IF3260 Grafika Komputer

WebGL Part 1: 2D Primitive Elements

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

Oleh

Fakhri Muhammad Mahendra	13521045
Razzan Daksana Yoni	13521087
Muhamad Aji Wibisono	13521095

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

2024

DAFTAR ISI

BAB I
SPESIFIKASI

Tugas besar ini akan menjadi pengenalan terhadap indahnya dunia grafika komputer dan melihatkan secara konkrit bagaimana permainan video tiga dimensi yang sering kamu mainkan itu di-render dalam layar monitor dua dimensi. Target platform yang akan digunakan adalah WebGL. Berikut adalah cakupan materi yang akan digunakan pada tubes ini:

- WebGL 2D
- Line, Polygon
- Transformations (translation, rotation, dilation)
- Geometry
- Input and Output

Implementasi juga mengerjakan beberapa bonus yang ada yaitu berupa:

- Convex Hull pada *polygon*
- Animasi pada bangun datar
- Animasi pada vertex (marker)
- Mengkalkulasi titik berat pada *polygon* untuk rotasi

BAB II
DESKRIPSI PROGRAM

2.1. Struktur Program

Berikut merupakan struktur dari program hasil implementasi WebGL Part 1: 2D Primitive Elements.

```
tugas-besar-grafkom-1-sukasamakamu
├── ...
├── public
├── src
│   ├── controller
│   │   ├── canvas-controller.ts
│   │   ├── mouse-controller.ts
│   │   ├── ui-controller.ts
│   │   └── webgl-controller.ts
│   ├── models
│   │   ├── base-model.ts
│   │   ├── line-model.ts
│   │   ├── polygon-model.ts
│   │   ├── rectangle-model.ts
│   │   └── square-model.ts
│   ├── shaders
│   │   ├── fragment-shader-2d.frag
│   │   └── vertex-shader-2d.vert
│   ├── types
│   │   ├── enum
│   │   │   ├── animate-type.ts
│   │   │   ├── buffer-type.ts
│   │   │   └── model-state.ts
│   │   ├── events
│   │   │   ├── canvas-mouse-event.ts
│   │   │   └── web-gl-events.ts
│   │   ├── buffer-info.ts
│   │   ├── color.ts
│   │   ├── coordinates.ts
│   │   ├── export-data.ts
│   │   └── setter.ts
│   ├── util
│   │   ├── convex-hull.ts
│   │   ├── m4.ts
│   │   └── math-util.ts
│   ├── config.ts
│   ├── main.ts
│   └── vite-env.d.ts
├── index.html
├── package.json
└── tsconfig.json
```

Program diimplementasikan dengan paradigma pemrograman berorientasi objek. Kelas-kelas dibagi berdasarkan tanggung jawabnya, kelas-kelas pada folder *controller* bertanggung jawab untuk mengatur kelas lain yang dimiliki seperti canvas-controller akan mengatur fungsionalitas dari canvas, kelas-kelas pada folder *models* merupakan kelas yang berupa model yang diimplementasikan pada canvas seperti *line*, *polygon*, *rectangle*, dan *square* serta model pembantu yaitu marker, pada folder lain yaitu *types*, *enum, events*, dan *utils* merupakan kelas-kelas pembantu agar fungsi-fungsi yang digunakan dapat di-reuse.

2.2. Detail Program Utama

a. Kelas CanvasController

Kelas `CanvasController` mengatur segala komponen yang ada pada *canvas* dari penyimpanan *instance* hingga penggambaran ke *canvas*.

Kelas ini diimplementasikan dengan detail implementasi sebagai berikut.

Tabel 2.2.1.1 Atribut Kelas CanvasController

Atribut	Keterangan
`canvas`	`canvas` yang berupa `HTMLCanvasElement` sebagai tempat untuk menampilkan model-model yang ada.
`glController`	`webGLController` yang berupa instance dari kelas `WebGlController`, yang diperlukan untuk penggambaran model pada canvas
`modelBuffer`	`Buffer` yang menyimpan `model` dan `key` yang ada pada `canvas`.
`modelMapKey`	*Identifier autoincrement* sebagai pembeda untuk suatu *instance* terhadap *instance* lain
`markerBuffer`	`Buffer` yang menyimpan `marker` dan `key` yang ada pada `canvas`.
`markerMapKey`	*Identifier autoincrement* sebagai pembeda untuk suatu `marker` terhadap `marker` lain
`lerpCode`	Sebagai alat bantu untuk animasi dengan menggunakan metode *lerp*.

Tabel 2.2.1.2 Metode Kelas CanvasController

Metode	Keterangan
`constructor()`	Menerima parameter `id` yang merupakan `id` dari elemen `canvas` pada file `index.html` yang akan digunakan untuk mengelola `canvas`.
`unsetModel()`	Menghapus `model` dari `modelBuffer.`
`setModel()`	Mengubah `model` menggunakan `key` sebagai *identifier*`.`
`setMarker()`	Menambahkan ataupun mengubah `marker` menggunakan `key` sebagai *identifier*`.`

`getModel()`	Mendapatkan `model` yang ada pada `modelBuffer` menggunakan `key`.
`getMarker()`	Mendapatkan `marker` yang ada pada `markerBuffer` menggunakan `key`.
`addModel()`	Menambahkan `model` pada `modelBuffer` dengan menambahkan `key` sebagai *identifier* dari model tersebut.
`updateModel()`	Meng-*update* `model` pada `modelBuffer` dengan menambahkan `key` sebagai *identifier* dari model tersebut.
`removeModel()`	Mengurangi `model` dari `modelBuffer` dengan menggunakan `key`.
`clear()`	Menghapus semua *object* ada pada `canvas.`
`clearMarker()`	Menghapus semua `marker` ada pada `canvas.`
`detectMarker()`	Mendeteksi `marker` pada *object* yang sedang difokuskan.
`save()`	*Export* semua *object* dan kondisinya yang ada pada `canvas`.
`load()`	*Load* semua *object* dan kondisinya dan ditampilkan pada `canvas.`
`animateModel()`	Fungsi tambahan untuk menampilkan animasi pada `model`.
`draw()`	Menggambarkan untuk menampilkan `model` pada `canvas`.

b. Kelas MouseController

Kelas `MouseController` sebagai *mouse state management* yang akan diaplikasikan pada *canvas*.

Kelas ini diimplementasikan dengan detail implementasi sebagai berikut.

Tabel 2.2.2.1 Atribut Kelas MouseController

Atribut	**Keterangan**
`state`	`state ModelType` yang aktif pada saat ini yang digunakan untuk penggambaran.
`currentModelKey`	`Key` dari suatu *instance* `model` yang sedang difokuskan.
`currentMarkerKey`	`Key` dari suatu *instance* `marker` yang sedang difokuskan.
`glWin`	`glWin` yang berupa instance dari kelas `CanvasController`, yang diperlukan untuk penggambaran pengelolaan *instance* pada canvas
`buffer`	Berisikan `coordinate` yang sedang di-klik oleh *user*
`hoverMarkerKey`	`Key` dari suatu `marker` yang sedang di-*hover*
`clickBlocked`	Atribut pembantu apakah bisa meng-klik pada *canvas* atau tidak

Tabel 2.2.2.2 Metode Kelas MouseController

Metode	Keterangan
`constructor()`	Menerima parameter `glWin` yang merupakan `glWin` untuk mengatur penggambaran *instance* pada *canvas*.
`setCurrentMarkerKey()`	Menjadikan suatu *instance* dari marker sebagai marker saat ini
`setCurrentModelKey()`	Menjadikan suatu *instance* dari model sebagai model saat ini
`reset()`	Mereset isi *buffer of coordinates*
`handleClick()`	Click *management* pada *canvas*
`handleHover()`	Hover *management* pada *canvas*
`setFocusModel()`	Memfokuskan suatu `model`
`restoreFocusModel()`	Mengembalikan fokus terhadap suatu model
`setFocusMarker()`	Memfokuskan suatu `marker`
`removeMarker()`	Menghapus marker maupun vertex pada *polygon*
`removeModel()`	Menghapus model
`getModelBufferData()`	Mendapatkan data dari model yaitu berupa color dan position

c. Kelas UIController

Kelas `UItroller` sebagai *mouse state management* yang akan diaplikasikan pada *canvas* juga mengatur *behaviour* setiap komponen pada *html*.

Kelas ini diimplementasikan dengan detail implementasi sebagai berikut.

Tabel 2.2.3.1 Metode Kelas UIController

Metode	Keterangan
`constructor()`	Menerima parameter `glWin` dan `mouseController` yang merupakan `glWin` untuk mengatur penggambaran *instance* pada *canvas* dan mengatur *state* dari mouse.

d. Kelas WebGlController

Kelas `WebGlController` sebagai media komunikasi secara langsung ke webgl untuk melakukan proses penggambaran pada *canvas*.

Kelas ini diimplementasikan dengan detail implementasi sebagai berikut.

Tabel 2.2.4.1 Atribut Kelas WebGlController

Atribut	**Keterangan**
`canvas`	`canvas` yang berupa `HTMLCanvasElement` sebagai tempat untuk menampilkan model-model yang ada
`gl`	`WebGLRenderingContext` sebagai komunikasi langsung terhadap webGL
`vertexShader`	WebGLShader
`fragmentShader`	WebGLShader
`program`	WebGLProgram
`positionAttribLocation`	Untuk mendapatkan atribut posisi pada *buffer*
`colorAttribLocation`	Untuk mendapatkan atribut *color* pada *buffer*
`positionBuffer`	WebGLBuffer
`colorBuffer`	WebGLBuffer
`uniformSetters`	Berisikan fungsi untuk mengatur *uniform*.

Tabel 2.2.4.2 Metode Kelas UIController

Metode	**Keterangan**
`constructor()`	Menerima parameter `id` yang merupakan `id` dari elemen `canvas` pada file `index.html` yang akan digunakan untuk mengelola `canvas`.
`draw()`	Menggambar untuk menampilkan objek pada *canvas*
`createShader()`	Membuat shader yang ke webGl
`createProgram()`	Membuat program pada webGl
`setPosition()`	Menerima BufferInfo yang berupa position dan diatur pada WebGL
`setColor()`	Menerima BufferInfo yang berupa color dan diatur pada WebGL
`setUniforms()`	Mengubah uniform
`createUniformSetters()`	Menambahkan uniformSetters
`resizeCanvasToDisplaySize()`	Mengatur ukuran canvas

e. Kelas BaseModel

Kelas `BaseModel` merepresentasikan sebuah model pada *canvas*. Kelas ini merupakan abstraksi kelas yang nantinya diturunkan menjadi kelas *model* sesuai dengan *model* yang ditampilkan.

Kelas ini diimplementasikan dengan detail implementasi sebagai berikut.

Tabel 2.2.5.1 Atribut Kelas BaseModel

Atribut	**Keterangan**
`type`	Tipe model pada objek
`positionBuffer`	Buffer yang berisikan position untuk model tersebut
`colorBuffer`	Buffer yang berisikan color untuk model tersebut
`uniforms`	Uniform pada model tersebut
`x_translation`	Jarak sumbu x yang ditranslasikan
`y_translation`	Jarak sumbu y yang ditranslasikan
`z_rotation`	Besar rotasi yang dirotasikan berupa degree
`width`	Besar lebar dari suatu model yang ditambahkan
`length`	Besar panjang dari suatu model yang ditambahkan

Tabel 2.2.5.2 Metode Kelas BaseModel

Metode	**Keterangan**
`getBufferData()`	Mendapatkan BufferData yang berupa position ataupun color
`clone()`	Clone model baru
`getCenter()`	Mendapatkan coordinate dari *center of an object*
`generateUniform()`	Menghasilkan uniform baru setelah ditransformasikan
`moveVertex()`	Memindahkan suatu vertex pada objek

f. Kelas LineModel

Kelas ini merupakan kelas turunan dari kelas `BaseModel`. Kelas ini merepresentasikan model pada *canvas* yang berupa garis.

Kelas ini diimplementasikan dengan detail implementasi sebagai berikut.

Tabel 2.2.6.1 Metode Kelas LineModel

Metode	Keterangan
`constructor()`	Mengonstruksikan model line baru

g. Kelas MarkerModel

Kelas ini merupakan kelas turunan dari kelas `BaseModel`. Kelas ini merepresentasikan model pada *canvas* yang berupa *marker* yang berguna untuk mengatur titik sudut pada model.

Kelas ini diimplementasikan dengan detail implementasi sebagai berikut.

Tabel 2.2.7.1 Atribut Kelas MarkerModel

Atribut	Keterangan
`ghost`	Marker kosong
`index`	Index marker pada model
`color`	Warna pada marker tersebut
`active`	Apakah marker tersebut active atau tidak

Tabel 2.2.7.2 Metode Kelas MarkerModel

Metode	Keterangan
`constructor()`	Mengonstruksi marker model baru
`isInside()`	Menentukan apakah marker terdapat di dalam model atau tidak
`clone()`	Clone marker baru
`setColor()`	Mengubah color pada marker
`highlight()`	Men-highlight marker
`unhighlight()`	Unhighlight marker
`setActive()`	Mengatur marker sebagai marker yang aktif
`isActive()`	Menentukan apakah marker sedang aktif atau tidak

h. Kelas PolygonModel

Kelas ini merupakan kelas turunan dari kelas `BaseModel`. Kelas ini merepresentasikan model pada *canvas* yang berupa poligon.

Kelas ini diimplementasikan dengan detail implementasi sebagai berikut.

Tabel 2.2.3.2 Metode Kelas PolygonModel

Metode	Keterangan
`constructor()`	Mengonstruksikan model poligon baru
`convexHull()`	Mengatur polygon model agar menjadi convex hull

i. Kelas RectangleModel

Kelas ini merupakan kelas turunan dari kelas `BaseModel`. Kelas ini merepresentasikan model pada *canvas* yang berupa persegi panjang.

Kelas ini diimplementasikan dengan detail implementasi sebagai berikut.

Tabel 2.2.3.2 Metode Kelas RectangleModel

Metode	Keterangan
`constructor()`	Mengonstruksikan model persegi panjang baru

j. Kelas SquareModel

Kelas ini merupakan kelas turunan dari kelas `BaseModel`. Kelas ini merepresentasikan model pada *canvas* yang berupa persegi.

Kelas ini diimplementasikan dengan detail implementasi sebagai berikut.

Tabel 2.2.3.2 Metode Kelas UIController

Metode	Keterangan
`constructor()`	Mengonstruksikan square model baru

k. Kelas Lainnya

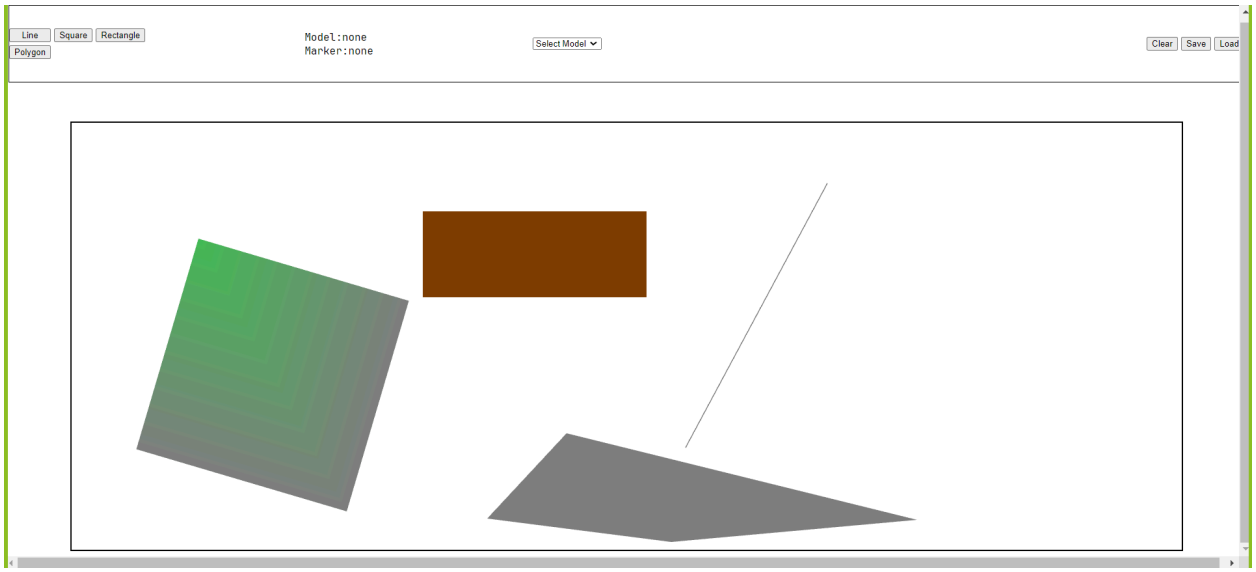
Kelas lainnya hanya berisikan kelas-kelas pembantu seperti typing.

BAB III
HASIL PENGUJIAN

Berikut merupakan hasil pengujian yang dilakukan pada berbagai fitur yang ada pada tugas besar kami, digunakan fitur perubahan warna, rotasi, scaling, translasi, berbagai bentuk (line, square, rectangle, dan polygon) serta fitur *load* dan *save.* Hasil dari *save file* kedua gambar dibawah bisa dilihat pada repository folder test.

1. Dua Pohon (test/dua-pohon.json)

2. Bangun Datar (test/bangun-datar.json)

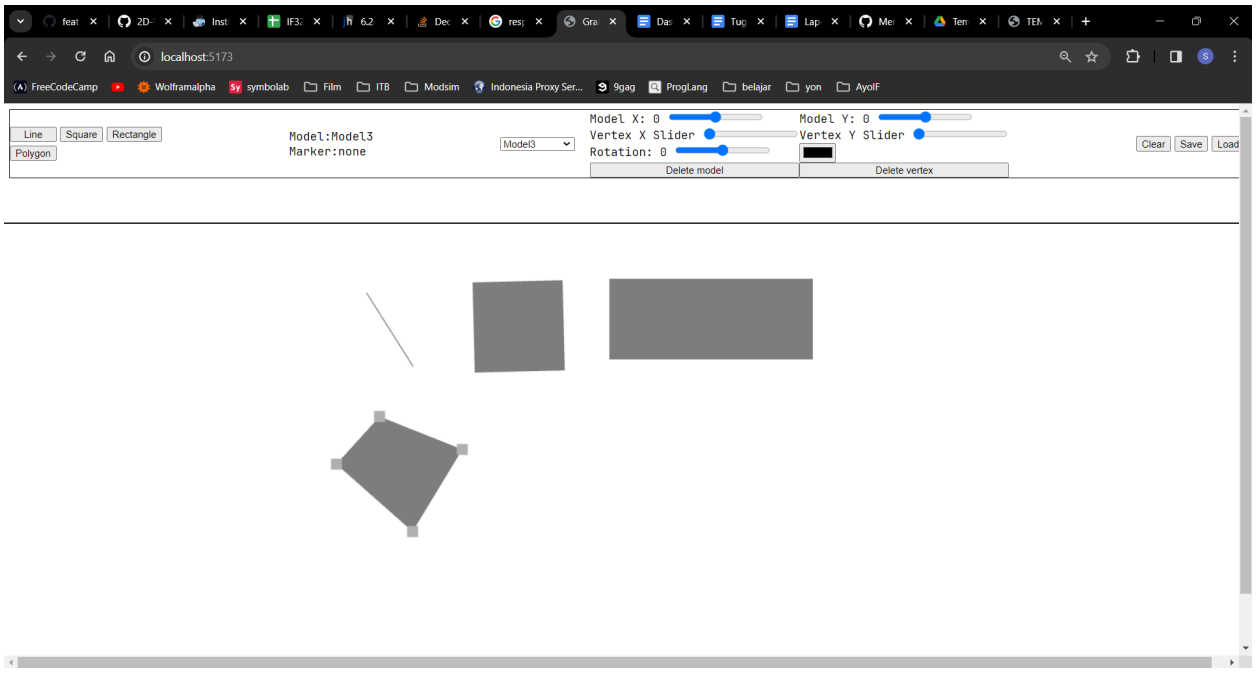


BAB IV
MANUAL

4.1 Menambahkan Model

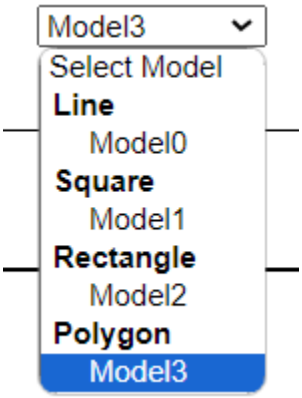
1. Pilih salah satu model dengan mengklik salah satu button

2. Klik pada canvas sebanyak vertex yang diperlukan
   Note: Line, Square, dan Rectangle berupa 2 vertex dan Polygon >= 4 vertex

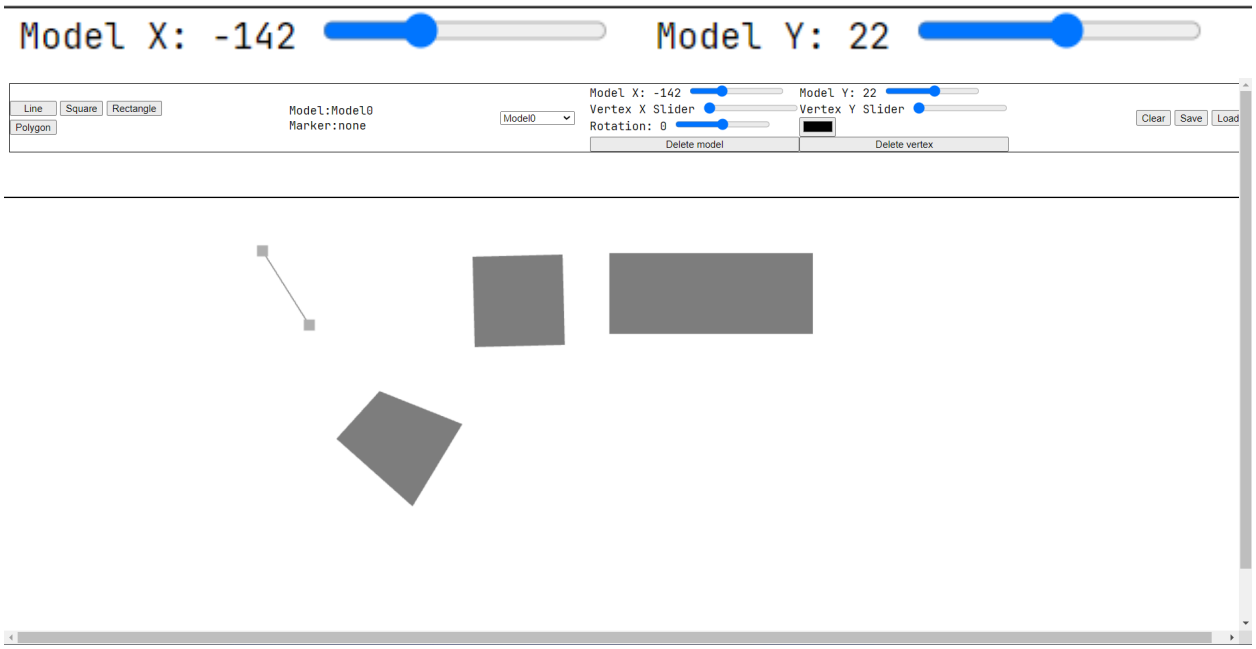


4.2 Mentranslasikan Model

1. Pilih salah satu model pada dropdown

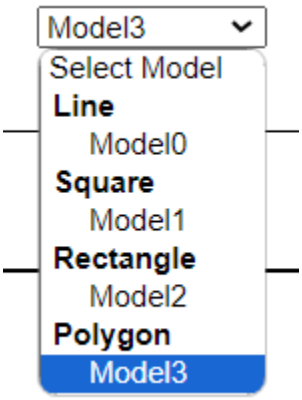


2. Gerakan pada salah satu axis menggunakan slider Model [Axis]

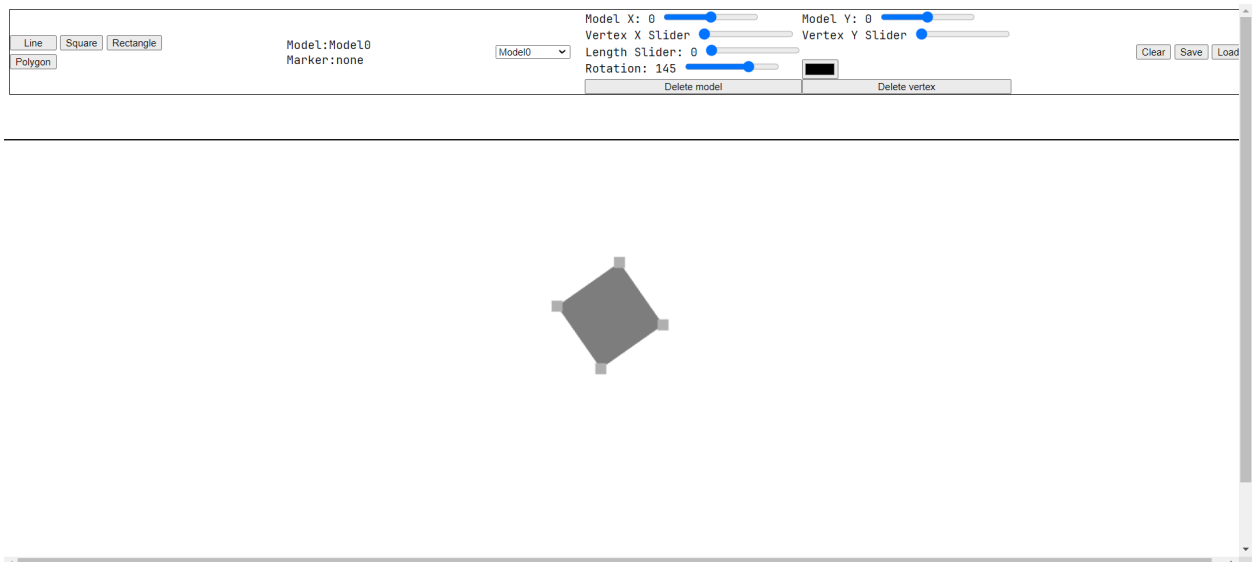


4.3 Merotasikan Model

1. Pilih salah satu model pada dropdown

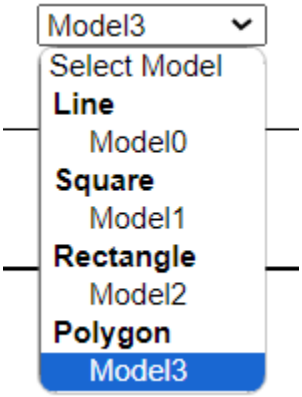


2. Gerakan pada salah satu axis menggunakan slider Rotation



4.4 Mengubah Warna Model

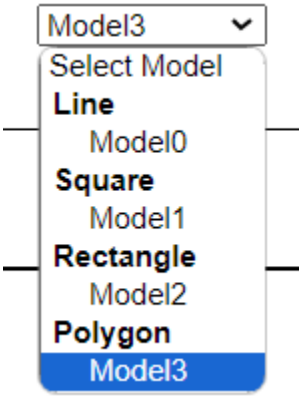
1. Pilih salah satu model pada dropdown



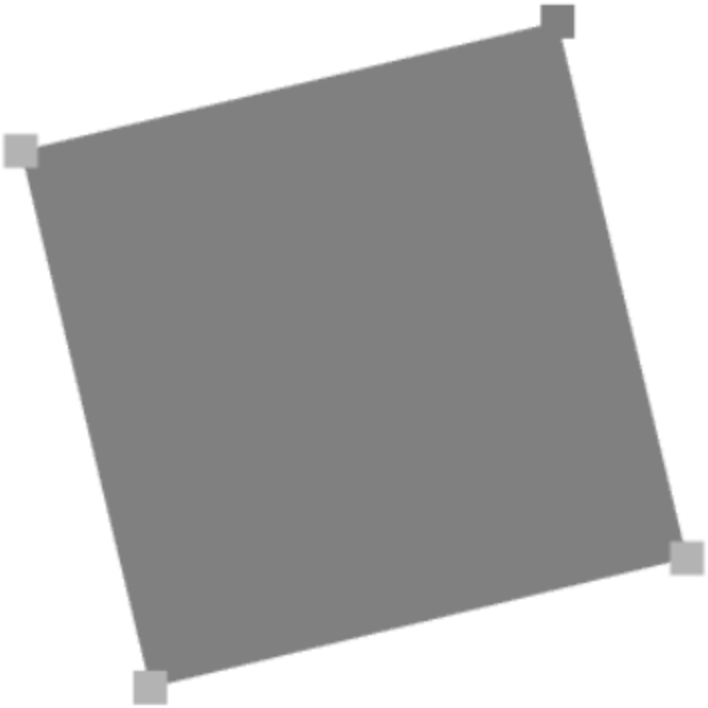
2. Pilih warna pada color input

4.5 Mengubah Warna untuk Satu Titik Sudut

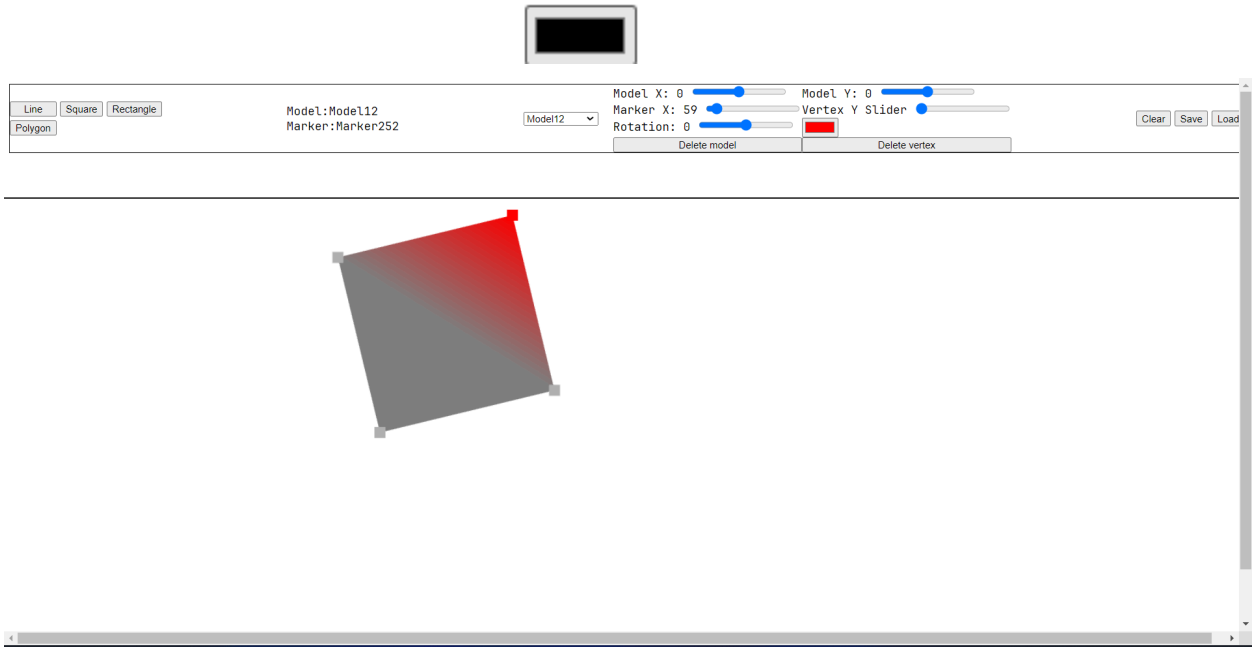
1. Pilih salah satu model pada dropdown



2. Pilih salah satu marker pada objek yang berupa kotak kecil pada sudut

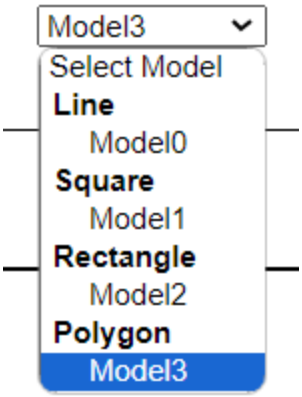


3. Pilih warna pada color input

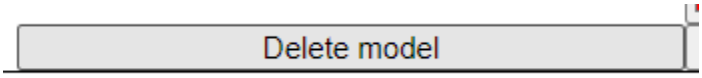


4.6 Menghapus Model

1. Pilih salah satu model pada dropdown

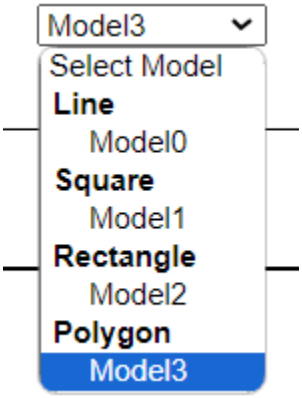


2. Klik delete model



4.7 Mengubah Panjang dan Lebar

1. Pilih salah satu model pada dropdown



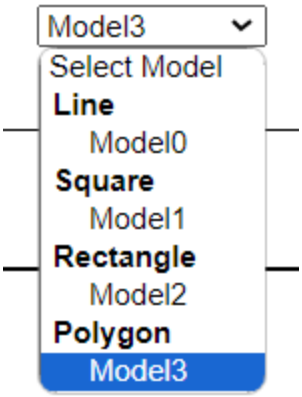
2. Gerakkan slider berikut
   *Note : pada model line dan square hanya terdapat length slider

Model sebelum

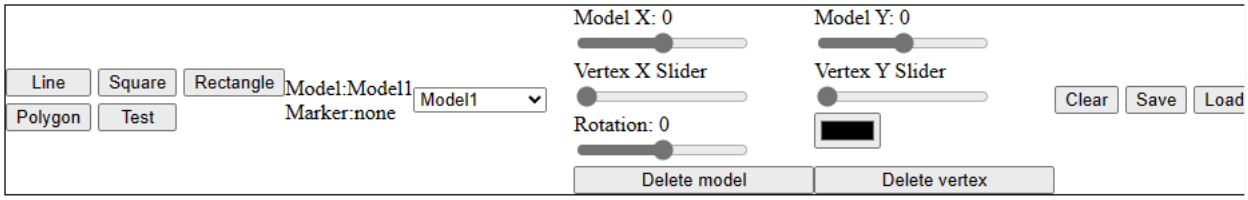
Model sesudah

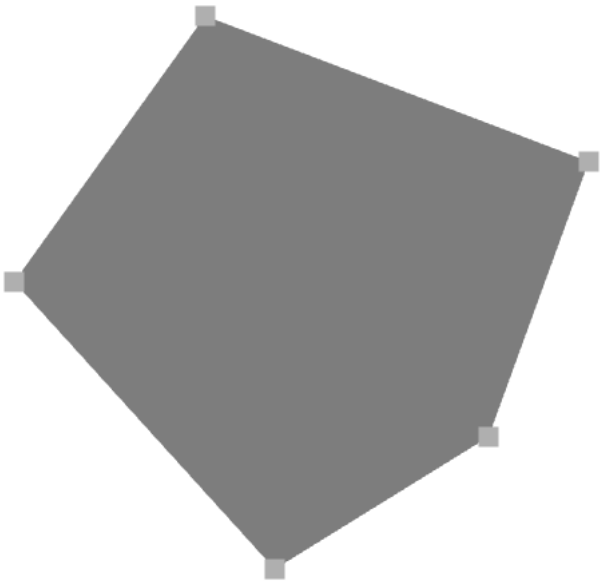
4.8 Penambahan dan Pengurangan Titik Sudut pada Polygon

1. Pilih salah satu model pada dropdown

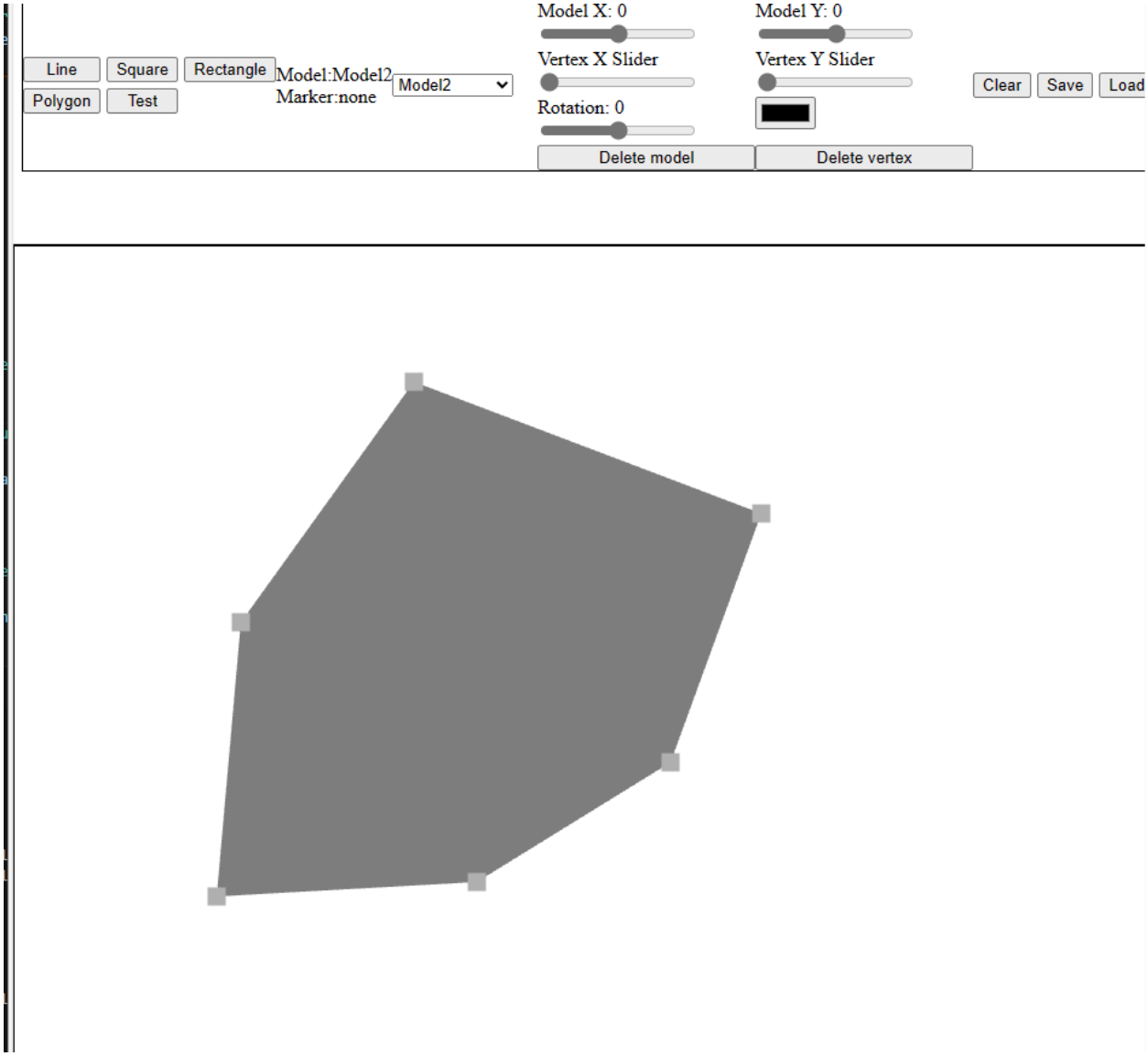


2. Klik di manapun pada layar selain pada tombol atau vertex untuk menambahkan vertex
   Model Sebelum

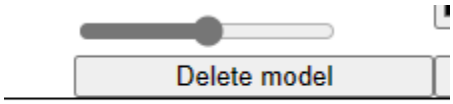


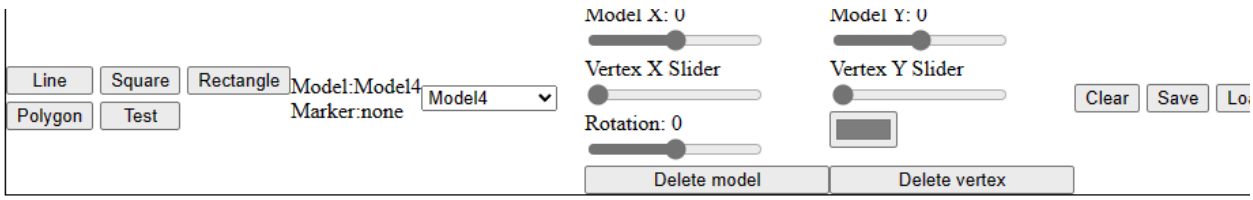


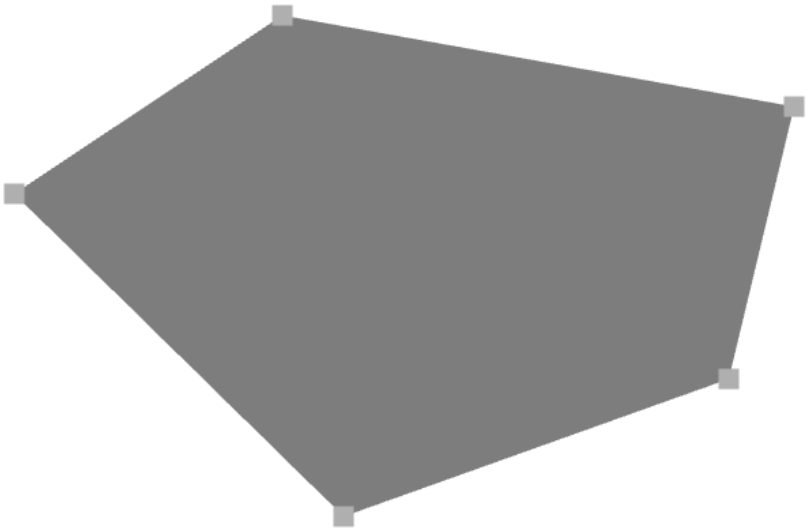
   Model sesudah

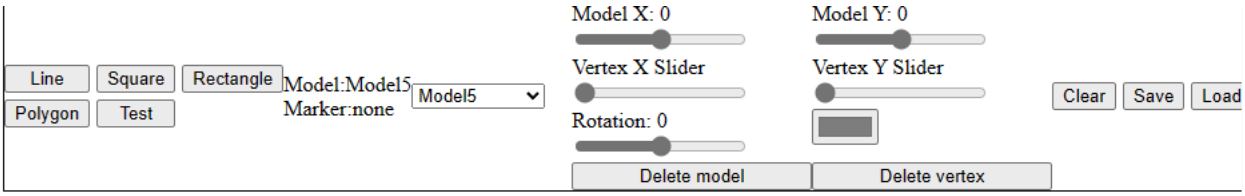


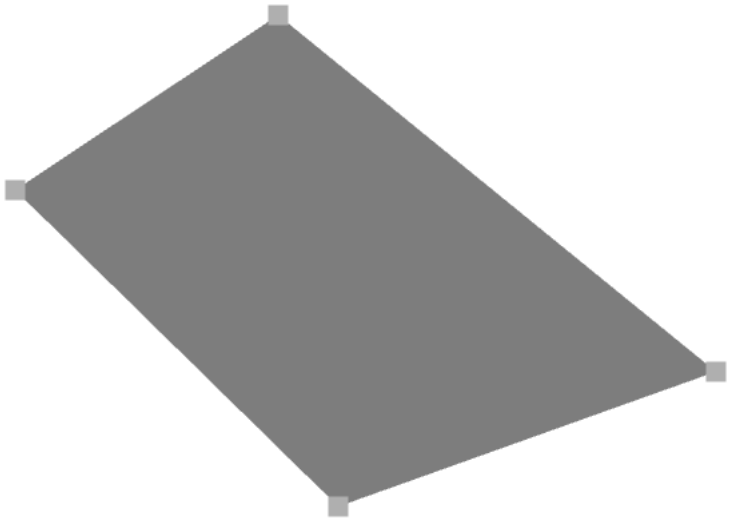
3. Klik delete model untuk mengurangi vertex, vertex harus berjumlah lebih dari 4




Model X: 0
Model Y: 0
Line
Square
Rectangle
Model:Model4
Marker:none
Model4
Vertex X Slider
Vertex Y Slider
Clear
Save
Polygon
Test
Rotation: 0
Delete model
Delete vertex

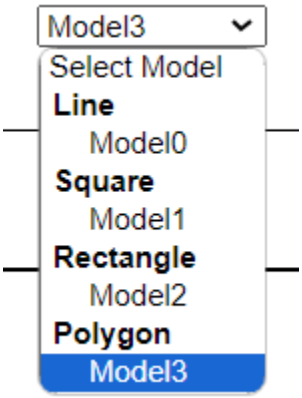




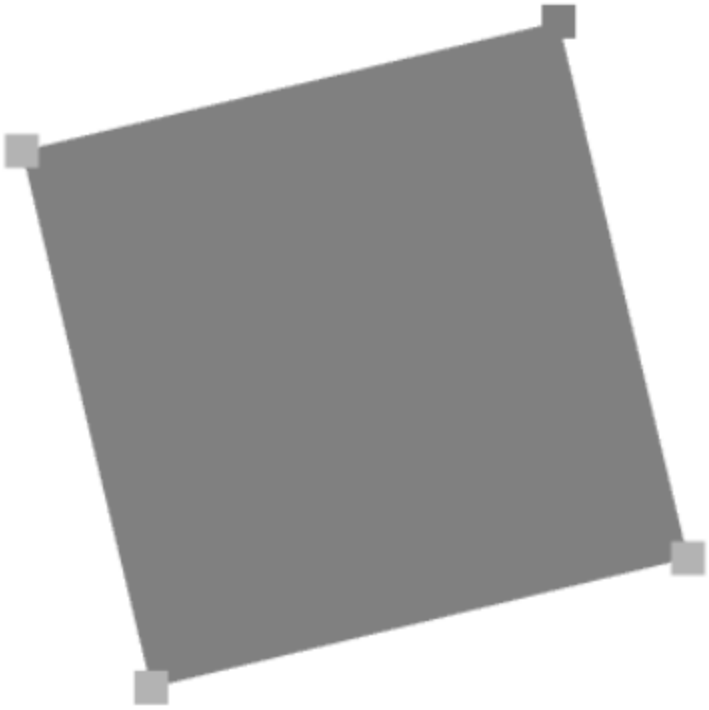


4.9 Menggerakkan salah satu titik sudut dengan slider

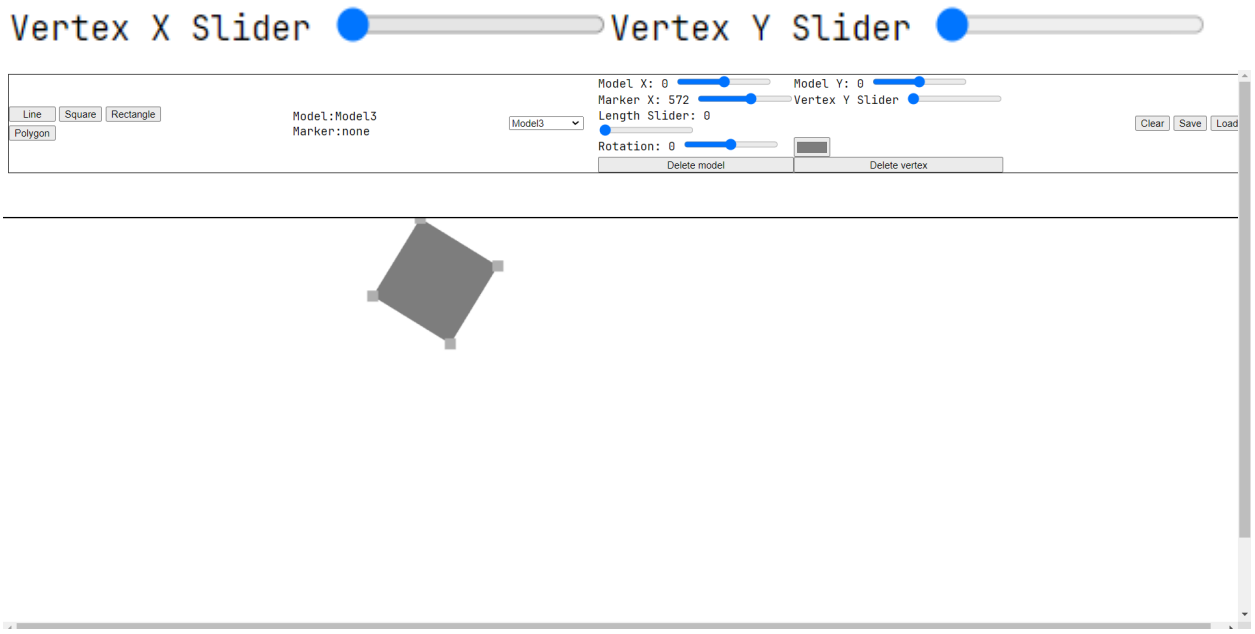
1. Pilih salah satu model pada dropdown



2. Pilih salah satu marker pada objek yang berupa kotak kecil pada sudut

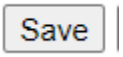


3. Gerakkan slider berikut



4.10 Menyimpan Object

1. Klik pada button save



4.11 Men-*load* Object

1. Pilih file dengan meng-klik button load