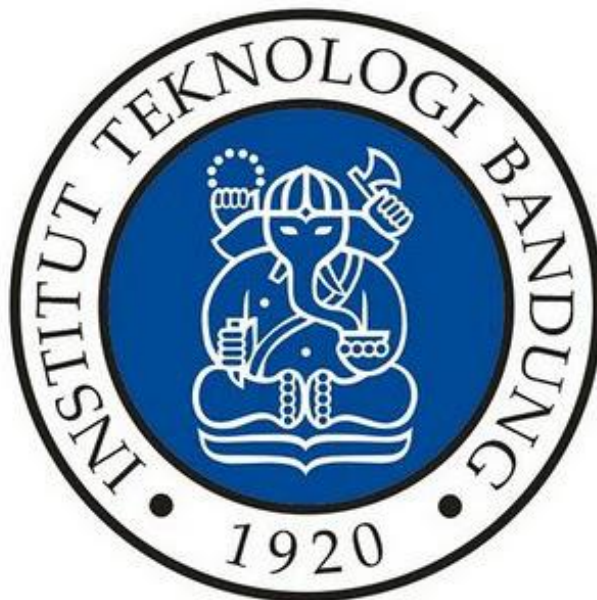


LAPORAN

Tugas Besar 3: 3D WebGL Articulated Model IF3260 Grafika Komputer

DISUSUN OLEH

13520124	Owen Christian Wijaya
13520125	Ikmal Alfaozi
13520159	Atabik Muhammad Azfa Shofi
13520165	Ghazian Tsabit Alkamil



**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2022/2023**

DAFTAR ISI

DAFTAR ISI	2
BAB I: DESKRIPSI	3
BAB II: HASIL IMPLEMENTASI	4
BAB III: MANUAL FUNGSIONALITAS	7
Pemilihan Articulated Model	7
Transformasi	8
Kontrol Kamera	10
Texture Mapping	12
BAB IV : KONTRIBUSI ANGGOTA	16

BAB I: DESKRIPSI

Laporan ini dituliskan sebagai bentuk dokumentasi dan *deliverable* dari Tugas Besar 3 mata kuliah IF3260 Grafika Komputer, program studi Teknik Informatika, Institut Teknologi Bandung, tahun ajaran 2022 / 2023.

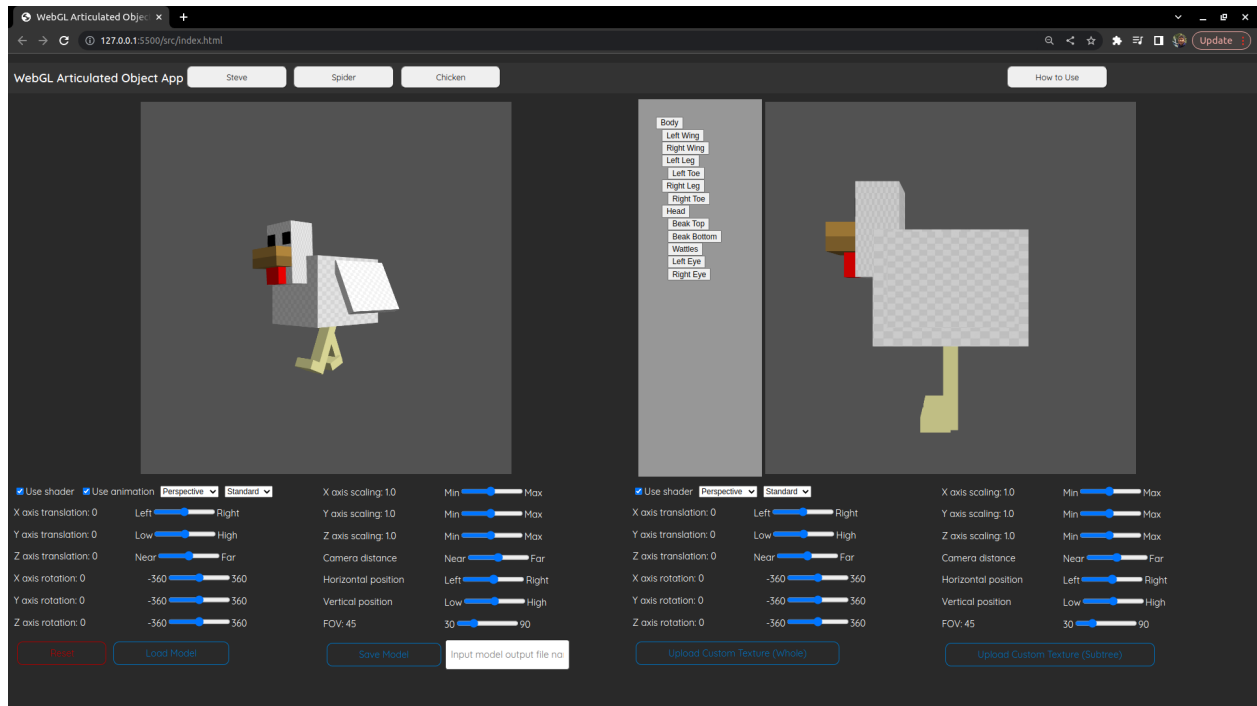
Tugas besar 3 bersangkutan dengan implementasi kaskas WebGL untuk mengembangkan sebuah aplikasi web yang dapat digunakan untuk melakukan transformasi terhadap *articulated model*. Sebuah *articulated model* adalah sebuah kesatuan dari beberapa model lain yang menyusun sebuah model lengkap. Sebuah *articulated model* mempunyai komponen-komponen yang mempunyai keterhubungan satu sama lain, misalnya tubuh manusia yang terhubung dengan kepala dan lengan, dan lengan atas yang terhubung dengan lengan bawah. Pada implementasinya, *articulated model* dibuat dengan membangun beberapa objek berbentuk kubus yang tersusun sedemikian rupa sehingga terdapat hubungan antar kubus dan beberapa bagian dapat digerakkan dengan sendi masing-masing.

Selain itu, tugas ini bersangkutan dengan simulasi beberapa jenis *mapping* dalam tekstur. Secara *default*, setiap *articulated model* mempunyai *texture* yang sudah disimpan di dalam direktori program. Terdapat tiga jenis simulasi tekstur yang dapat digunakan oleh pengguna:

1. **Standar:** simulasi tekstur pada permukaan objek, yang dapat digunakan dengan *lighting* atau tidak
2. **Bump mapping:** simulasi tekstur dengan kedalaman pada permukaan objek, menggunakan *normal*, *diffuse*, dan *depth texture*
3. **Reflection mapping:** simulasi permukaan reflektif dengan *cubemap* yang membuat objek seakan-akan memantulkan permukaan di sekitarnya

Tugas ini juga bersangkutan dengan pembuatan animasi dan manipulasi komponen pada setiap objek. Pada implementasi, bonus yang dikerjakan adalah bonus *component*, sehingga animasi yang dibuat hanya dapat dimainkan namun tidak dapat dimanipulasi secara bebas. Pengguna dapat melakukan manipulasi komponen menggunakan *canvas* komponen yang mengatur manipulasi masing-masing komponen.

BAB II: HASIL IMPLEMENTASI



Gambar 1. Tampilan program yang sudah dibuat

Program yang dibuat dapat digunakan untuk memilih *articulated model* yang hendak digunakan. Ada empat buah *articulated model* yang dapat digunakan pengguna:

1. **Steve**: objek berbasis humanoid
2. **Spider**: objek berbentuk laba-laba
3. **Chicken**: objek berbentuk ayam
4. **Dragon**: objek berbentuk naga

Setiap *articulated model* didefinisikan dalam sebuah *file* di direktori `src/js/model`, dan tekstur yang digunakan disimpan di direktori `src/js/model/texture`. Setiap model disusun oleh sekumpulan objek kubus (`Cube.js`) yang mempunyai titik tengah, panjang sisi, titik putar / pivot, *path* menuju tekstur, dan konfigurasi yang berguna untuk menyimpan nilai transformasi masing-masing kubus. Setiap *articulated model* juga menyimpan sebuah *array* berisi objek tekstur WebGL, untuk mengurangi pembuatan tekstur yang dibuat.

Pada saat inisialisasi model, model akan melakukan *loading* terhadap tekstur yang dibutuhkan objek. Terdapat dua buah fungsi *loading*, yaitu fungsi *loading* tekstur ke konteks `gl` dan ke konteks `componentGl`. Hal ini dikarenakan oleh implementasi bonus yang menerapkan dua buah canvas,

canvas pertama untuk menampilkan keseluruhan model dan canvas kedua untuk menampilkan komponen spesifik beserta komponen *child* dari komponen yang dipilih.

Pada saat instansiasi, model juga akan membuat *map* berupa relasi antara sebuah komponen dengan komponen lainnya. Setiap *key* menyatakan *parent component*, dan setiap *value* adalah sebuah *array* yang menyatakan nama-nama dari komponen lain yang merupakan *child component* dari *parent component*. Hal ini mewakili hubungan antara komponen dan akan digunakan untuk manipulasi *child component* pada canvas komponen. Selain itu, instansiasi akan menghasilkan *map* berisi konfigurasi untuk membuat serangkaian animasi. Animasi ini disimpan dalam pasangan *key-value*, di mana *key* mewakili nama komponen dan *value* berisi konfigurasi-konfigurasi (urutan *frame* animasi). Animasi ini akan di-*render* secara kontinu apabila pengguna memilih untuk mengaktifkan animasi.

Aturan transformasi dan perubahan proyeksi telah diimplementasikan di tugas besar 2 sebelumnya (dapat dilihat [di sini](#)), menggunakan aturan perkalian matriks. Untuk melakukan rotasi dan *scaling* pada komponen yang berpusat pada titik pivot, dibuat dua fungsi tambahan, yaitu *rotateWithPivot* dan *scaleWithPivot*, yang melakukan perputaran dan *scaling* terhadap titik *pivot*. Perubahan rotasi dan *scale* per komponen akan dilakukan melakukan fungsi tersebut. Untuk menyesuaikan dengan pergerakan *main component* / komponen utama dari model (misal, tubuh utama), komponen juga akan bergerak mengikuti rotasi dan *scaling* dari komponen utama.

Perubahan lain terletak pada instansiasi *buffer*. Karena semua objek dasar pada tugas ini adalah sebuah kubus, maka informasi mengenai *indices*, *normal vertices*, dan *buffer* lainnya akan sama. Oleh karena itu, perhitungan *vertices* akan dilakukan berdasarkan informasi titik tengah dan panjang sisi pada saat membuat *buffer*, dan untuk properti lain telah di-*hard code* pada pembuatan *buffer*. Hal ini dilakukan untuk meningkatkan *readability* pada *save file*, sehingga tidak penuh dengan informasi *vertices* dan *indices*.

Transformasi pada canvas komponen akan berpengaruh pada transformasi pada canvas utama. Karena setiap komponen dapat digerakkan secara bebas, maka nilai untuk menentukan transformasi akan ditentukan oleh dua buah konfigurasi:




- *model.globalConfig*: nilai konfigurasi yang akan di-*update* apabila melakukan transformasi pada canvas utama atau komponen utama pada canvas komponen
- *obj.config*: nilai konfigurasi setiap komponen

Penentuan nilai ini juga akan dibedakan sesuai dengan mode penggambaran yang dipilih. Terdapat tiga buah metode penggambaran yang ditentukan saat melakukan *rendering*:

1. **Whole**: menggambar model secara keseluruhan pada canvas utama
2. **Component**: menggambar komponen secara spesifik beserta *child component* dari komponen yang dipilih
3. **Animation**: menggambar animasi pada model secara kontinu

Pada mode *Animation*, nilai transformasi yang digunakan bukanlah nilai dari *slider*, melainkan nilai yang telah ditentukan pada *map* yang dibentuk saat instansiasi model. *Array* konfigurasi akan digunakan layaknya *frame* untuk menggambarkan posisi masing-masing komponen pada waktu tertentu. Animasi akan digambarkan pada canvas utama, sehingga pengguna masih dapat memanipulasi komponen pada canvas komponen.

Pada tugas ini, terdapat tiga buah mode tekstur yang dapat digunakan. Mode ini dapat digunakan juga pada kedua buah *canvas*, dengan *canvas* komponen mengikuti mode tekstur pada *canvas* utama namun dapat diubah secara bebas juga. Mode pertama adalah *texturing* standar yang digunakan dengan melakukan *loading* tekstur pada permukaan kubus. Pada mode ini, pengguna dapat mengaktifkan *shader* / *lighting* sehingga muncul bayangan. Pengguna juga dapat menggunakan gambar sendiri sebagai tekstur menggunakan tombol *Upload Custom Texture*, untuk keseluruhan objek ataupun komponen tertentu.

		
Mode standar menggunakan tekstur	Mode <i>bump mapping</i> dengan permukaan tampak tidak rata	Model <i>reflection mapping</i> yang memantulkan lingkungan

Mode kedua adalah *bump mapping*, yang mensimulasikan permukaan tidak rata pada permukaan model. Pada *bump mapping*, digunakan tiga buah *map texture* yang dapat dilihat pada direktori [js/model/texture/bump](#):

1. **Normal texture**: untuk mensimulasikan normal dari tekstur
2. **Diffuse texture**: permukaan kayu sebagai tekstur dasar
3. **Depth texture**: untuk mewakili kedalaman pada saat melakukan simulasi

Mode *bump mapping* yang digunakan adalah *parallax mapping*. Pada *parallax mapping*, perhitungan *lighting* dipengaruhi oleh *depth map* sebelum dilanjutkan ke penentuan warna. Hal ini dilakukan supaya *bump* yang tertutupi juga akan diperhitungkan dan efek *parallax* lebih tampak. Perhitungan ini dilakukan pada *shader program*, sehingga penentuan warna *fragment* pada *shader program* akan dibedakan sesuai mode *rendering*. Untuk melakukan *bump mapping*, diperlukan informasi mengenai *tangent vertex* dan *bitangent vertex* untuk melakukan perhitungan warna dan efek timbul dari *fragment*.

Mode ketiga adalah *reflection mapping*, yang melakukan *rendering* sedemikian rupa supaya objek seakan-akan terbuat dari permukaan reflektif dan memantulkan permukaan di sekitarnya. Untuk melakukan *reflection mapping*, digunakan *cube map* untuk melakukan *loading texture* di lingkungan sekitar objek. Perhitungan warna *fragment* pada *shader program* akan dilakukan untuk memberikan efek pantulan dari posisi kamera dan matriks transformasi. Untuk mendapatkan hasil matriks transformasi saja, dilakukan perkalian antara *modelViewMatrix* dan hasil invers *lookAtMatrix*, karena sebelumnya *modelViewMatrix* adalah hasil penggabungan matriks transformasi dan matriks *lookAt*. Hasilnya adalah tampilan objek yang “reflektif” dan memantulkan lingkungan sekitarnya.

Bonus yang dikerjakan adalah **component tree** dan **subtree component view**. *Component tree* adalah sekumpulan *button* pada sisi kiri *canvas component* yang dapat digunakan untuk mempermudah pemilihan komponen dan *child component*. *Subtree component view* melibatkan manipulasi komponen tertentu menggunakan *canvas* komponen dan *slider* untuk *canvas* komponen.

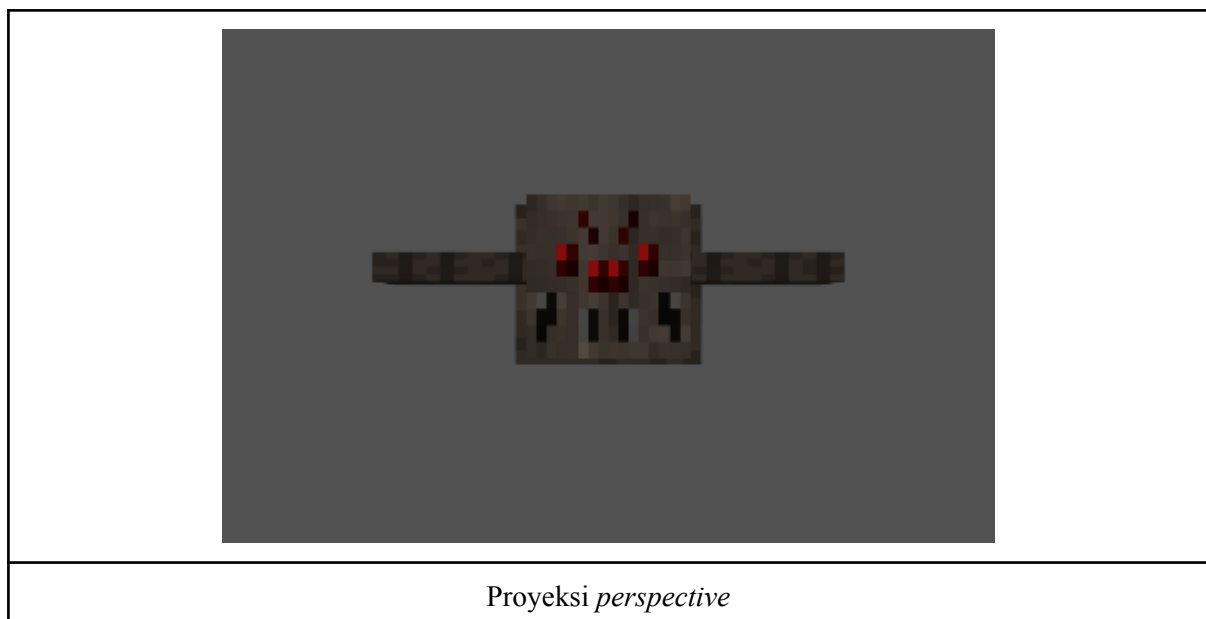
BAB III: MANUAL FUNGSIONALITAS

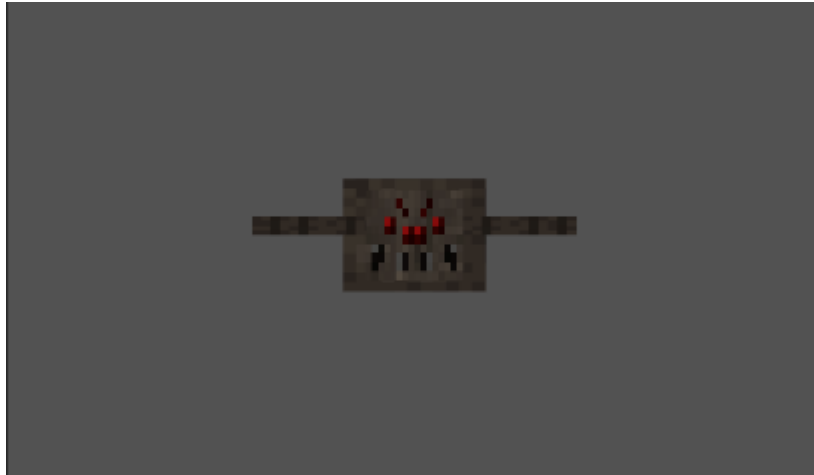
Pemilihan Articulated Model

1. Pengguna dapat memilih salah satu dari tiga objek:
 - Steve (**secara default model yang akan di load adalah model Steve**)
 - Spider
 - Chicken



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 juga dapat mengubah texture yang ada pada model baik secara keseluruhan atau hanya untuk komponen subtree tertentu saja.





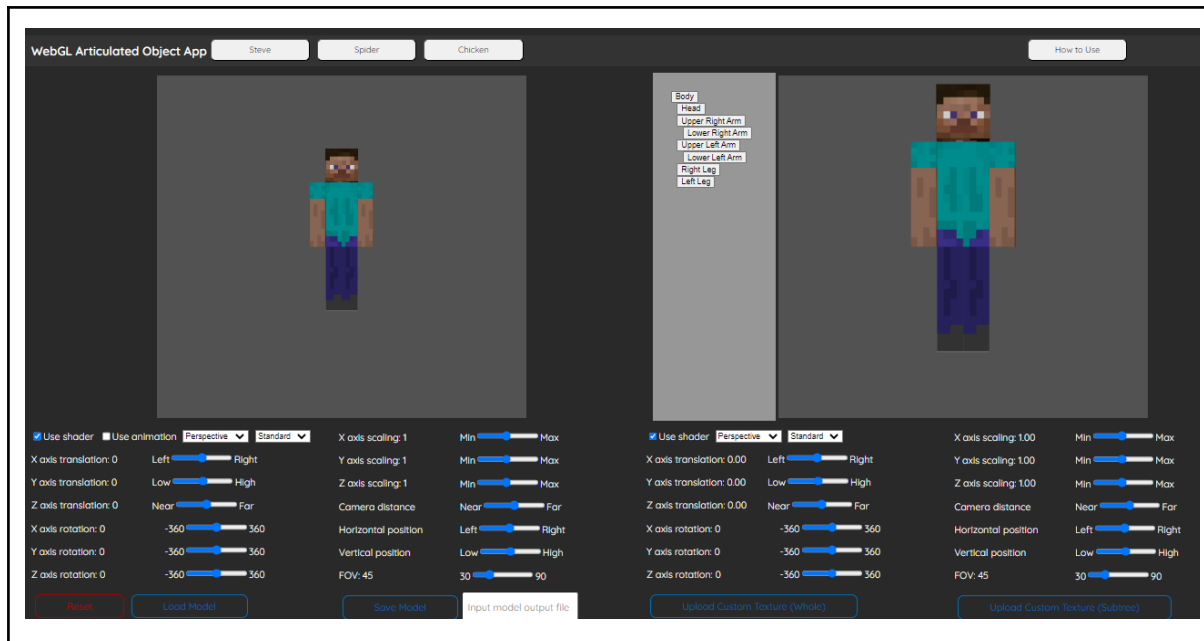
Proyeksi *orthographic*



Proyeksi *oblique*

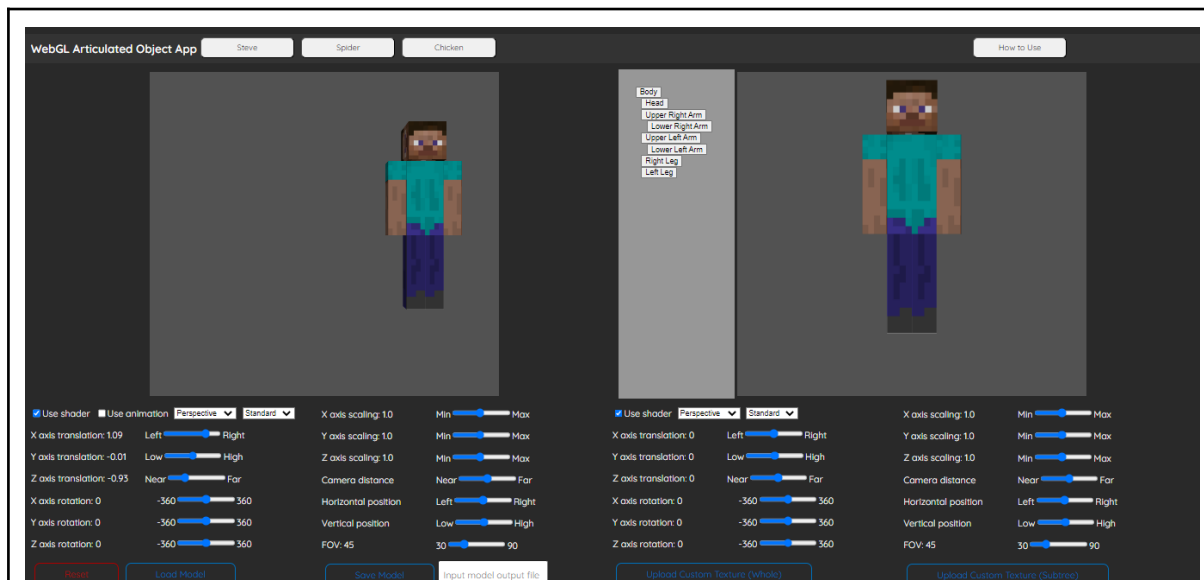
Transformasi

1. Pengguna dapat melakukan proses translasi objek pada sumbu X, Y, dan Z menggunakan *translation slider* yang tersedia. Proses translasi dapat dilakukan untuk keseluruhan *articulated model* atau khusus untuk komponen tertentu yang ada pada *articulated model*.
2. Pengguna dapat melakukan proses rotasi objek pada sumbu X, Y, dan Z menggunakan *rotation slider* yang tersedia. Proses rotasi dapat dilakukan untuk keseluruhan *articulated model* atau khusus untuk komponen tertentu yang ada pada *articulated model*.
3. Pengguna dapat melakukan proses *scaling* objek pada sumbu X, Y, dan Z menggunakan *scaling slider* yang tersedia. Proses *scaling* juga dapat dilakukan per komponen pada *articulated model*.

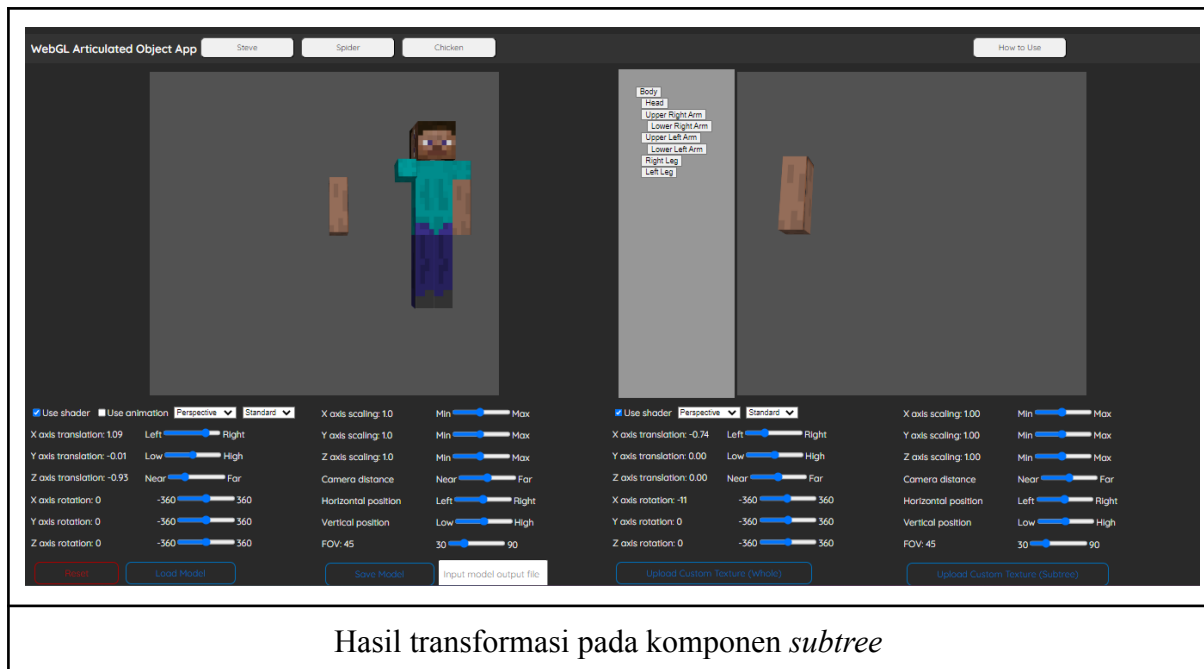


Pada web aplikasi terdapat dua fitur untuk melakukan proses transformasi, yang pertama adalah fitur untuk transformasi keseluruhan objek dan transformasi untuk komponen *subtree*. Dapat dilihat pada gambar di atas, *controller* untuk objek keseluruhan terdapat pada bagian kiri dan *controller* untuk komponen pada *subtree* terdapat pada bagian kanan.

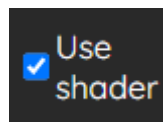
Proses transformasi yang dilakukan pada komponen *subtree* akan langsung muncul pada canvas yang menampilkan *articulated model* secara keseluruhan.



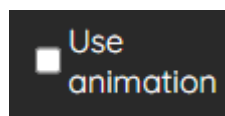
Hasil transformasi pada keseluruhan *articulated model*



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.
5. Pengguna dapat mengaktifkan animasi pada *articulated model* menggunakan *checkbox* "Use animation". Setiap *articulated model* memiliki animasi khususnya masing masing.



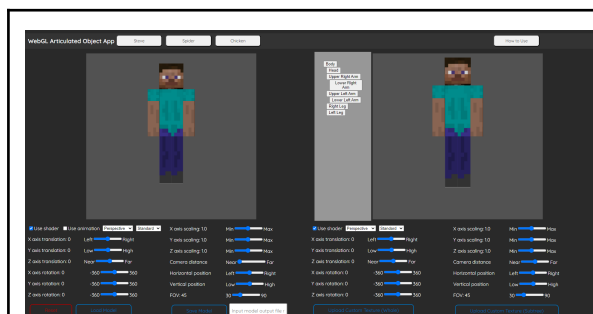
- Menyalakan *checkbox* akan membuat *articulated model* melakukan animasi sesuai dengan yang telah ditentukan.
- Menonaktifkan *checkbox* akan membuat *articulated model* berhenti melakukan animasi dan kembali ke *state* sebelum awal.

Kontrol Kamera

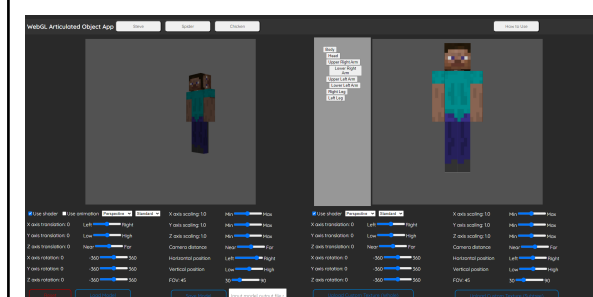
- Penggunaan dapat mengatur jarak kamera dari objek menggunakan slider "Camera distance".
- Pengguna dapat menggerakkan kamera mengelilingi objek secara horizontal menggunakan slider "Horizontal Position".

- 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*.

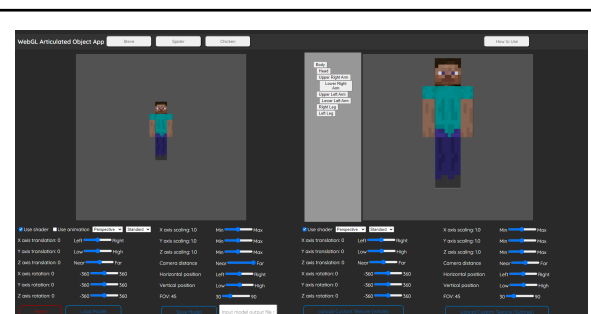
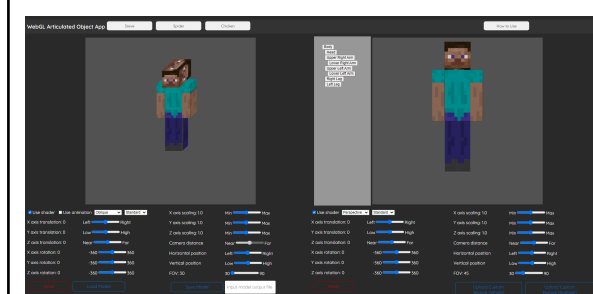
Untuk proyeksi orthogonal, tidak perlu dilakukan pengaturan *FOV* karena tampilan tegak lurus. Untuk proyeksi *oblique*, *FOV* mengatur sudut antara kamera dengan bidang proyeksi. Pada *FOV* = 90 derajat, tampilan *oblique* akan tampak seperti *orthogonal*.



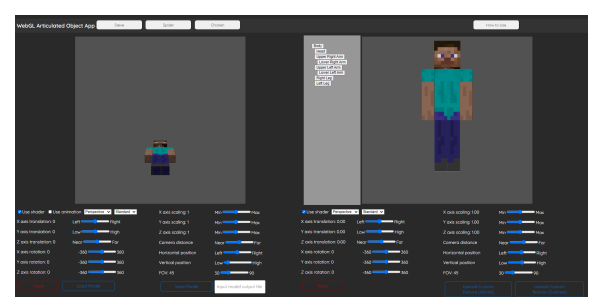
Hasil mendekatkan kamera pada model



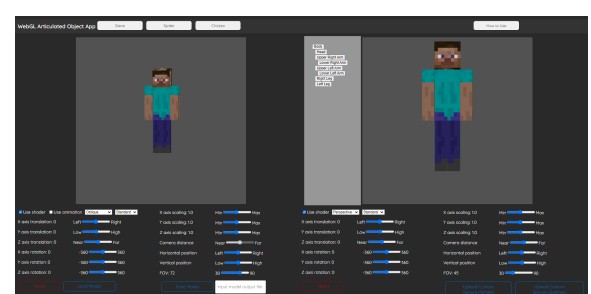
Hasil menggerakkan posisi kamera secara horizontal



Hasil menjauhkan kamera dari model



Hasil menggerakkan posisi kamera secara vertical



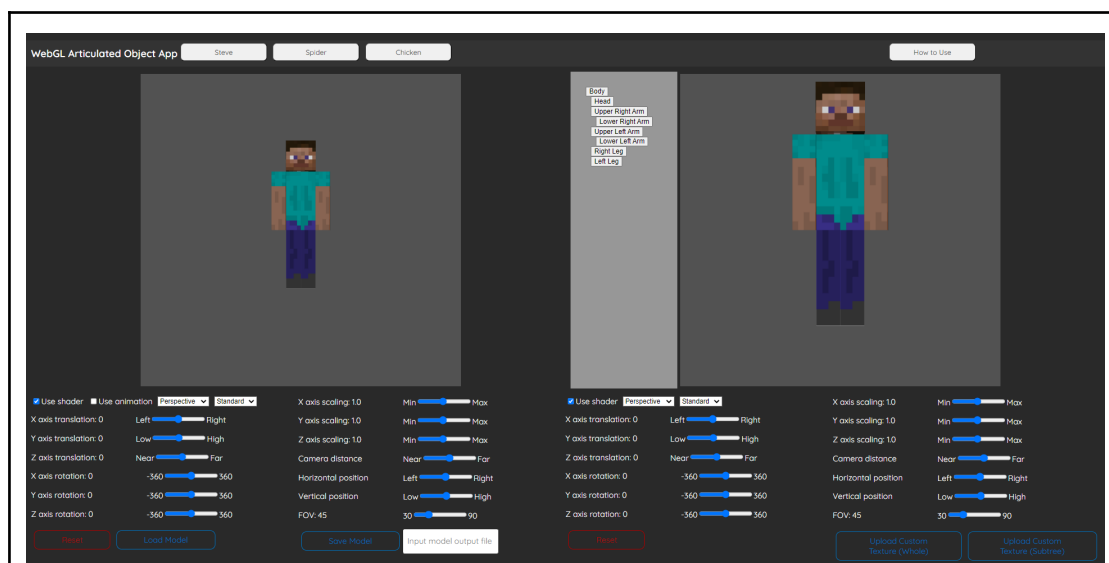
Perbandingan perbedaan sudut proyeksi pada proyeksi *oblique* (kiri = sudut rendah, kanan = sudut tinggi)

Texture Mapping

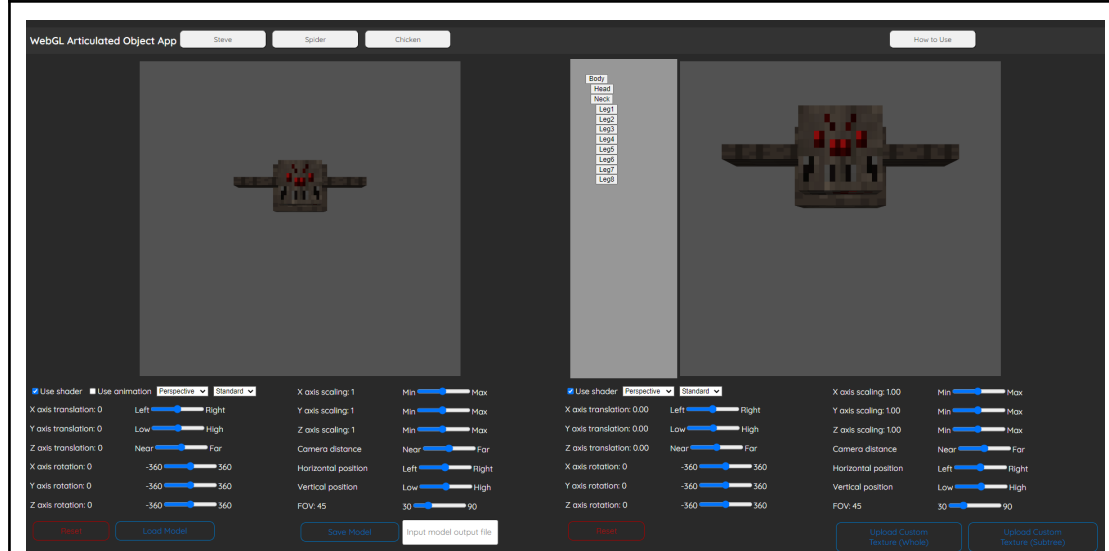
Pada web aplikasi ini pengguna dimungkinkan untuk melakukan proses pengubahan texture yang digunakan oleh *articulated model*. Texture yang digunakan pada web aplikasi ini adalah sebagai berikut:

1. Texture Default

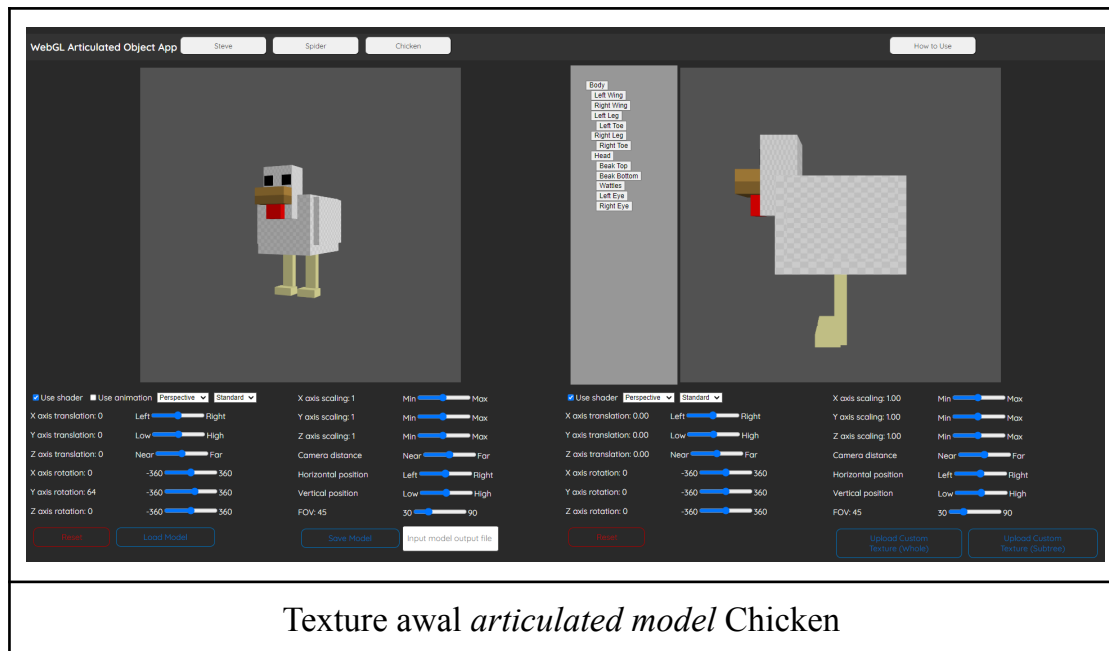
Texture default adalah texture awal yang akan di load pada awal program dijalankan. Setiap *articulated model* memiliki texture default yang berbeda.



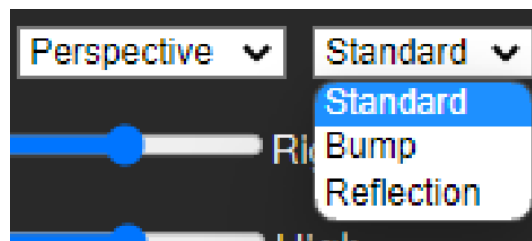
Texture awal *articulated model* Steve



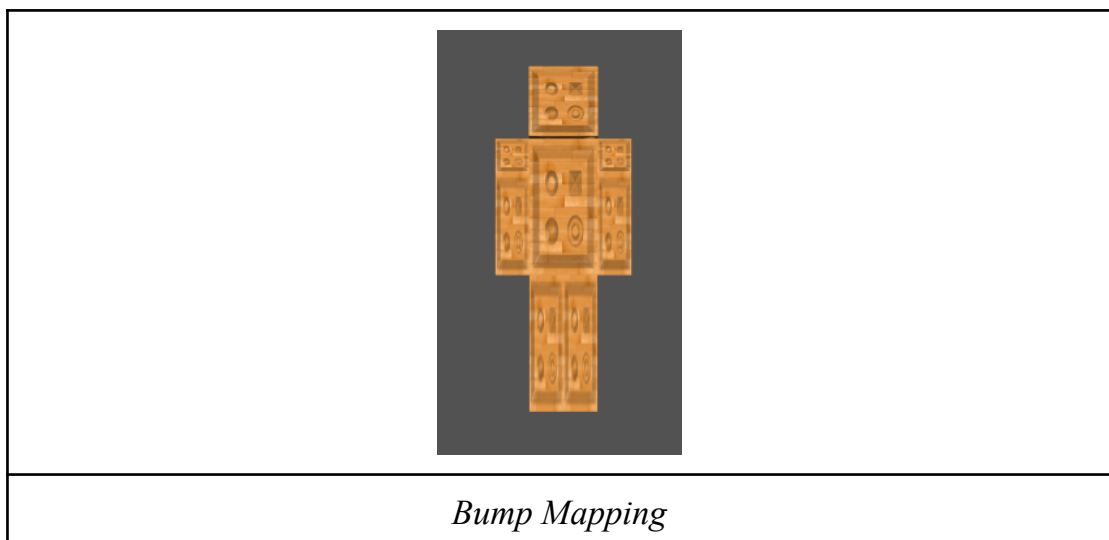
Texture awal *articulated model* Spider

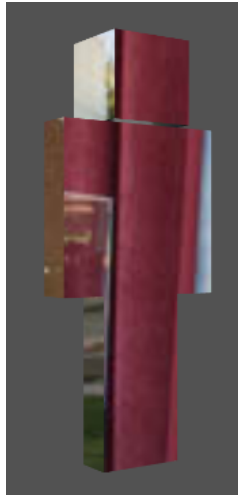


2. Bump dan Reflection Mapping



Pengguna dapat menggunakan *bump* atau *reflection mapping* dengan memilih opsi yang tersedia.

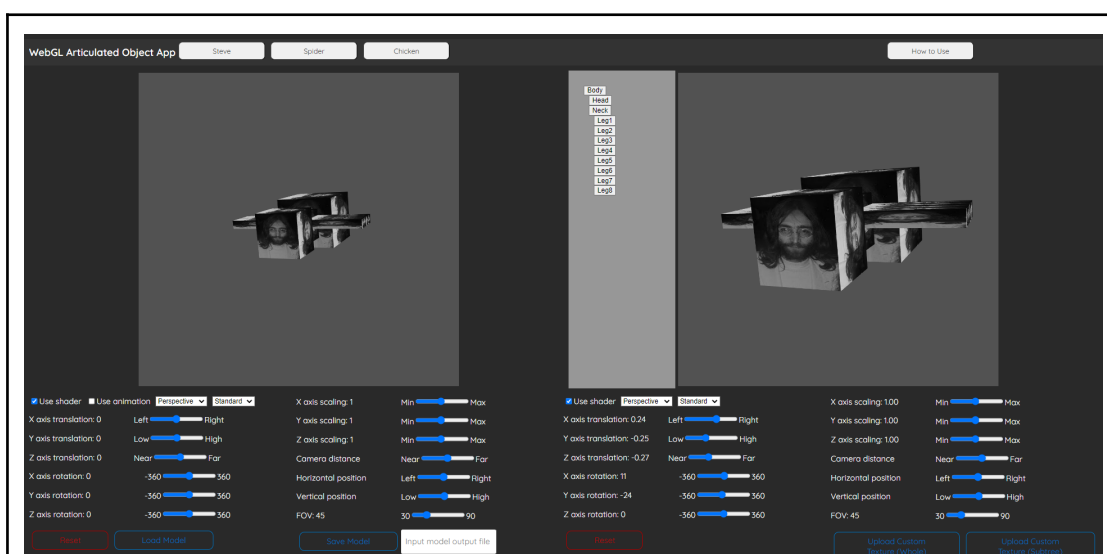




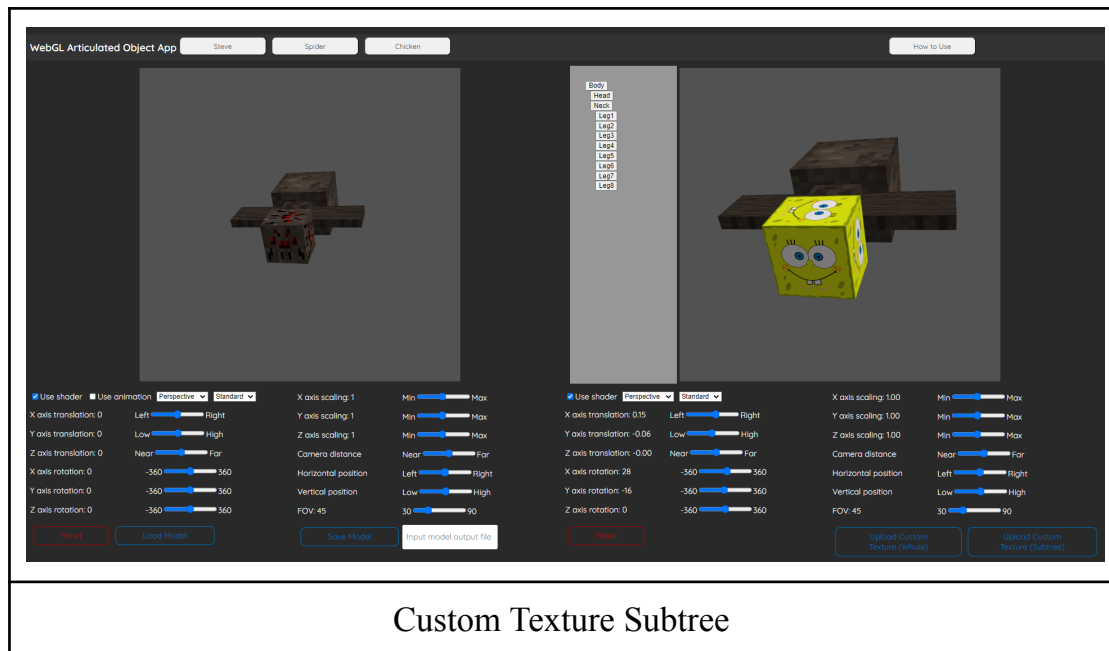
Reflection Mapping

3. Custom Texture

Pada web aplikasi ini juga pengguna dapat melakukan custom terhadap texture yang digunakan pada *articulated model*. Custom texture dapat dilakukan untuk keseluruhan *articulated model* atau hanya untuk komponen *subtree* saja. Penggunaan custom texture dapat dilakukan dengan menekan tombol dan kemudian mengupload file texture yang ingin digunakan.



Custom Texture Keseluruhan



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 menuliskan nama *file* di sebelah tombol **Save Model**, lalu menekan tombol **Save Model**

BAB IV : Kontribusi Anggota

NIM	Nama	Kontribusi
13520124	Owen Christian Wijaya	Integrasi transformasi, pembuatan bonus, pembuatan <i>bump</i> dan <i>reflective mapping</i> Articulated Model Steve
13520125	Ikmal Alfaozi	Articulated Model Chicken beserta animasi, testing dan debugging
13520159	Atabik Muhammad Azfa Shofi	Articulated Model Dragon beserta animasi, testing dan debugging
13520165	Ghazian Tsabit Alkamil	Articulated Model Spider beserta animasi, testing dan debugging