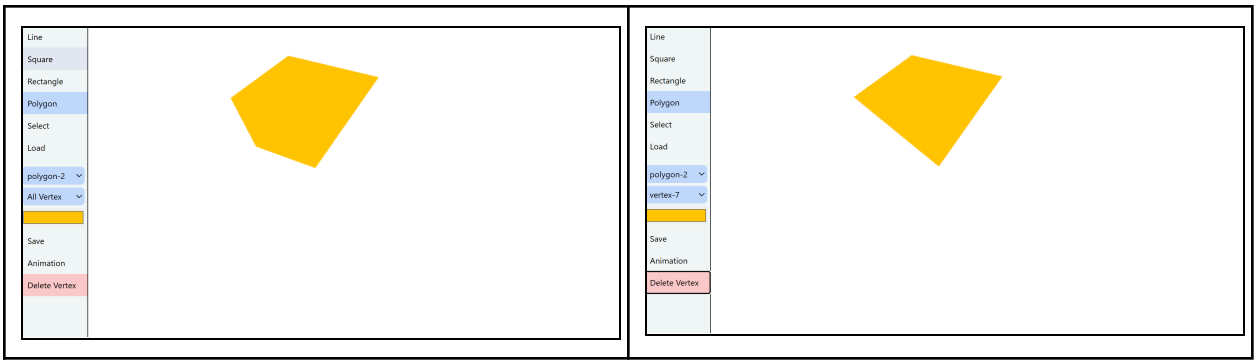
1. Pilih poligon yang titik sudutnya ingin ditambah menggunakan dropdown pada sidebar.
2. Tekan cursor pada canvas untuk untuk menambahkan titik sudut baru.

Sebelum	Sesudah
	

3.9 Menghapus Titik Sudut Poligon

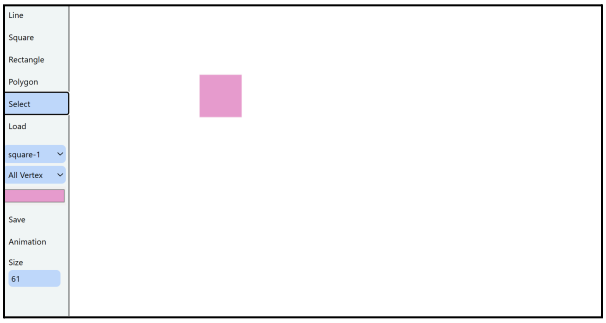
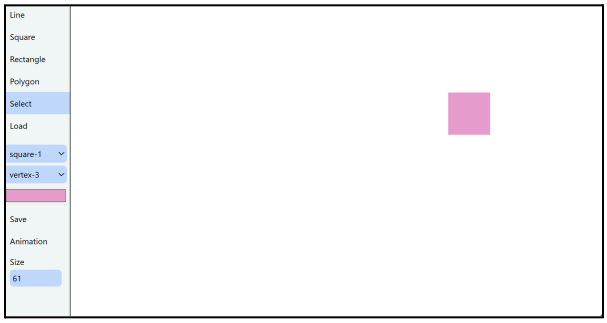
1. Pilih poligon yang titik sudutnya ingin dihapus menggunakan dropdown pada sidebar.
2. Pilih titik sudut yang ingin dihapus menggunakan dropdown pada sidebar.
3. Tekan tombol ‘Delete Vertex’ untuk menghapus titik sudut.

Sebelum	Sesudah



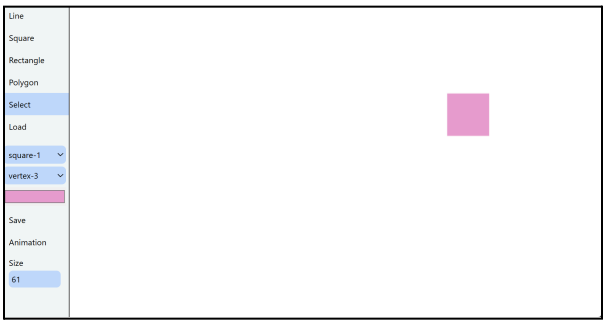
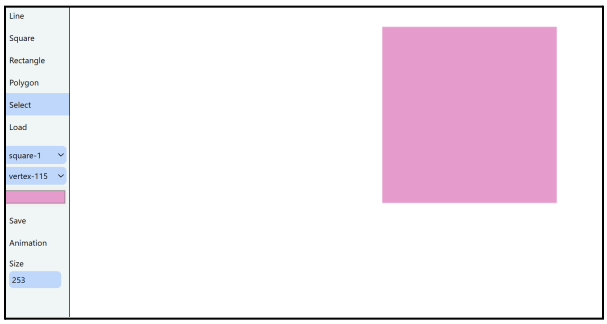
3.10 Translasi Model

1. Pilih opsi ‘Select’ pada sidebar untuk mengaktifkan fitur transformasi.
2. Drag model manapun untuk melakukan translasi terhadap model tersebut.

Sebelum	Sesudah
	

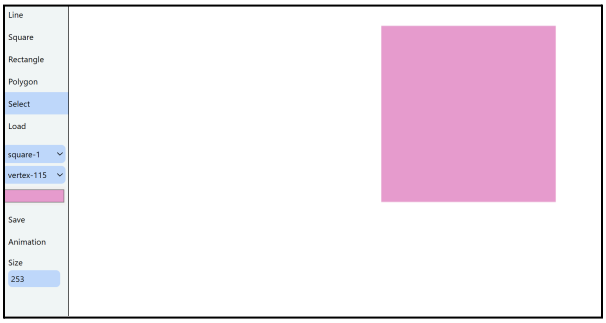
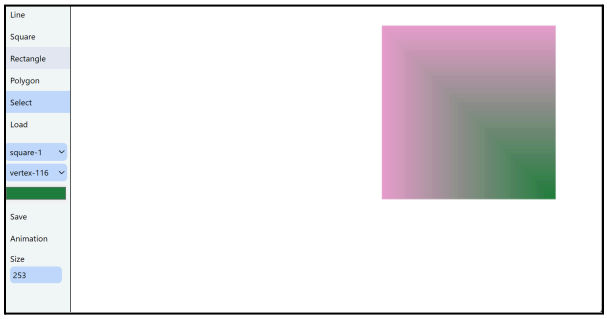
3.11 Dilatasi Model

1. Pilih opsi ‘Select’ pada sidebar untuk mengaktifkan fitur transformasi.
2. Drag salah satu titik sudut model untuk melakukan dilatasi terhadap model tersebut.

Sebelum	Sesudah
	

3.12 Mengubah Warna Titik Sudut

1. Pilih model yang ingin diubah warna titik sudutnya menggunakan dropdown pada sidebar.
2. Pilih titik sudut yang warnanya ingin diubah menggunakan dropdown pada sidebar.
3. Gunakan color picker untuk mengubah warna titik sudut.

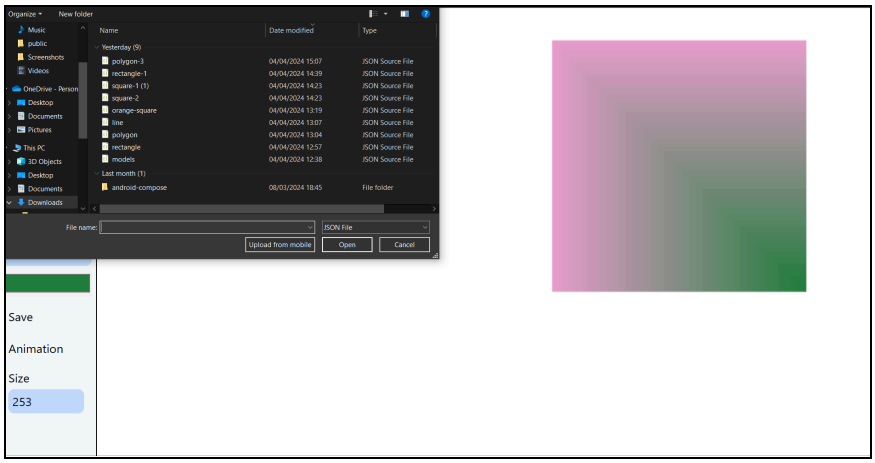
Sebelum	Sesudah
	

3.13 Save Model

1. Pilih model yang ingin disimpan menggunakan dropdown pada sidebar.
2. Tekan tombol ‘Save’ untuk menyimpan model.

3.14 Load Model

1. Tekan tombol ‘Load’ pada sidebar untuk memilih file model yang akan di-load.



3.15 Menampilkan Animasi

1. Tekan tombol ‘Animate’ pada sidebar untuk menampilkan animasi pada seluruh model.
2. Tekan tombol ‘Animate’ kembali untuk menghentikan animasi.

DAFTAR PUSTAKA

WebGL Fundamentals. (n.d.). WebGL Fundamentals. Diakses dari *https://webglfundamentals.org/* pada 20 Maret 2024.

LAMPIRAN

Lampiran 1 Pranala ke Repositori Github

https://github.com/GAIB20/tugas-besar-grafkom-1-hanifbaik

Lampiran 2 Pembagian Tugas

NIM	Nama	Tugas
13521129	Chiquita Ahsanunnisa	● Implementasi menggambar poligon ● Transformasi geometri: translasi ● Implementasi algoritma untuk menggambar poligon dengan *convex hull*
13521149	Rava Maulana Azzikri	● Implementasi menggambar garis dan persegi ● Transformasi geometri: dilatasi ● *Save* model
13521157	Hanif Muhammad Zhafran	● Implementasi menggambar persegi panjang ● Menggerakkan titik sudut dengan slider atau *drag and drop* ● Mengubah warna salah satu atau semua titik sudut ● Integrasi animasi pada salah satu fitur