



1. Inisialisasi Canvas dan Variabel

js

Copy Edit

```
const canvas = document.getElementById('canvas'),
      ctx = canvas.getContext('2d'),
      W = canvas.width, H = canvas.height;
```

- Mengambil elemen `<canvas>` dari HTML.
- Mendapatkan context 2D untuk menggambar (`ctx`).
- Menyimpan lebar dan tinggi canvas dalam variabel `W` dan `H`.

js

Copy Edit

```
let mode = 1, rotX = 0, rotY = 0;
```

- `mode = 1` berarti mode awal adalah wireframe.
- `rotX`, `rotY` adalah sudut rotasi dalam radian untuk sumbu X dan Y.



2. Definisi Geometri Objek

js

Copy Edit

```
const nodes = [...]
```

- Array 8 titik 3D dari balok (panjang \neq lebar \neq tinggi).
- Setiap titik adalah koordinat `[x, y, z]`.

js

Copy Edit

```
const edges = [...]
```

- Menentukan garis-garis penghubung antar titik (`nodes`).
- Tiap elemen adalah pasangan indeks dari `nodes`.

js



Copy Edit

```
const faces = [...]
```

- Setiap elemen adalah 1 permukaan (face) dari balok.
- Digunakan untuk menghitung permukaan yang terlihat pada mode 2.

3. Fungsi Rotasi Titik 3D

js



 Copy  Edit

```
function rotate([x, y, z]) { ... }
```

- Melakukan transformasi rotasi terhadap titik `(x, y, z)` terhadap sumbu X dan Y:
 - Rotasi X: memutar nilai `y` dan `z`
 - Rotasi Y: memutar nilai `x` dan `z` (yang sudah dirotasi sebelumnya)
- Output adalah titik 3D yang sudah dirotasi: `[x', y', z']`

4. Proyeksi Titik ke Layar 2D

js



 Copy  Edit

```
function project([x, y]) { ... }
```

- Melakukan **proyeksi ortografis** (tanpa perspektif) dari 3D ke 2D.
- Mengatur titik ke tengah canvas (`+ W/2`, `+ H/2`) dan membalik arah `y` (karena canvas `y` positif ke bawah).

5. Menghitung Posisi Z Tengah Sebuah Face

js



 Copy  Edit

```
function faceCentroidZ(face) { ... }
```

- Mengambil semua titik dari `face`, dirotasi, dan ambil nilai `z` masing-masing.
- Mengembalikan nilai **rata-rata Z** dari face tersebut.
- Face dianggap **menghadap ke depan (terlihat)** jika $Z < 0$.

6. Fungsi `draw()` — Proses Gambar

js



 Copy  Edit

```
function draw() { ... }
```

- Menghapus layar (`clearRect`) lalu menggambar ulang:
 - Menghitung hasil rotasi & proyeksi semua titik (`proj`)
 - Menentukan face mana yang terlihat (`visFaces`)
- Untuk setiap edge:
 - Jika `mode === 2` (hidden surface removal):
 - Gambar hanya edge yang bagian dari face terlihat
 - Gambar garis dari `p` ke `q` dengan `strokeStyle` hijau.

7. Fungsi `animate()` — Loop Gambar

js



 Copy  Edit

```
function animate() { draw(); requestAnimationFrame(animate); }
```

- Memanggil `draw()` terus-menerus per frame menggunakan `requestAnimationFrame` .
- Memberikan efek animasi interaktif.

8. Keyboard Event Listener

js

 Copy  Edit

```
window.addEventListener('keydown', e => { ... });
```

- Mendengarkan tombol yang ditekan:
 - A / D → rotasi sumbu Y
 - W / S → rotasi sumbu X
- `Math.PI / 90` berarti setiap kali tombol ditekan, objek diputar sebesar 2 derajat.

📄 9. Event Tombol UI (Wireframe / Hidden Surface)

js

📄 Copy ✎ Edit

```
document.getElementById('part1')... → mode = 1  
document.getElementById('part2')... → mode = 2
```

- Jika tombol diklik:
 - `mode = 1` : tampilkan semua edge (wireframe).
 - `mode = 2` : tampilkan hanya edge yang bagian dari face yang menghadap ke kamera.

🌟 10. Fungsi `updateUI()` — Tampilan Aktif

js

📄 Copy ✎ Edit

```
function updateUI(active) {  
  document.querySelectorAll('#controls button').forEach(btn => {  
    btn.classList.toggle('active', btn.id === 'part' + active);  
  });  
}
```

- Menambahkan kelas `active` ke tombol mode yang aktif, dan menghapusnya dari tombol lain.
- Bertanggung jawab untuk **feedback visual** tombol saat mode dipilih.