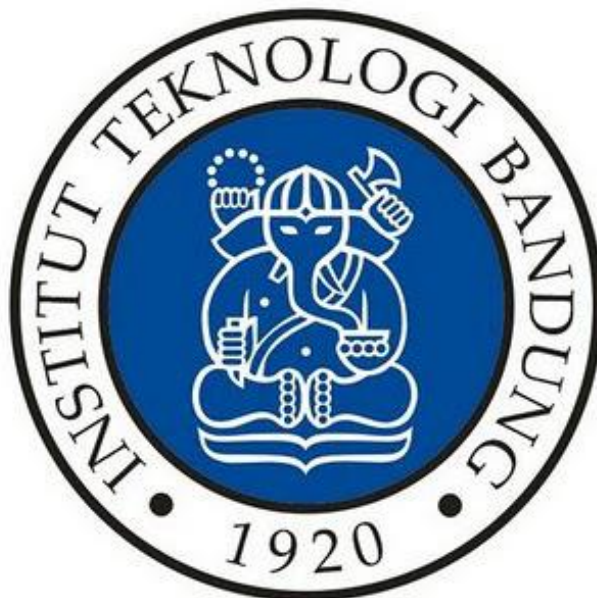


# LAPORAN

**Tugas Besar 2: 3D WebGL Hollow Object  
IF3260 Grafika Komputer**

**DISUSUN OLEH**

13520124	Owen Christian Wijaya
13520125	Ikmal Alfaozi
13520165	Ghazian Tsabit Alkamil



**PROGRAM STUDI TEKNIK INFORMATIKA  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG  
2022**

# DAFTAR ISI

DAFTAR ISI	2
BAB I: DESKRIPSI	3
BAB II: HASIL PROGRAM	4
BAB III: MANUAL FUNGSIONALITAS PROGRAM	7
<a href="#">Pemilihan Objek</a>	<a href="#">7</a>
<a href="#">Transformasi</a>	<a href="#">8</a>
<a href="#">Kontrol Kamera</a>	<a href="#">10</a>
<a href="#">Lain-lain</a>	<a href="#">11</a>

# BAB I: DESKRIPSI

Laporan ini dituliskan sebagai dokumentasi dan *deliverable* dari Tugas Besar 2 mata kuliah IF3260 Grafika Komputer, program studi Teknik Informatika, Institut Teknologi Bandung, tahun ajaran 2022/2023.

Tugas besar 2 bersangkutan dengan penggunaan WebGL untuk mengembangkan sebuah web aplikasi yang dapat membuat dan memanipulasi 3D *hollow object*. WebGL sendiri adalah sebuah teknologi grafis yang memungkinkan proses rendering tiga dimensi di browser web tanpa memerlukan plugin tambahan. WebGL menggunakan API JavaScript untuk mengakses perangkat grafis yang terdapat pada komputer dan menghasilkan grafis 3D yang interaktif secara *real-time* di dalam browser. WebGL menggunakan bahasa pemrograman OpenGL ES, yang biasa digunakan dalam pengembangan aplikasi grafis.

WebGL yang digunakan pada implementasi kali ini merupakan WebGL murni, tanpa library/framework tambahan. Pengembangan lanjutan melibatkan pembuatan kakas pengolahan dan manipulasi matriks untuk melakukan transformasi *vertex*.

Pada web aplikasi 3D *hollow object* ini diimplementasikan beberapa model objek, dengan setiap object memiliki interaksi yang sama yaitu mengubah jenis proyeksi, melakukan rotasi, translasi, dan scaling. Selain itu, pada web aplikasi ini dimungkinkan untuk mengubah jarak kamera, me-reset objek ke mode default, dan juga teknik shading pada tiap objek.

Model objek yang diimplementasikan pada web aplikasi yang dibuat adalah sebagai berikut:

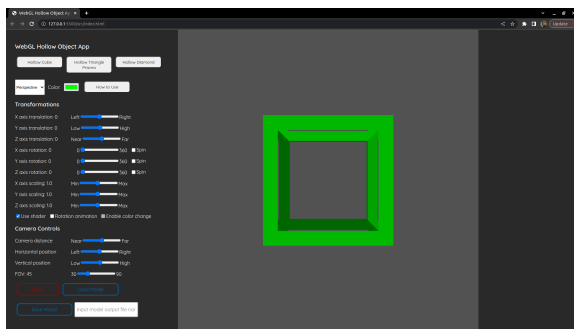
1. *Hollow Cube*
2. *Hollow Triangle Prism*
3. *Hollow Diamond*

## BAB II: HASIL PROGRAM

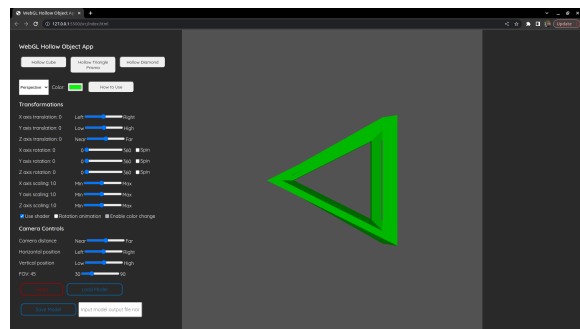
Hasil program yang dibuat adalah sebuah *website* dimana pengguna dapat melakukan manipulasi dan transformasi terhadap model 3D (translasi, rotasi, dan *scaling*). Pengguna berinteraksi menggunakan *slider* untuk menentukan perubahan yang terjadi pada objek.

Pada tugas besar ini, model yang dibuat merupakan model tiga-dimensi *hollow*. Hal ini berarti setiap *vertex* mempunyai informasi nilai pada sumbu Z, dan harus disusun sedemikian rupa untuk menunjukkan bagian kosong dari model. Penggambaran dilakukan menggunakan fungsi `gl.DrawElements`, yang menerima sebuah parameter `indices`. Parameter `indices` menandakan urutan penggambaran *vertex* sedemikian rupa sehingga membentuk sebuah model *hollow* dari *vertex* yang diberikan. Setiap *vertex* akan diwakilkan oleh sebuah nilai index sesuai dengan urutan deklarasinya. Proses penggambaran mengikuti *pipeline* yang serupa dengan tugas besar 1, yaitu kompilasi *shader program*, pembuatan *buffer*, asosiasi *buffer* ke program utama (terdapat *indices buffer* dan *normal buffer*), dan penggambaran.

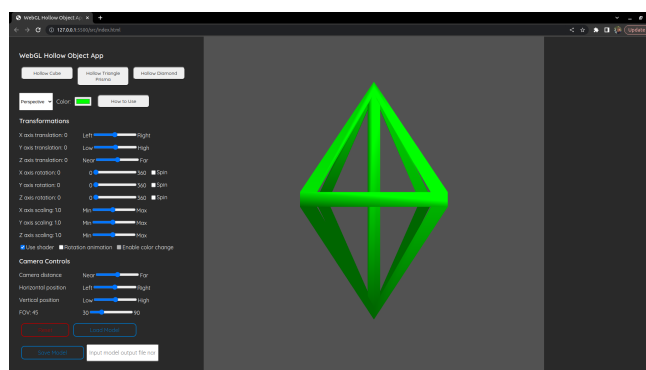
Terdapat tiga buah model *hollow* yang diimplementasikan, yaitu:



Model *hollow cube* (kubus kosong)



Model *hollow triangle prisma* (prisma segitiga kosong)



Model *hollow diamond* (berlian kosong)

Transformasi *vertex* yang dilakukan berbeda dengan transformasi konvensional yang dilakukan pada tugas besar sebelumnya (langsung melakukan manipulasi nilai *vertex*), namun menggunakan operasi matriks pada *shader program*. Pada *shader program*, terdapat dua buah variabel tambahan untuk menentukan nilai `gl.Position`, yaitu matriks proyeksi (*projection matrix*) dan matriks *model view*

(*model view matrix*). *Projection matrix* digunakan untuk melakukan transformasi proyeksi terhadap model yang akan ditampilkan, dan matriks *model view* menyimpan hasil transformasi dalam bentuk matriks. Melalui persamaan `projectionMatrix * modelViewMatrix * vec4(aVertexPosition, 1.0)`, nilai `glPosition` akan merepresentasikan sebuah vertex yang sudah melalui proses proyeksi dan transformasi.

Hasil dari *model view matrix* akan diperoleh melalui masukan pengguna pada program (akan dijelaskan lebih lanjut pada Bab III). Pengguna memasukkan parameter transformasi dalam bentuk nilai translasi, rotasi, dan scaling pada sumbu X, Y, atau Z. Kakas matriks akan membentuk sebuah matriks yang diperlukan untuk merepresentasikan transformasi terhadap vertex. Transformasi akan dilakukan dengan mengkalikan *model view matrix* dengan matriks transformasi dengan urutan:

```
modelViewMatrix = scaleMatrix * rotationXMatrix * rotationYMatrix *  
rotationZMatrix * translateMatrix * lookAtMatrix
```

Dengan:

- `scaleMatrix` merepresentasikan matriks untuk melakukan operasi *scaling*
- `rotationXMatrix` merepresentasikan matriks untuk melakukan operasi rotasi pada sumbu X
- `rotationYMatrix` merepresentasikan matriks untuk melakukan operasi rotasi pada sumbu Y
- `rotationZMatrix` merepresentasikan matriks untuk melakukan operasi rotasi pada sumbu Z
- `translateMatrix` merepresentasikan matriks untuk melakukan operasi translasi
- `lookAtMatrix` merepresentasikan matriks `lookAt`, yang digunakan untuk memposisikan kamera

*Projection matrix* akan diinisialisasi tergantung tipe proyeksi yang dipilih, sedangkan *model view matrix* akan diinisialisasi dengan matriks `lookAt`. Inisialisasi matriks `lookAt` juga dipengaruhi oleh tipe proyeksi yang dipilih, dikarenakan adanya perubahan sistem koordinat pada proyeksi perspektif dan paralel. Sumbu Z pada NDC berlawanan arah dengan sumbu Z pada proyeksi perspektif. Oleh karena itu, untuk menyesuaikan, parameter `at` pada matriks `lookAt` perspektif akan diletakkan pada koordinat  $[0, 0, -1]$ , sedangkan pada matriks `lookAt` perspektif akan diletakkan pada koordinat  $[0, 0, 1]$ . Hal ini dilakukan supaya translasi dan proyeksi sumbu X dapat dilakukan secara akurat.

Perbedaan lain terletak pada sistem *shading* yang diimplementasikan via *shader program*. Setiap model mempunyai kumpulan *normal vertex* yang mewakili ke arah mana sebuah cahaya seharusnya dipancarkan. Untuk tugas besar kali ini, *normal vertex* dari setiap *vertex* akan diarahkan menuju bidang yang menjadi acuan hadapan sebuah *vertex*. Misalnya, apabila sebuah *vertex* dibuat untuk menghadap ke kiri, maka *normal vertex* dari *vertex* tersebut adalah  $[-1.0, 0.0, 0.0]$ .

*Normal vertex* akan diproses oleh *shader program* dengan mempertimbangkan *ambient light*, *light color*, dan *directional vector*. *Ambient light* adalah intensitas dari cahaya yang dipancarkan, *light color* mewakili warna cahaya (untuk tugas besar ini warna cahaya yang digunakan adalah putih), dan *directional vector* menandakan dari mana cahaya akan dipancarkan. Hasil yang diperoleh adalah

efek bayangan pada model yang ditampilkan, yang dapat berubah secara dinamis mengikuti transformasi. Efek bayangan ini sendiri muncul karena hasil perkalian dari *lighting vector* hasil perhitungan *vertex shader* untuk menentukan variabel `gl_FragColor` pada *fragment shader*. Fitur *shading* ini dapat dinyalakan atau dimatikan. Apabila dimatikan, maka *ambient light* akan disetel menjadi [1.0, 1.0, 1.0] yang menandakan intensitas penuh, sehingga warna dari keseluruhan model akan merata.

Terdapat dua fitur lanjutan yang diimplementasikan, yaitu **pembuatan animasi** dan **fitur load-save lanjutan**

### 1. Pembuatan animasi

Animasi yang dibuat adalah animasi rotasi secara otomatis seiring berjalannya waktu. Pengguna dapat mengaktifkan rotasi secara otomatis pada ketiga buah sumbu secara otomatis. Pengguna juga dapat mengaktifkan rotasi pada sumbu X, Y, dan Z secara individual. Pada saat melakukan animasi rotasi, rotasi manual tidak dapat dilakukan, sehingga animasi rotasi ini dapat dianggap sebagai *preview* dari objek. Selain itu, pengguna juga dapat mengaktifkan animasi perubahan warna saat melakukan rotasi. *Update* warna dilakukan terhadap setiap *vertex* seiring berjalannya waktu.

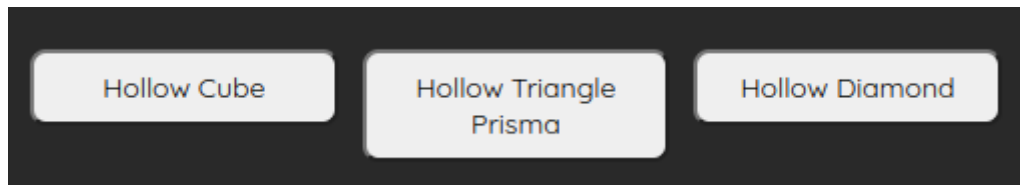
### 2. Fitur load-save

Pengguna dapat mengaplikasikan transformasi, menyimpan model, dan melakukan *load* ulang terhadap objek. Pada saat menyimpan, nilai transformasi yang sudah dilakukan akan disimpan dalam sebuah struktur bernama `config` dalam *save model*. Parameter ini akan dijadikan basis transformasi apabila model di-*load*. Hal ini dikarenakan apabila melakukan transformasi *scaling* dan rotasi pada model sebelum di-*save*, model yang di-*load* akan diproses menggunakan sumbu yang berbeda dengan sumbu saat manipulasi, sehingga transformasi *scaling* akan menjadi transformasi *shearing*. Untuk mencegah hal ini, parameter transformasi akan dijadikan basis perhitungan pada saat *loading*, sehingga *model view matrix* akan disesuaikan dengan parameter transformasi. Transformasi setelah *loading* masih dapat diobservasi, dan konfigurasi hanya akan di-*update* ketika akan melakukan *save*.

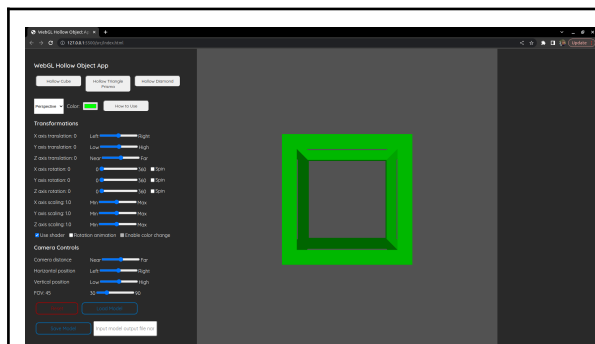
# BAB III: MANUAL FUNGSIONALITAS PROGRAM

## Pemilihan Objek

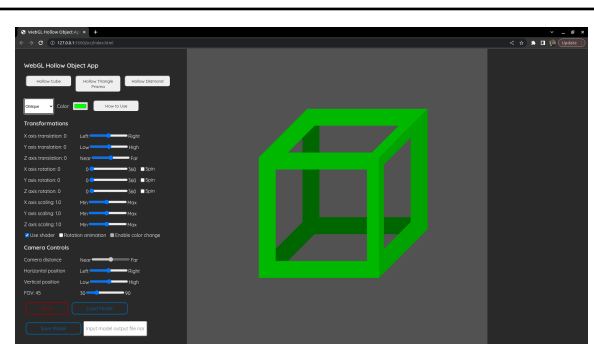
1. Pengguna dapat memilih salah satu dari tiga objek:
  - Kubus Kosong (*Hollow Cube*)
  - Prisma Segitiga Kosong (*Hollow Triangle Prisma*)
  - Berlian Kosong (*Hollow Diamond*) (**Secara default akan me-load objek ini**)



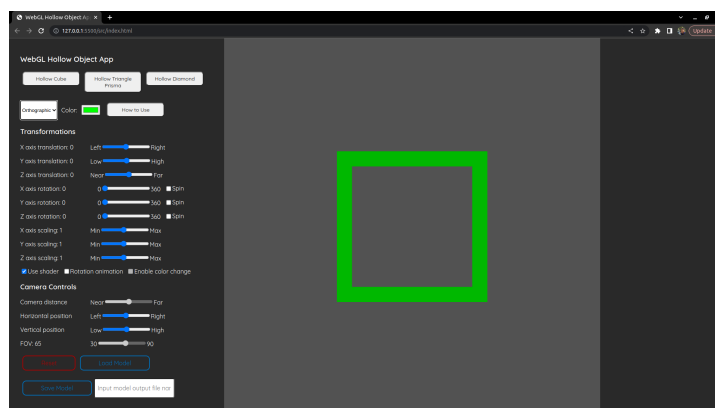
2. Pengguna dapat memilih jenis proyeksi *perspective*, *oblique*, dan *orthographic*.
  - Khusus proyeksi *perspective* dan *oblique*, pengguna dapat mengubah sudut tampilan dengan *FOV slider*
3. Pengguna dapat mengubah warna objek yang ditampilkan menggunakan *color picker*.



Proyeksi *perspective*



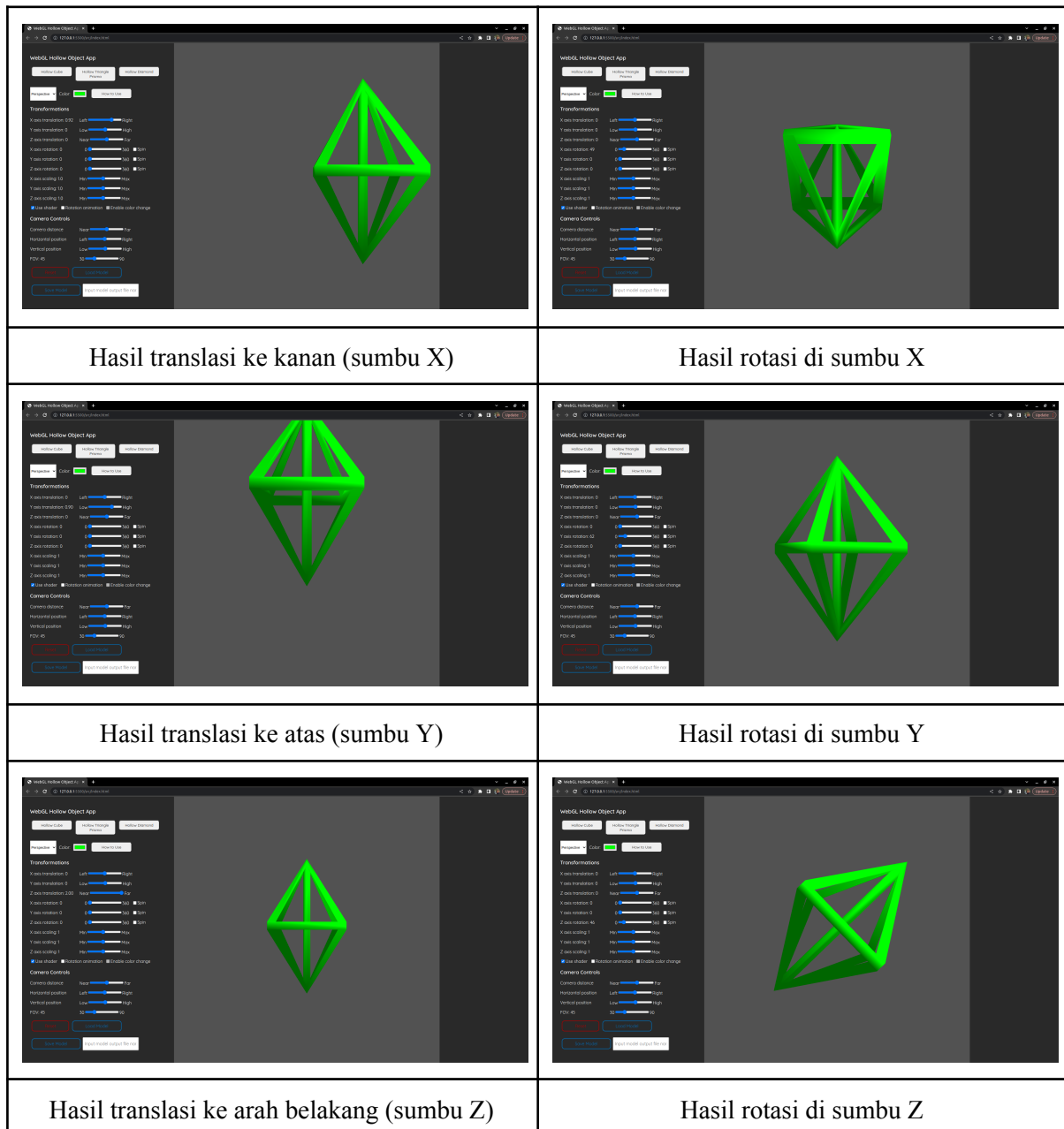
Proyeksi *oblique*



Proyeksi *orthographic*

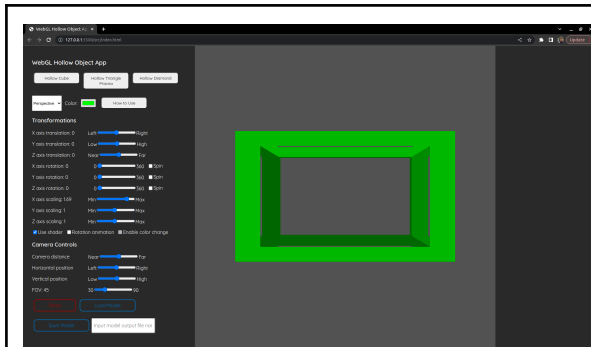
## Transformasi

1. Pengguna dapat melakukan translasi objek pada sumbu X, Y, dan Z menggunakan *translation slider* yang tersedia.
2. Pengguna dapat melakukan rotasi objek pada sumbu X, Y, dan Z menggunakan *rotation slider* yang tersedia.
  - o Pengguna dapat mengaktifkan rotasi otomatis per sumbu menggunakan *checkbox Spin* di kanan *slider*

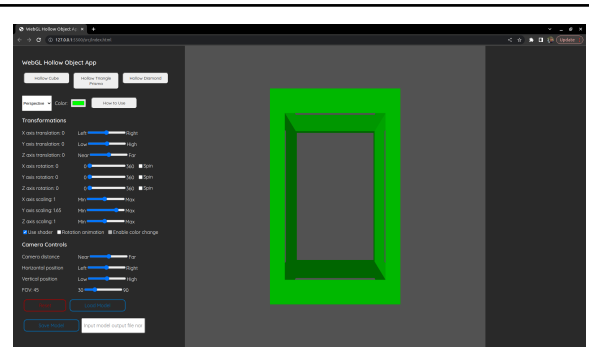


3. Pengguna dapat melakukan scaling objek pada sumbu X, Y, dan Z menggunakan *scaling slider* yang tersedia

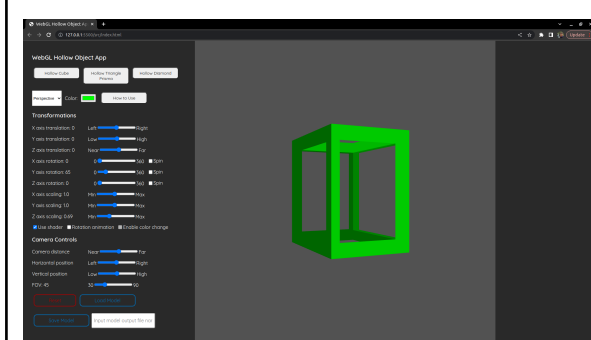




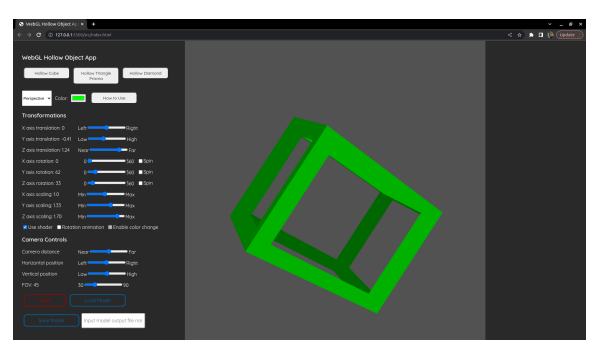
Hasil *scaling* sepanjang sumbu X



Hasil *scaling* sepanjang sumbu Y



Hasil *scaling* sepanjang sumbu Z (setelah dirotasikan)

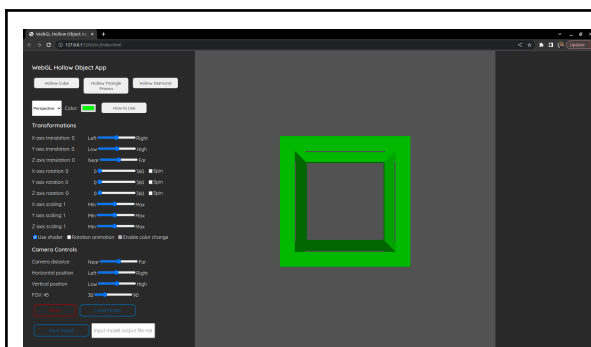


Hasil beberapa transformasi (translasi, rotasi, *scaling*)

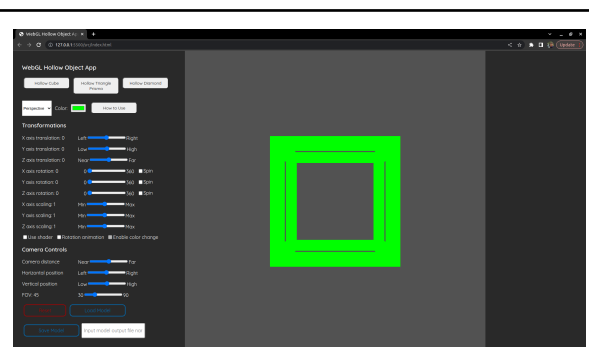
#### 4. Pengguna dapat mengaktifkan *shader* menggunakan *checkbox* "Use **shader**?"

- Menyalakan *shader* akan memberikan efek bayangan pada objek yang ditampilkan.
- Menonaktifkan *shader* akan membuat objek ditampilkan dengan satu warna solid.

Arah datang cahaya berasal dari bagian kanan atas *canvas*. Apabila objek dirotasikan, maka arah cahaya akan tetap konstan berasal dari titik yang sama. Apabila kamera dipindahkan, posisi cahaya akan disesuaikan supaya tampak tetap konstan. Hal ini disebabkan oleh koordinat kamera yang tidak berubah setelah melakukan transformasi pada matriks **lookAt**, sehingga tampak seakan-akan hanya model yang bergerak, atau cahaya bergerak bersamaan dengan kamera.

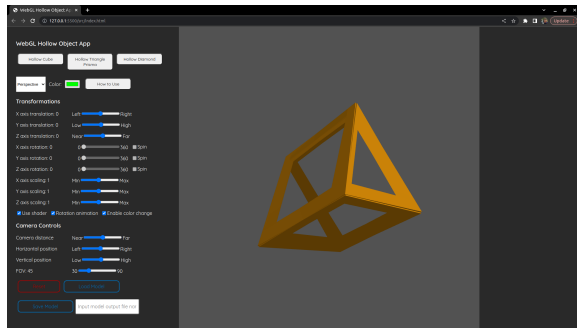
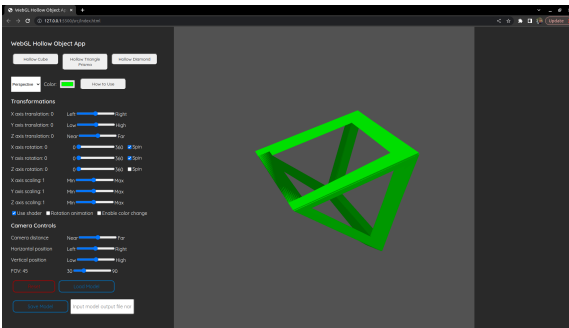


Tampilan apabila *shader* diaktifkan



Tampilan apabila *shader* dinonaktifkan

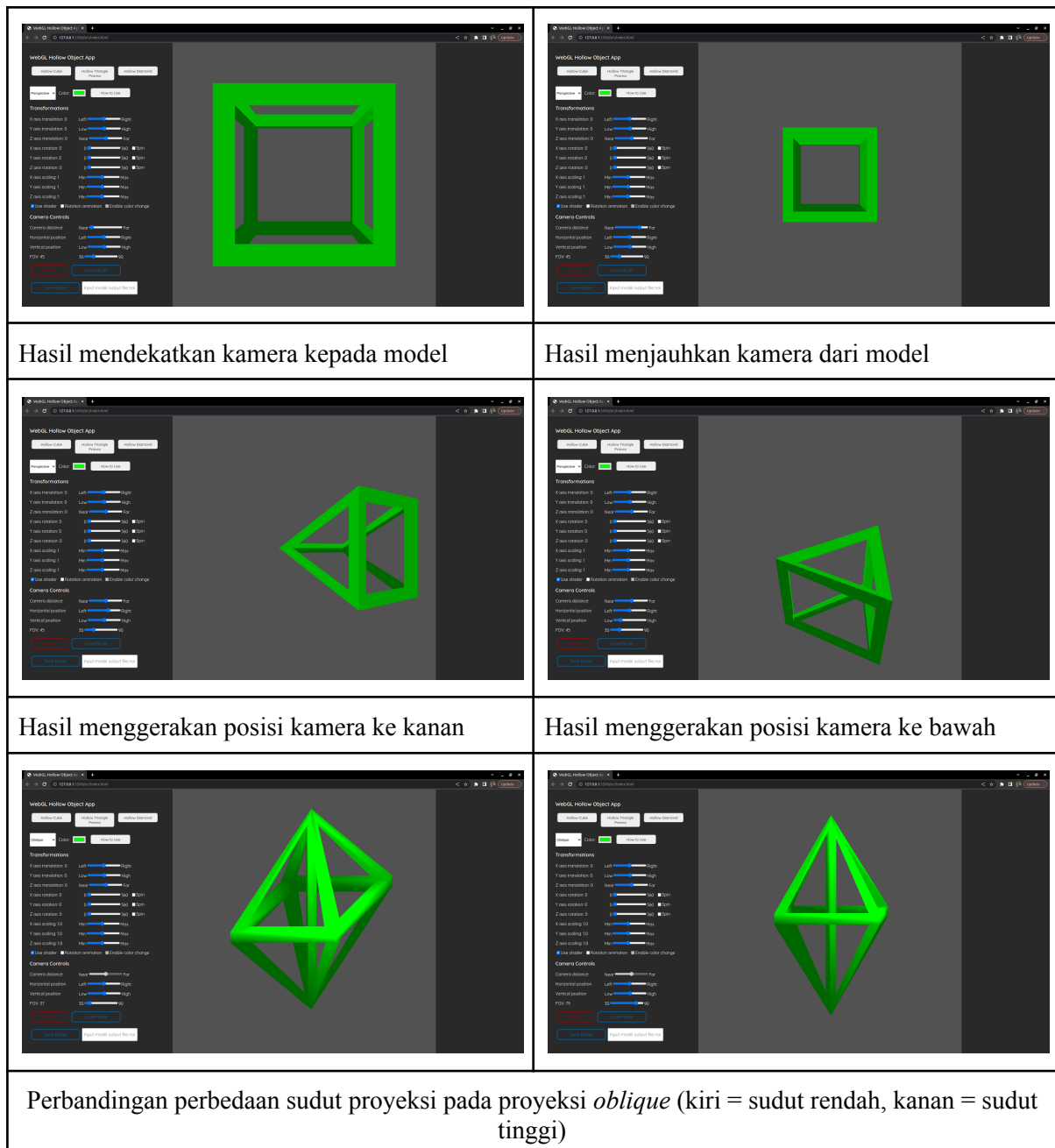
5. Pengguna dapat mengaktifkan animasi rotasi menggunakan **checkbox "Rotation animation"**
  - Menyalakan *checkbox* akan membuat objek berputar secara otomatis secara random dan berubah warna seiring berjalannya waktu.
  - Menonaktifkan *checkbox* akan membuat objek berhenti berputar dan kembali ke *state* sebelum berputar.
  - Pada saat animasi aktif, *rotation slider* dan *color picker* **tidak dapat digunakan**, namun dapat digunakan setelah *checkbox* dinonaktifkan
6. Pengguna dapat mengaktifkan perubahan warna saat berputar dengan menggunakan **checkbox "Enable color change"**
  - Dapat digunakan juga pada *checkbox Spin* pada rotasi X, Y, dan Z
  - *Checkbox* akan aktif apabila terdapat proses rotasi otomatis yang aktif

	
<p>Tampilan menggunakan animasi rotasi + <i>color change</i>. <i>Slider</i> rotasi akan di-<i>disable</i></p>	<p>Tampilan menggunakan animasi rotasi putaran per sumbu (sumbu X + Y).</p>

## Kontrol Kamera

1. Pengguna dapat mengatur jarak kamera dari objek menggunakan *slider* **"Camera distance"**
2. Pengguna dapat menggerakkan kamera mengelilingi objek secara horizontal menggunakan *slider* **"Horizontal position"**
  - Pergerakan dilakukan sejauh **maksimal 360 derajat** dari posisi awal kamera
3. Pengguna dapat menggerakkan kamera mengelilingi objek secara vertical menggunakan *slider* **"Vertical position"**
  - Pergerakan dilakukan sejauh **maksimal 90 derajat** dari posisi awal kamera
4. Khusus proyeksi *perspective* dan *oblique*, pengguna dapat mengubah sudut tampilan dengan *FOV slider*

Untuk proyeksi *orthogonal*, tidak perlu dilakukan pengaturan FOV karena tampilan tegak lurus. Untuk proyeksi *oblique*, FOV mengatur sudut antara kamera dengan bidang proyeksi. Pada FOV = 90 derajat, tampilan *oblique* akan tampak seperti *orthogonal*



## Lain-lain

1. Pengguna dapat me-reset parameter transformasi menggunakan tombol **Reset**
2. Pengguna dapat melakukan *loading* objek yang sudah disimpan selanjutnya dengan tombol **Load Model**
3. Pengguna dapat melakukan *save* objek yang telah dibuat dengan mengisi nama *file* di sebelah tombol **Save Model**, lalu menekan tombol **Save Model**
  - Transformasi yang dilakukan akan disimpan, dan akan diaplikasikan **hanya pada saat loading** sebagai state awal.
  - Hal ini dikarenakan apabila mengaplikasikan transformasi pada koordinat *vertex*, hasil *load* akan ditampilkan menggunakan sumbu yang berbeda. Hal ini berpotensi menyebabkan fitur *scaling* menjadi *shearing*, karena sumbu yang digunakan sebagai referensi pada saat pembuatan *file* akan berbeda dengan sumbu awal.

- Konfigurasi ini **hanya akan disimpan** pada saat melakukan penyimpanan, sehingga setelah proses *loading*, konfigurasi dapat di-*reset* sesuai dengan kondisi awal dan proses beberapa transformasi masih dapat diobservasi. Saat model dimanipulasi, konfigurasi tidak akan diubah.