PERANCANGAN APLIKASI GRAFIS UNTUK MENAMPILKAN MOTIF PAKAIAN DENGAN PROYEKSI VISUAL 3D MENGGUNAKAN DIRECTX

SKRIPSI

KONSENTRASI REKAYASA KOMPUTER

Diajukan untuk memenuhi sebagian persyaratan
Memperoleh gelar Sarjana Teknik

Disusun oleh:
IRHAMNI LUQMAN
NIM. 0510633046

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
UNIVERSITAS BRAWIJAYA
FAKULTAS TEKNIK
MALANG
2012

PENGANTAR

Puji syukur kehadirat Allah SWT, Sang Maha Pencipta yang telah memberikan limpahan rahmat, hidayah, serta anugerah sehingga penulis dapat menyelesaikan Tugas Akhir dengan judul "**Perancangan Aplikasi Grafis untuk Menampilkan Motif Pakaian dengan Proyeksi Visual 3D Menggunakan DirectX**". Tugas Akhir ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Teknik di Jurusan Teknik Elektro Konsentrasi Rekayasa Komputer Fakultas Teknik Universitas Brawijaya Malang. Tidak banyak yang bisa penulis sampaikan kecuali ungkapan terima kasih kepada berbagai pihak yang telah dengan tulus ikhlas memberikan bimbingan, arahan, dan dukungan hingga penulisan tugas akhir ini dapat terselesaikan. Pada kesempatan kali ini, penulis mengucapkan banyak terima kasih kepada:

1. Ayah dan Ibu yang senantiasa memberikan dukungan serta do'a yang tak ternilai harganya untuk terus maju hingga terselesaikan skripsi ini.
2. Dr. Ir. Sholeh Hadi Pramono, MS selaku Ketua Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
3. M. Azis Muslim, S.T.,M.T.,Ph.D selaku Sekretaris Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
4. Bapak Waru Djuriatno, ST., MT. selaku KKDK Teknik Informatika dan Komputer yang telah banyak memberikan bimbingan, masukan dan arahan dalam penyusunan tugas akhir ini
5. Bapak Muhammad Aswin, Ir., MT. selaku dosen pembimbing I yang telah sudi dan senantiasa meluangkan waktu untuk memberikan bimbingan, masukan, arahan, dan bantuan yang luar biasa berarti dalam penyusunan tugas akhir ini.
6. Bapak Adharul Muttaqin, ST.,MT selaku dosen pembimbing II yang telah sudi dan senantiasa mendorong, memberikan motivasi, bimbingan, masukan, arahan, dan bantuan yang luar biasa berarti dalam penyusunan tugas akhir ini.
7. Bapak dan Ibu Dosen serta karyawan jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.

8. Rekan-rekan TEUB 2005, Fachmanda Idyanto'05, M. Bernie Fityanto'05, Andika Wijaya S.'05, Samsul Arif'05, M. Nur Lanta A.'05, Ahmad Afrianto'05, Lastono Risman'05, Bagus Arifin'05, Heroe Soebagio'05, Rinanto Demi'05, Raffi Ananta'05, Prasetyo Adi'05, Andi Kurniawan'05, Christian Jonathan'05, Agung Pamungkas'05, Antok Tri S.'05, Eka Panji'05, Agung Budi U.'05, Ade '05, Hesieka S. Dona'05, dan rekan-rekan jurusan TEUB lain yang telah banyak membantu serta dukungan hingga tugas akhir ini dapat diselesaikan.
9. Rekan-rekan Teknik Mesin 2005, Ferry Dwi Cahya'05, Aris Kurniawan M'05, Yuswandita Toesa M'05, Anis Fuadi M'05, Arif M'05, Sulistyawan M'05, dan Husein Irsyad M'05 yang telah banyak memberikan kesan selama menjalani kuliah di FTUB.
10. Serta semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun tidak langsung demi terselesaikannya tugas akhir ini.

Penulis menyadari sepenuhnya bahwa Tugas Akhir ini masih banyak kekurangan. Segala kritik dan saran yang membangun akan penulis terima demi kesempurnaan skripsi ini. Harapan penulis semoga skripsi ini bermanfaat bagi pembaca dan khususnya mahasiswa jurusan teknik elektro dimasa yang akan datang.

Malang, Juli 2012

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Irhamni Luqman (0510633046), Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya, Juli 2012, Perancangan Aplikasi Grafis untuk Menampilkan Motif Pakaian dengan proyeksi visual 3D Menggunakan DirectX, Dosen pembimbing: Muhammad Aswin, Ir., MT., Adharul Muttaqin, ST., MT.,

Semakin maraknya *distribution outlet* atau umum dikenal dengan istilah distro menuntut adanya simulasi pemetaan sketsa motif 2D pada objek pakaian 3D untuk menekan biaya produksi. Sketsa motif 2D terkesan minim realistis dibandingkan dengan ilusi perspektif dari hasil proyeksi 3D sehingga dibutuhkan adanya sebuah *software* yang atraktif, interaktif, dan terutama simpel sebagai bagian dari solusi simulasi.

DirectX merupakan API berbasis COM yang mengenkapsulasi fungsi-fungsi penanganan spesifik terhadap operasi aras rendah perangkat keras multimedia. Dengan adanya kemudahan implementasi akses terhadap kartu grafis maka DirectX dirasa tepat dalam melakukan pemrograman grafis yang berorientasi pada 3D.

Pada tugas akhir ini, perangkat lunak aplikasi grafis dibuat menggunakan Bahasa C++ dengan implementasi objek-objek COM DirectX sebagai antarmuka antara program aplikasi Win32 dengan perangkat keras grafis dan input. Proses yang dilakukan untuk memproyeksikan objek 3D adalah dengan menggunakan konkatenasi matriks pada tahapan-tahapan transformasi dengan menggunakan matriks ruang, matriks pandang, dan matriks proyeksi perspektif. Metode pengalamatan yang digunakan untuk mendistribusikan alamat gambar tekstur adalah dengan cara planar, spheris, dan silindris.

Berdasarkan pengujian dan analisis sistem, perangkat lunak berhasil memproyeksikan objek 3D secara perspektif menggunakan perkalian matriks meskipun masih ada anomali karena tidak memperhitungkan nilai z sebagai faktor pengali. Kesesuaian gambar asli tekstur berhasil dipetakan dan dimanipulasi pada objek pakaian 3D yang memiliki bidang datar dan bidang lengkung dengan menggunakan metode pengalamatan planar, spheris, atau silindris.

Kata kunci : motif 2D, DirectX, proyeksi 3D, pengalamatan tekstur

BAB I
PENDAHULUAN

1.1 Latar Belakang

Proses desain yang dilakukan dengan menggunakan media 2D saat ini sekiranya masih dirasa kurang karena minimnya ilusi dari kesan realistis. Bagi para pemilik *distribution outlet* (umum dikenal dengan istilah distro) yang terbiasa melakukan desain pakaian dengan media 2D, mungkin pernah merasa penasaran seperti apa manifestasi 3D dari hasil desain 2D yang sudah mereka buat sebelum benar-benar melakukan upaya produksi. Tentu setelah pakaian diproduksi barulah mereka dapat melihat hasil desain 2D dari sudut pandang perspektif 3D. Hal ini kurang efisien karena proses desain umumnya dilakukan berulang kali dan pasti memerlukan biaya produksi. Dari alasan yang telah disebutkan dapat ditarik kesimpulan adanya tuntutan untuk sebuah *software* yang dapat menampilkan hasil sketsa motif pakaian 2D terhadap objek pakaian 3D secara atraktif, interaktif, dan terutama simpel sebagai sebuah solusi simulasi.

DirectX merupakan *API (Application Programming Interface)* yang berbasis *COM (Component Object Model).* COM DirectX merupakan antarmuka pemrograman berbasis objek yang mengenkapsulasi fungsi-fungsi penanganan spesifik terhadap operasi aras rendah perangkat keras multimedia. Konsep COM memberikan kemudahan dalam implementasi karena seorang programmer tidak lagi diharuskan melakukan akomodasi terhadap spesifikasi fisik perangkat keras multimedia.

DirectX Graphics adalah bagian dari DirectX yang berorientasi pada pemrograman grafis, menyediakan fasilitas antarmuka untuk berkomunikasi dengan *display adapter* seperti kartu grafis melalui *HAL (Hardware Abstraction Layer).* Dengan kemudahan akses terhadap kartu grafis maka DirectX dirasa tepat dalam melakukan pemrograman grafis yang berorientasi pada 3D.

Dari alasan-alasan yang telah disebutkan mendorong penulis untuk membuat sebuah program aplikasi grafis yang dapat menampilkan hasil pemetaan sketsa motif pakaian berupa gambar 2D pada objek pakaian 3D dengan memanfaatkan fasilitas-fasilitas yang ada di dalam DirectX.

1.2 Rumusan Masalah

Berdasarkan latar belakang, maka dapat ditetapkan rumusan masalah sebagai berikut :

1. Bagaimana cara merancang dan membuat sebuah aplikasi yang dapat memproyeksikan objek pakaian 3D secara perspektif.
2. Bagaimana menentukan keberhasilan proyeksi perspektif yang didapat dari tahap perancangan.
3. Bagaimana cara memberi alamat gambar 2D pada titik-titik milik objek 3D supaya gambar dapat dipetakan pada objek 3D.
4. Bagaimana kesesuaian tekstur asli setelah dipetakan pada objek pakaian 3D dengan menggunakan metode-metode pengalamatan yang didapat dari tahap perancangan.
5. Bagaimana cara memanipulasi porsi dan posisi gambar 2D terhadap objek pakaian 3D.

1.3 Batasan Masalah

Agar penulisan ini lebih terarah dan sesuai dengan tujuan yang diinginkan maka permasalahan perlu dibatasi sebagai berikut:

1. Pemrograman difokuskan pada subset DirectX yaitu DirectX Graphics.
2. Penanganan data yang diproses secara manual merupakan data vektor sedangkan penanganan data raster diproses secara metodikal oleh DirectX
3. Objek 3D yang dibuat dan digambar secara primitif merupakan 3 bangun sederhana yaitu kubus, bola, dan silinder.
4. Objek 3D berupa pakaian disusun dari daftar verteks dan indeks yang disimpan dalam file berekstensi .x dan digenerasi dari 3DSMAX.
5. *Gender* laki-laki dipilih sebagai batasan dalam perancangan model 3D di dalam aplikasi.
6. Versi yang digunakan yaitu versi DirectX 8.0 dan DirectX 9.0 keatas.
7. Format file gambar yang bisa dikenali merupakan format file yang disediakan akomodasi oleh DirectX yaitu .bmp, .dds, .dib, .hdr, .jpg, .pfm, .png, .ppm, dan .tga.

8. Bahasa pemrograman yang digunakan adalah Bahasa C.
9. IDE yang digunakan yaitu Visual C++
10. Tidak membahas perihal animasi mesh.

1.4 Tujuan

Tujuan penulisan skripsi adalah sebagai berikut :

1. Membuat program aplikasi dengan keluaran berupa tampilan proyeksi perspektif objek pakaian 3D.
2. Membuat program aplikasi agar *user* dapat memberikan masukan berupa gambar 2D untuk dipetakan pada objek pakaian 3D.
3. Membuat program aplikasi agar user dapat memanipulasi gambar 2D setelah terpetakan pada objek pakaian 3D.
4. Membuat program aplikasi dengan orientasi pemetaan adhesif bagi *user* yang membutuhkan simulasi tempel sketsa motif pada objek pakaian 3D.

1.5 Manfaat

Diharapkan manfaat yang dapat diperoleh melalui pengerjaan skripsi ini adalah:

a) Bagi Penyusun

1. Menerapkan ilmu yang telah diperoleh dalam Teknik Elektro konsentrasi sistem informatika dan komputer Universitas Brawijaya.
2. Menperoleh pengetahuan mengenai pemrograman grafis 3D beserta kegunaan dari DirectX.

b) Bagi Pengguna

1. Mendapatkan tampilan visual pemetaan gambar motif pakaian terhadap objek pakaian 3D tanpa harus membuat model 3D.
2. Melakukan manipulasi tampilan teksur terpetakan pada objek pakaian 3D

1.6 Sistematika Penulisan

Sistematika penulisan laporan skripsi ini adalah sebagai berikut :

BAB I Pendahuluan

Dalam bab ini akan dijelaskan latar belakang, rumusan masalah, batasan masalah, tujuan dan manfaat, metode penelitian dan sistematika penulisan laporan tugas akhir.

BAB II Tinjauan Pustaka

Dalam bab ini akan dibahas dan dijelaskan mengenai dasar teori yang menjadi landasan dan mendukung pelaksanaan penulisan tugas akhir.

BAB III Metodologi Penelitian

Dalam bab ini akan membahas tentang metode yang dilakukan penulis untuk menyelesaikan laporan tugas akhir.

BAB IV Perancangan

Menjelaskan langkah-langkah perancangan untuk membuat aplikasi grafis penampil motif pakaian.

BAB V Implementasi

Menjelaskan langkah-langkah pembuatan program aplikasi dengan menggunakan Bahasa C++ dan DirectX.

BAB VI Pengujian

Dalam bab ini akan disampaikan hasil pengujian dari aplikasi yang telah dibuat.

BAB VII Kesimpulan dan Saran

Memuat kesimpulan yang diperoleh dari analisis algoritma dan pengembangan aplikasi, serta saran-saran untuk pengembangan lebih lanjut.

BAB II
TINJAUAN PUSTAKA

2.1 API *(Application Programming Interface)*

API*(Application Programming Interface)* merupakan sekumpulan instruksi maupun standar atau protokol yang berguna sebagai antarmuka agar suatu aplikasi bisa berkomunikasi dengan aplikasi lain. Windows API merupakan antarmuka antara aplikasi Win32 (yaitu aplikasi yang berjalan di dalam sistem operasi Microsoft Windows) dengan program aplikasi lainnya.

2.2 COM (Component Object Model)

COM*(Component Object Model)* adalah model pemrograman yang berbasis objek. COM bersifat mengenkapsulasi prosedur-prosedur sesuai dengan klasifikasi kebutuhan dan bersifat *language independent* artinya tidak memiliki ketergantungan terhadap objek COM lainnya. Karena COM berbasis objek maka dalam pengaksesan prosedur didalam COM dibutuhkan referensi pointer atau pendefinisian objek COM. Sebuah objek COM minimal mempunyai antarmuka bernama IUnknown yang mempunyai 3 *method* sebagai berikut :

a. QueryInterface

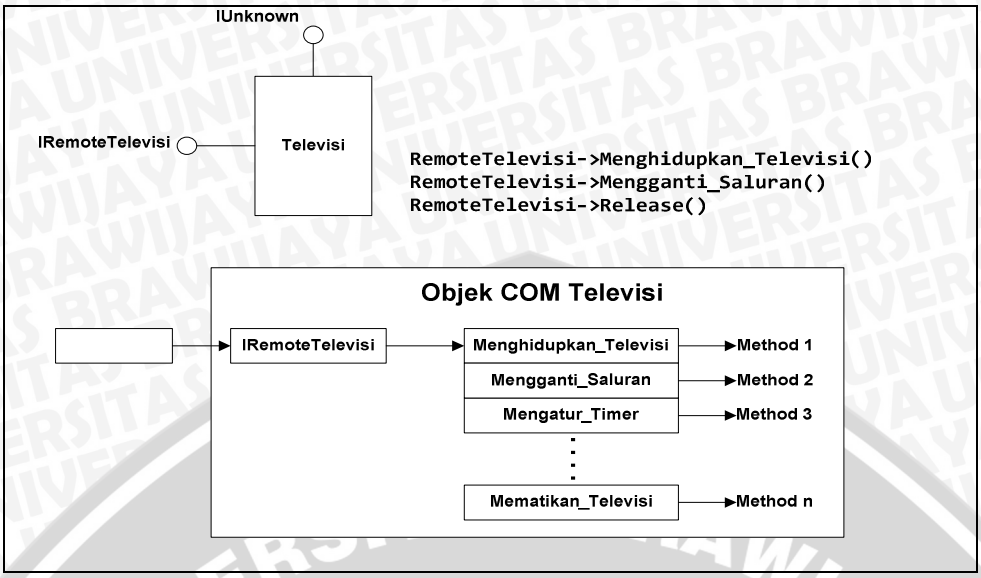
   Yaitu method yang digunakan untuk mendapatkan alamat pointer dari objek COM lainnya.

b. Addref

   Yaitu method yang digunakan apabila sebuah objek COM sedang direferensi oleh objek COM lain sehingga mencegah penghapusan objek COM yang direferensi.

c. Release

   Yaitu method yang digunakan apabila sebuah objek selesai direferensi sehingga dapat dihapus secara otomatis oleh sistem.



Gambar 2.1 Contoh objek antarmuka COM

2.3 DirectX

DirectX adalah salah satu API milik Microsoft Windows yang berbasis COM. DirectX memfokuskan pada akselerasi pengaksesan perangkat-perangkat keras multimedia seperti kartu grafis *(video card)*, perangkat audio, *joystick* dan sebagainya.

DirectX terbagi menjadi beberapa subset atau klasifikasi antara lain :

1. DirectX Graphics

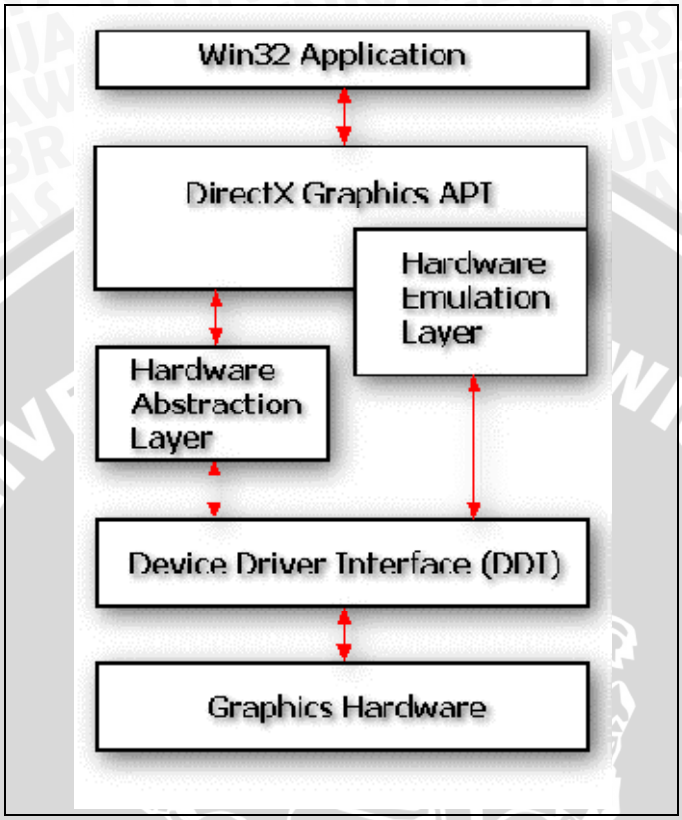
   DirectX Graphics adalah API yang bertugas melakukan penanganan grafis. Sebelum DirectX versi 8.0 DirectX Graphics terbagi menjadi 2 yaitu Direct3D (untuk penanganan grafis 3D) dan DirectDraw (untuk penanganan grafis 2D). Saat ini masih banyak yang mengidentikkan DirectX Graphics sebagai Direct3D karena sebagaian besar fungsi, struct, dan antarmuka DirectX Graphics dimulai dengan awalan D3D.

2. DirectInput

   DirectInput adalah API yang bertugas melakukan penanganan perangkat keras masukan seperti *mouse* dan *keyboard*. DirectInput berisikan fungsi-fungsi untuk membaca dan me-*manage* nilai masukan dari perangkat keras tersebut.

Pada lapisan hardware (*hardware layer)* DirectX Graphics berada dan menghubungkan antara aplikasi Win32 dengan *HAL(Hardware Abstraction*

Layer). HAL adalah program yang disediakan oleh manufaktur *graphic hardware* agar pengendali devais *(device driver)* milik sebuah produk dapat berinteraksi dengan DirectX Graphics.



Gambar 2.2 Hubungan antara DirectX dengan aplikasi dan HAL

Keunggulan yang dimiliki DirectX Graphics antara lain adalah

a. Berbasis COM sehingga mudah dalam implementasi.
b. Bersifat *backward compatible* sehingga versi sebelumnya masih dapat berjalan dengan baik pada versi terbaru.
c. Membebaskan pemrogram *(programmer)* terhadap masalah pengakomodasian perangkat keras grafis.

2.4 Dasar-dasar Perhitungan 3D pada Pemrograman Grafis

2.4.1 Vektor

Vektor berperan penting pada pemrograman grafis karena dapat memberikan informasi letak, arah, dan jarak *(magnitude)* sebuah titik atau dikenal sebagai verteks pada komputer grafis. Selain itu vektor juga dapat digunakan untuk mencari normal atau unit vektor sebuah verteks maupun bidang datar *(plane)* yang berguna untuk menentukan intensitas cahaya yang diterima oleh sebuah verteks. Selain kegunaan tersebut juga terdapat cross product yang dapat digunakan untuk mencari sebuah vektor yang tegak lurus terhadap dua vektor dan dot product yang dapat digunakan untuk mencari besar sudut antara dua vektor.

2.4.2 Matriks

Dalam pemrograman 3D matriks tidak kalah penting dibandingkan dengan vektor karena komposisi matriks serupa dengan array pada bahasa pemrograman. Sebuah titik atau verteks atau vektor dapat juga dinyatakan sebagai sebuah matriks sehingga dapat dilakukan perkalian antar matriks sebagai representasi dari perkalian antara vektor dengan matriks. Berikut ilustrasi perkalian antara sebuah vektor dengan matriks yang menghasilkan vektor baru

$$[\boldsymbol{x} \quad \boldsymbol{y} \quad \boldsymbol{z}] = \begin{bmatrix} \boldsymbol{m_{11}} & \boldsymbol{m_{12}} & \boldsymbol{m_{13}} \\ \boldsymbol{m_{21}} & \boldsymbol{m_{22}} & \boldsymbol{m_{23}} \\ \boldsymbol{m_{31}} & \boldsymbol{m_{32}} & \boldsymbol{m_{33}} \end{bmatrix} = [\boldsymbol{x'} \quad \boldsymbol{y'} \quad \boldsymbol{z'}]$$

2.4.3 Matriks Homogen *(Homogeneous Matrix)*

Matriks homogen adalah sebuah matriks yang ditambahkan elemen ke-n untuk penyeragaman dimensi matriks supaya bisa dilakukan perkalian antar matriks. Proses pengembalian nilai matriks yang seragam atau homogen dinamakan *homogenizing*. Jika sebuah vektor mempunyai empat dimensi seperti matriks $[\boldsymbol{x} \quad \boldsymbol{y} \quad \boldsymbol{z} \quad \boldsymbol{w}]$ maka untuk mengembalikan lagi ke koordinat tiga dimensi perlu dibagi dengan elemen homogen yaitu elemen w, sehingga diperoleh $\left[\frac{\boldsymbol{x}}{\boldsymbol{w}} \quad \frac{\boldsymbol{y}}{\boldsymbol{w}} \quad \frac{\boldsymbol{z}}{\boldsymbol{w}} \quad \frac{\boldsymbol{w}}{\boldsymbol{w}}\right]$. Jika elemen homogen bernilai satu maka diperoleh matriks $[\boldsymbol{x} \quad \boldsymbol{y} \quad \boldsymbol{z} \quad \mathbf{1}]$ yang berupa informasi koordinat 3 dimensi.

2.4.4 Transformasi Geometri

Ada beberapa elemen transformasi geometri dalam pemrograman grafis yaitu :

a. Rotasi

Transformasi geometri rotasi dapat dinyatakan dengan bentuk perkalian matriks. Berikut merupakan contoh perkalian matriks untuk melakukan transformasi rotasi posisi :

Rotasi dengan sumbu x sebagai titik putar

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta & sin\theta & 0 \\ 0 & -sin\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotasi dengan sumbu y sebagai titik putar

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] \begin{bmatrix} cos\theta & 0 & -sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ sin\theta & 0 & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotasi dengan sumbu z sebagai titik putar

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] \begin{bmatrix} cos\theta & sin\theta & 0 & 0 \\ -sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

b. Translasi

Transformasi geometri translasi dapat dinyatakan dengan bentuk perkalian matriks dengan penambahan elemen keempat TX, TY, dan TZ

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ TX & TY & TZ & 1 \end{bmatrix}$$

c. Skalasi

Transformasi geometri skalasi dapat dinyatakan dengan bentuk perkalian matriks dengan faktor perbesaran s untuk masing-masing sumbu koordinat.

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

d. Kemiringan *(shearing)*

Transformasi geometri *shearing* dapat dinyatakan dengan bentuk perkalian matriks dengan notasi $\boldsymbol{h_{n1n2}}$ dimana semisal $\boldsymbol{h_{yx}}$ menunjukkan nilai $\boldsymbol{y}$ mempengaruhi nilai $\boldsymbol{x'}$.

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] \begin{bmatrix} 1 & h_{xy} & h_{xz} & 0 \\ h_{yx} & 1 & 0 & 0 \\ h_{zx} & h_{zx} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.4.5 Konkatenasi Matriks (Matrix Concatenation)

Konkatenasi matriks adalah rententan perkalian antar matriks-matriks yang berdimensi sama. Apabila matriks tidak berdimensi sama maka perlu dilakukan penyeragaman atau penambahan elemen homogen terlebih dahulu.

$$[x' \quad y' \quad z' \quad 1]$$

$$= [x \quad y \quad z \quad 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta & sin\theta & 0 \\ 0 & -sin\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ TX & TY & TZ & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

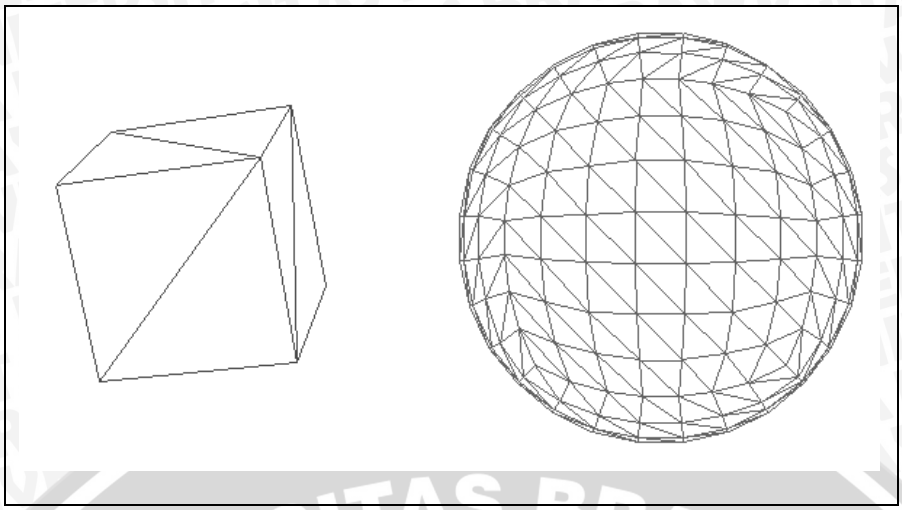
2.5 Geometri dalam Pemrograman 3D

2.5.1 Verteks

Sebuah titik dalam pemrograman grafis disebut dengan istilah verteks. Verteks dapat diberi atribut lain selain koordinat posisi seperti warna, koordinat tekstur dan normal vektor.

2.5.2 Geometri Triangular

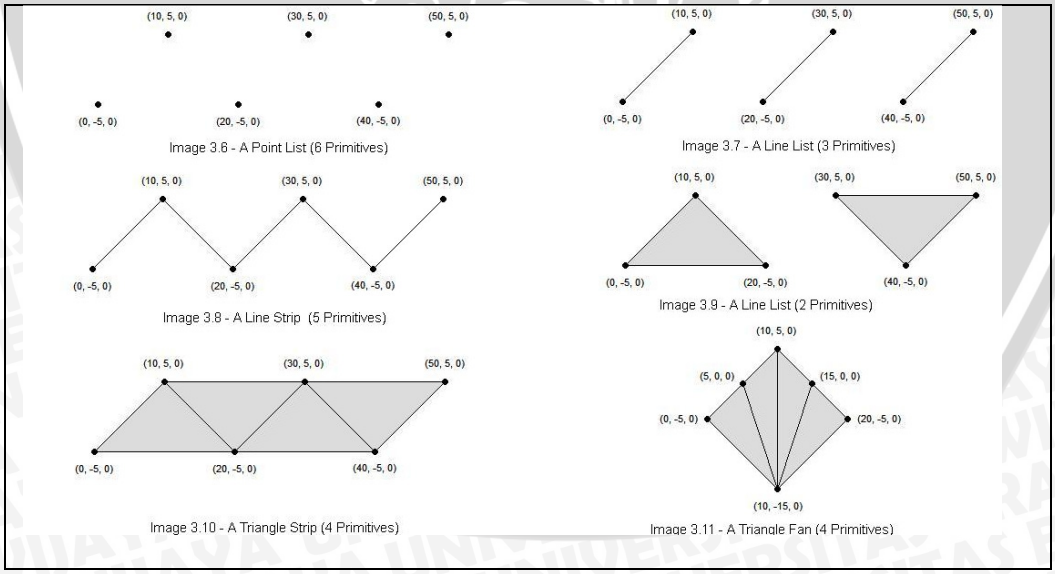
Segitiga merupakan bangun dasar yang sangat berguna dalam pemrograman 3D. Segitiga dapat menyusun bangun 3D hingga ke bentuk yang sangat kompleks sekalipun. Syarat minimal agar kumpulan verteks mempunyai permukaan adalah apabila verteks-verteks tersebut minimal membentuk sebuah segitiga. Keistimewaan-keistimewaan dari segitiga inilah yang membuat DirectX Graphics bekerja dalam lingkungan segitiga.



Gambar 2.3 Geometri triangular

2.5.3 Primitif

Primitif adalah elemen bangun yang paling sederhana dalam pemrograman grafis 3D. primitif dapat berupa titik, garis, dan segitiga. Dalam pemrograman grafis terutama DirectX Graphics, penggambaran secara primitif dapat dinyatakan dengan menyatakan daftar titik*(point lists),* daftar garis*(line lists),* daftar garis yang saling berhubungan*(line strips),* daftar segitiga*(triangle lists),* daftar segitiga yang saling berhubungan(triangle strips), dan daftar segitiga yang berbagi titik atau verteks*(triangle fans).*



Gambar 2.4 Geometri primitif

2.5.4 Indeks

Pembentukan bangun 3D dengan mendefinisikan banyak verteks kemudian menggambarnya secara primitif masih mempunyai kekurangan.

Apabila objek yang digambar berupa poligon quad dan penggambaranya dilakukan dengan menggunakan daftar segitiga atau triangle list maka terdapat dua verteks yang didefinisikan secara berulang. Hal ini kurang efisien terlebih lagi apabila objek yang dibentuk merupakan objek yang kompleks. Solusi dari permasalahan ini adalah pendefinisian dengan menggunakan indeks. Indeks tidak hanya membantu dalam hal pemodelan tetapi juga dapat menghemat array.

2.5.5 *Winding Order*

Winding order adalah aturan yang mengaruskan agar sebuah model didefinisikan dengan urutan verteks atau indeks dengan cara berurutan searah jarum jam. Hal ini dimaksudkan agar permukaan bagian belakang permukaan sebuah objek tidak dilukis atau di-*render* pada saat aplikasi berjalan. Proses pembuangan bagian belakang ini disebut dengan istilah *backface culling.*

2.5.6 Mesh

Mesh adalah koleksi verteks yang membentuk sebuah objek solid. Istilah mesh dalam lingkungan pemrograman grafis secara umum diidentikkan dengan sebuah objek atau model 3D.

2.6 File .X

File .x adalah format file berisikan struktur data yang digunakan DirectX untuk menyimpan dan memuat geometri objek 3D. file .x sangat berguna untuk menstorasi informasi geometri dalam skala besar. Programmer grafis dapat mengenerasi daftar verteks dan indeks menggunakan *software modelling* seperti 3DSMAX ke dalam dalam bentuk file .x untuk selanjutnya digunakan dalam pemrograman yang menggunakan DirectX Graphics.

2.7 Ruang-ruang Koordinat pada Pemrograman Grafis 3D

Ruang-ruang koordinat pada pemrograman grafis 3D adalah ruang-ruang koordinat yang menentukan informasi lokasi verteks atau titik pembentuk objek 3D. Ruang-ruang koordinat pada 3D bisa berupa koordinat kartesian, koordinat unit atau koordinat ternormalisasi dan sekaligus koordinat 2 dimensi.

a. Ruang lokal *(local space)*

Ruang lokal atau *local space* adalah ruang koordinat kartesian 3D dimana biasanya sebuah objek 3D dimodelkan. Objek 3D biasanya didefinisikan pada koordinat terpisah karena pemodelan lebih mudah dilakukan jika objek 3D terletak di titik pusat koordinat .

b. Ruang *world (world space)*

world space adalah ruang koordinat kartesian 3D digunakan bersama oleh objek-objek 3D hasil pemodelan yang mempunyai masing-masing ruang lokal. *world space* digunakan untuk mendefinisikan hubungan spasial antara objek-objek yang nantinya akan dilukis atau di-*render* pada layar media penampil.

c. Ruang pandang *(view space)*

Ruang pandang atau *view space* adalah ruang koordinat yang dapat dianggap sebagai ruang lokal atau koordinat lokal milik kamera virtual. Ruang pandang digunakan untuk menentukan orientasi sudut pandang *(point of view)* objek-objek 3D yang berada dalam *world space*.

d. Ruang proyeksi *(projection space)*

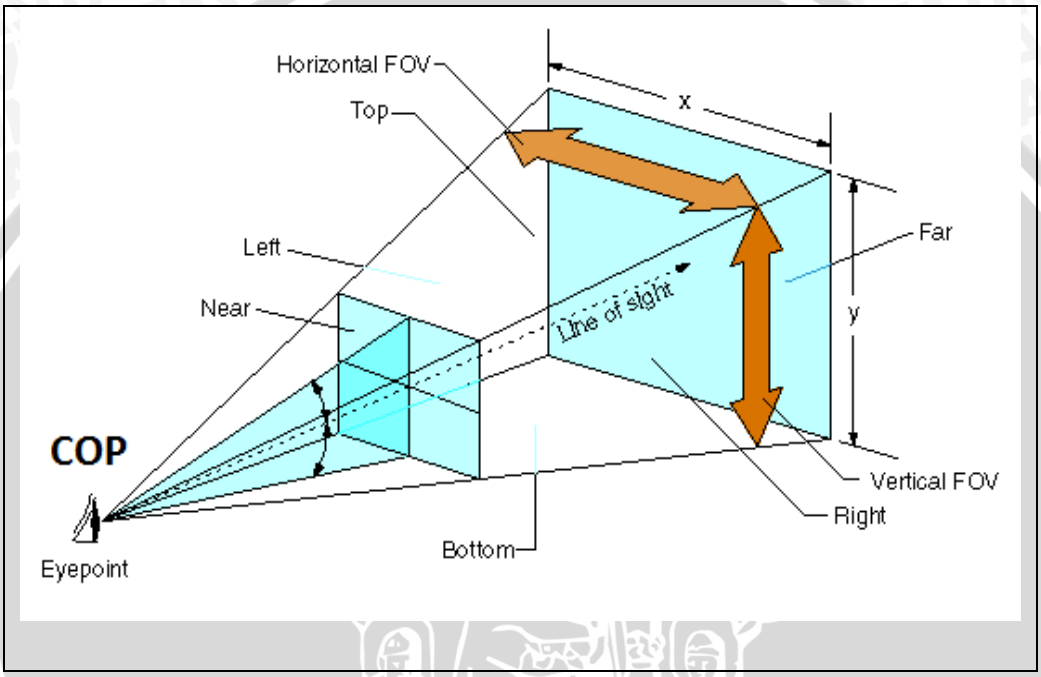
Ruang proyeksi atau *projection space* adalah ruang koordinat yang menentukan dimana lokasi-lokasi verteks milik objek 3D berada setelah dikenai transformasi proyeksi dari ruang pandang. Ruang proyeksi merupakan bidang datar 2 dimensi dengan skala ternormalisasi yaitu nilai posisi verteks x dan y berada dalam range $(-1,-1)$ hingga $(1,1)$. Pada ruang proyeksi ditambahkan elemen ketiga yaitu elemen z yang berkisar antara $0-1$ sebagai penentu prioritas kedalaman objek-objek 3D yang dilukis secara bersamaan.

e. Ruang layar *(screen space)*

Ruang layar atau screen space adalah ruang koordinat yang berisikan piksel-piksel milik layar media penampil. Nilai verteks pada ruang layar adalah hasil akhir dari transformasi objek 3D sebelum pada akhirnya DirectX Graphics melakukan komputasi untuk melakukan interpolasi atau penyisipan data-data pixel yang terletak di dalam poligon.

2.8 Proyeksi Perspektif

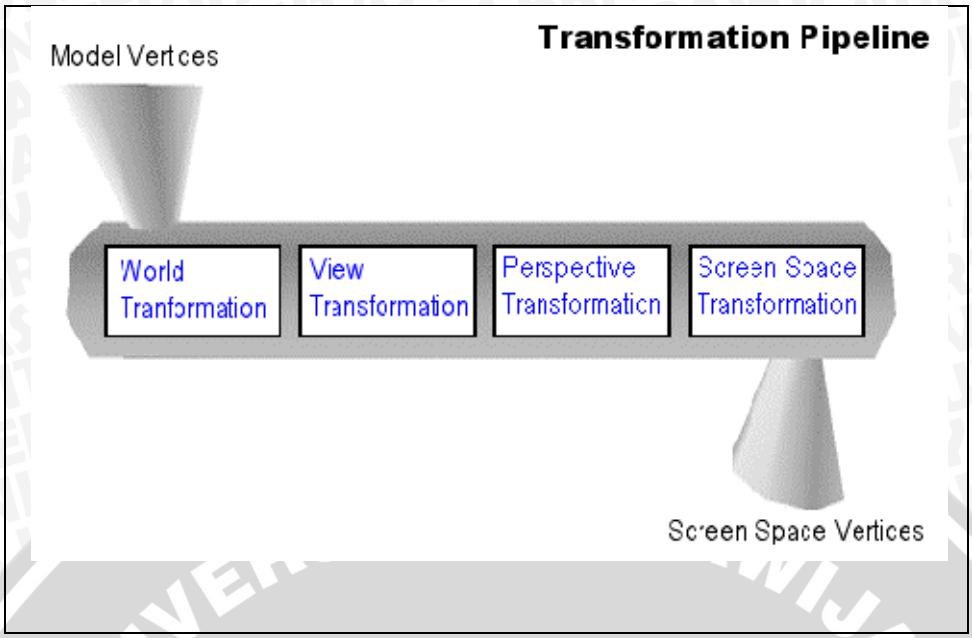
Proyeksi perspektif dapat didefinisikan sebagai penarikan garis-garis cahaya dari sebuah objek 3D menuju ke suatu titik pusat proyeksi COP*(Center of Projection)* dan melewati sebuah bidang datar 2D yang disebut sebagai bidang proyeksi. Gambar objek yang terbentuk pada bidang datar 2D yang dilewati oleh penarikan garis-garis cahaya tersebut merupakan hasil dari proyeksi.



Gambar 2.5 Ilustrasi dari proyeksi perspektif

2.9 *Transformation Pipeline*

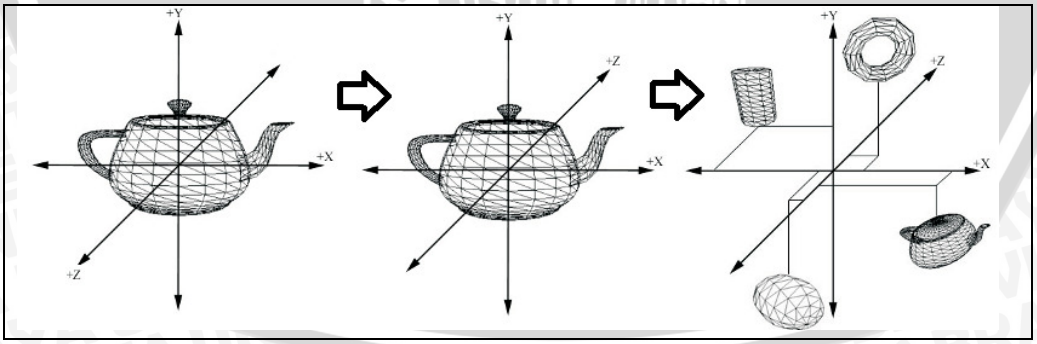
Transformation pipeline adalah tahapan-tahapan yang dilalui sebuah verteks penyusun objek 3D yang hanya mempunyai nilai dalam sistem koordinat 3D, supaya nantinya mendapatkan nilai di dalam bentuk sistem koordinat 2D pada layar media penampil. Ada empat tahapan penting dalam transformation pipeline yaitu transformasi ruang *(world transformation)*, transformasi sudut pandang *(view transformation)*, transformasi proyeksi *(projection transformation),* dan transformasi layar *(screen space transformation).*



Gambar 2.6 *Transformation pipeline*

2.9.1 Transformasi Ruang *(World Transformation)*

Transformasi ruang atau *world transformation* adalah transformasi yang bertujuan untuk merubah nilai-nilai verteks penyusun sebuah objek 3D yang berada dalam koordinat lokal menjadi mempunyai nilai-nilai di dalam koordinat bersama sebelum dapat dilakukan transformasi ruang lain seperti translasi, rotasi, dan skalasi bersama dengan objek-objek lain dan mendapatkan hasil sesuai yang diharapkan.



Gambar 2.7 Ilustrasi dari transformasi ruang *(world transformation)*

Transformasi ruang di dalam pemrograman grafis dapat dilakukan dengan mengalikan input posisi verteks di dalam koordinat lokal objek dengan matriks pengali berdimensi 4×4 yang disebut dengan matriks ruang *(world matrix).* Matriks ruang terdiri dari unit-unit vektor atau arah bidang datar *(plane)* sumbu-

sumbu x, y, dan z milik koordinat lokal yang diposisikan terhadap koordinat ruang beserta posisi verteks di dalam koordinat lokal.

$$[verteks\ world] = [verteks\ lokal] \begin{bmatrix} RightLokal.x & RightLokal.y & RightLokal.z & 0 \\ UpLokal.x & UpLokal.y & UpLokal.z & 0 \\ LookLokal.x & LookLokal.y & LookLokal.z & 0 \\ PosisiLokal.x & PosisiLokal.y & PosisiLokal.z & 1 \end{bmatrix}$$

Ketiga baris pertama pada matriks ruang atau *world matrix* di atas menyatakan nilai unit vektor tiap-tiap *plane* atau bidang datar milik koordinat lokal objek pada koordinat ruang sedangkan baris terakhir menyatakan posisi verteks pada koordinat lokal. Hasil perkalian posisi verteks pada koordinat lokal objek dengan matriks ruang menghasilkan posisi baru di dalam koordinat ruang.

2.9.2 Transformasi Pandang (View Transformation)

Transformasi pandang atau *view transformation* adalah transformasi yang bertujuan untuk merubah nilai-nilai verteks penyusun sebuah objek 3D yang berada dalam koordinat ruang menjadi mempunyai nilai-nilai di dalam koordinat pandang atau *view space*. Koordinat pandang atau *view space* bisa dianalogikan seperti koordinat kamera virtual karena koordinat pandang bersifat menentukan sudut pandang atau *point of view* relatif terhadap objek-objek yang berada di dalam koordinat ruang.

Pada transformasi ruang, pengubahan koordinat lokal objek ke dalam koordinat ruang didapatkan dari perkalian posisi verteks dengan koordinat lokal objek dengan matriks ruang. Maka untuk mendapatkan kembali posisi verteks dalam koordinat lokal objek didapatkan dengan inverse matriks dari matriks ruang. Karena koordinat lokal baru bisa dianalogikan sebagai koordinat lokal kamera maka pencarian matriks dilakukan dengan melakukan invers dari matriks ruang milik koordinat lokal kamera.

$$V_{ruang} = V_{lokal}\, M_{ruang}$$

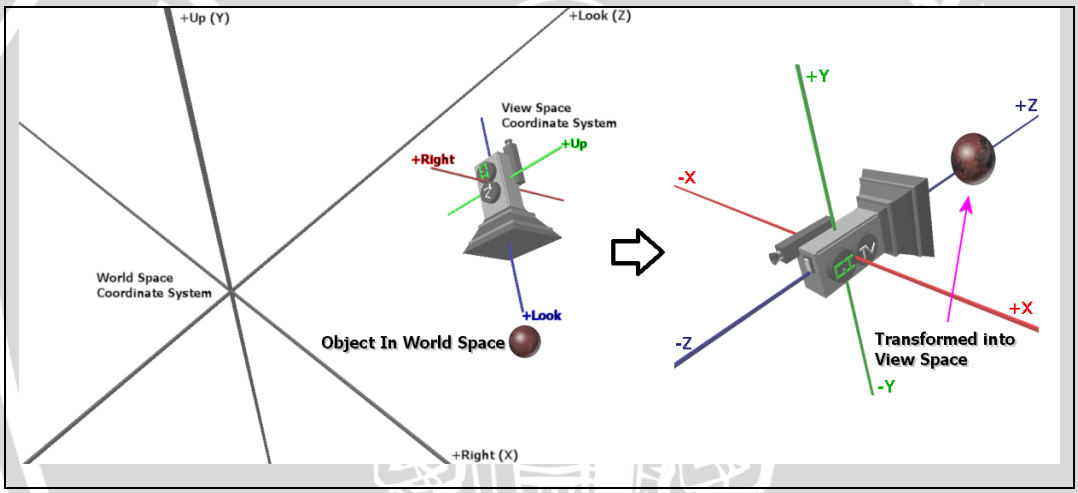
$$V_{ruang} \frac{1}{M_{ruang}} = V_{lokal\ kamera} \left(M_{ruang} \left(\frac{1}{M_{ruang}} \right) \right)$$

$$V_{lokal\ kamera} = V_{ruang}\, {M_{ruang}}^{-1}$$

$$M_{ruang}{}^{-1} = \frac{\begin{bmatrix} KameraRight.x & KameraRight.y & KameraRight.z & 0 \\ KameraUp.x & KameraUp.y & KameraUp.z & 0 \\ KameraLook.x & KameraLook.y & KameraLook.z & 0 \\ PosisiKamera.x & PosisiKamera.y & PosisiKamera.z & 1 \end{bmatrix}}{|M_{ruang}|}$$

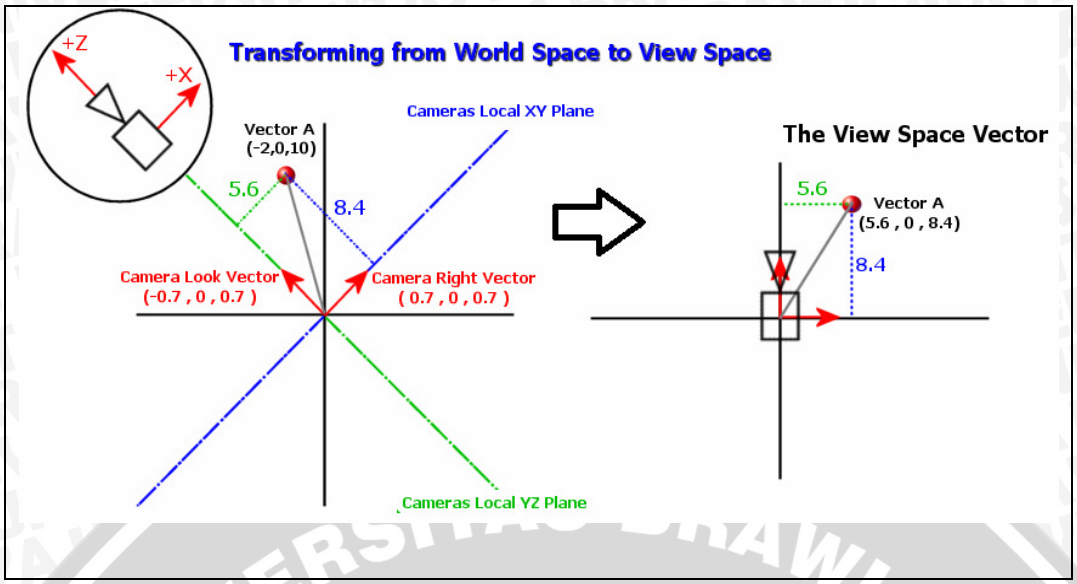
$$=$$

$$\begin{bmatrix} KameraRight.x & KameraUp.x & KameraLook.x & 0 \\ KameraRight.y & KameraUp.y & KameraLook.y & 0 \\ KameraRight.z & KameraUp.z & KameraLook.z & 0 \\ -(\overrightarrow{\quad}_{PosisiKamera}.\overrightarrow{\quad}_{KameraRight}) & -(\overrightarrow{\quad}_{PosisiKamera}.\overrightarrow{\quad}_{KameraUp}) & -(\overrightarrow{\quad}_{PosisiKamera}.\overrightarrow{\quad}_{KameraLook}) & 1 \end{bmatrix}$$



Gambar 2.8 Ilustrasi dari transformasi pandang *(view space)*

Konsep transformasi pandang atau *view transformation* sebenarnya lebih mudah dipahami dengan menggunakan unit vektor bidang-bidang datar *(plane)* milik koordinat lokal kamera virtual. Jarak terhadap plane dapat dicari dengan menggunakan dot produt dengan unit vektor plane atau arah permukaan bidang datar terhadap verteks pada koordinat ruang sehingga nantinya akan didapatkan koordinat verteks baru dengan nilai yang bersesuaian dengan koordinat lokal kamera.



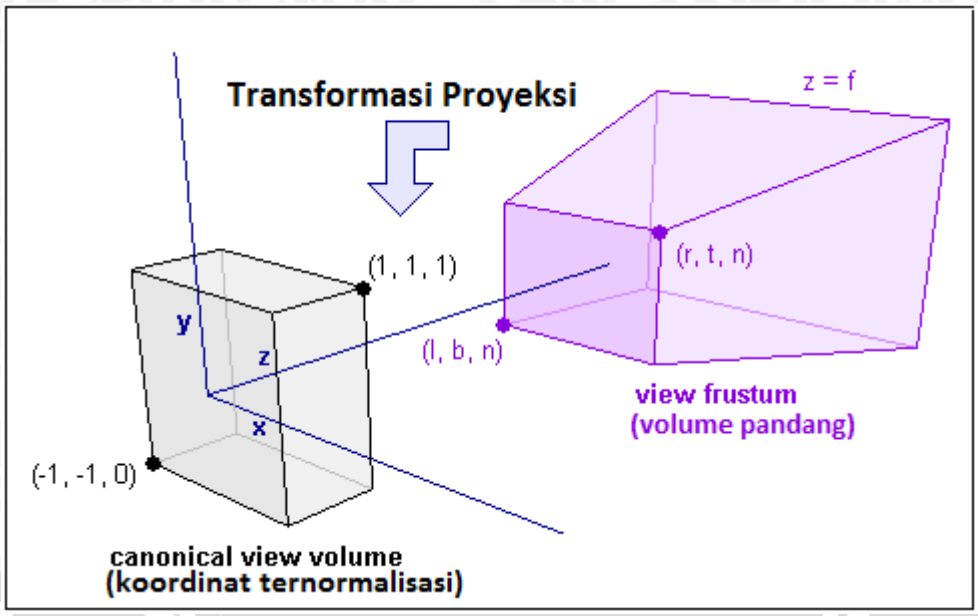
Gambar 2.9 *View space* yang didapat dari normal *plane* kamera

2.9.3 Transformasi Proyeksi (Projection Transformation)

Transformasi proyeksi atau *projection transformation* adalah transformasi yang bertujuan untuk merubah nilai-nilai verteks penyusun sebuah objek 3D yang berada dalam koordinat lokal kamera atau *view space* menjadi koordinat 2D yang disebut koordinat proyeksi atau ruang proyeksi. Meskipun bidang proyeksi hanya berdimensi dua, nilai z masih diperlukan untuk menentukan nilai kedalaman *(depth)* sehingga tidak terjadi anomali dalam hal pelukisan *(render).*

Objek 3D sebelum diproyeksikan berada dalam sebuah volume pandang *(view volume).* Volume pandang adalah ruang berbentuk *frustum* yaitu potongan limas atau piramid yang membatasi objek 3D yang akan diproyeksikan. Volume pandang berada diantara dua bidang datar yang disebut *near plane* dan *far plane.* Karena volume pandang semakin jauh semakin besar dan bidang proyeksi sifatnya konstan maka apabila sebuah objek 3D berada semakin dekat dengan far plane maka sebuah objek 3D akan tampak semakin kecil sehingga memberi kesan ilusi jauh dekat. Pada pemrograman grafis 3D, bidang proyeksi dinyatakan sebagai koordinat ternormalisasi atau volume pandang kanonikal.

Formulasi transformasi proyeksi didapatkan dengan mencari nilai x', y', z' pada koordinat ternormalisasi yang berasal dari nilai $x, y,$ dan z pada volume pandang.



Gambar 2.10 Volume pandang dan koordinat ternormalisasi

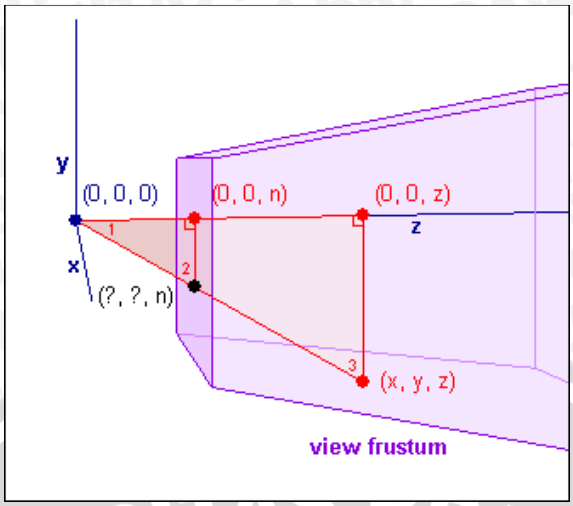
Dari gambar di atas terlihat bahwa Koordinat ternormalisasi memiliki range antara $(-1,-1,0)$ hingga $(1,1,0)$. *Near plane* pada volume pandang berkisar antara $(left, bottom, near)$ hingga $(right, top, near)$ sedangkan untuk *far plane* keempat titiknya berpotongan dengan bidang datar $z = f$. Dengan mengubah nilai $l \leq x \leq r$ dan $b \leq y \leq t$ menjadi $-1 \leq x' \leq 1$ dan $-1 \leq y' \leq 1$, sedangkan untuk nilai $n \leq z \leq f$ diubah menjadi $0 \leq z \leq 1$, maka didapatkan persamaan-persamaan

$$x' = \frac{2x}{r-l} - \frac{r+l}{r-l}$$

$$y' = \frac{2y}{t-b} - \frac{t+b}{t-b}$$

$$z' = \frac{z}{f-n} - \frac{n}{f-n}$$

Karena proyeksi perspektif dipengaruhi oleh nilai z, maka perlu dicari pengaruh nilai z terhadap nilai x' dan y'.



Gambar 2.11 Dua titik pada segitiga sebangun

Pada gambar diatas titik dengan nilai z berupa n dan z berada pada dua segitiga sebangun, sehingga didapatkan rasio perbandingan nilai x' dan y' pada *near plane* dengan titik yang ada dalam volume pandang sebesar $^{n}/_{z}$ sehingga didapatkan persamaan baru yaitu

$$x'z = \left(\frac{2n}{r-l}\right)x - \frac{r+l}{r-l}z$$

$$y'z = \left(\frac{2n}{t-b}\right)y - \frac{t+b}{t-b}z$$

Karena z hanya berfungsi untuk penentu kedalaman dan tidak terpengaruh oleh nilai x dan y maka untuk mencari persamaan $z'z$ digunakan persamaan garis. Diketahui apabila nilai z berada pada range (n, f), maka setelah ditransformasi ke dalam koordinat ternormalisasi akan berada pada range (0,1), sehingga diketahui nilai $z' = 0$ apabila $z = n$, dan $z' = 1$ apabila $z = f$. Maka untuk mendapatkan persamaan garis $z'z = pz + q$ dilakukan subtitusi sebagai berikut

$$0 = pn + q$$

$$f = pf + q$$

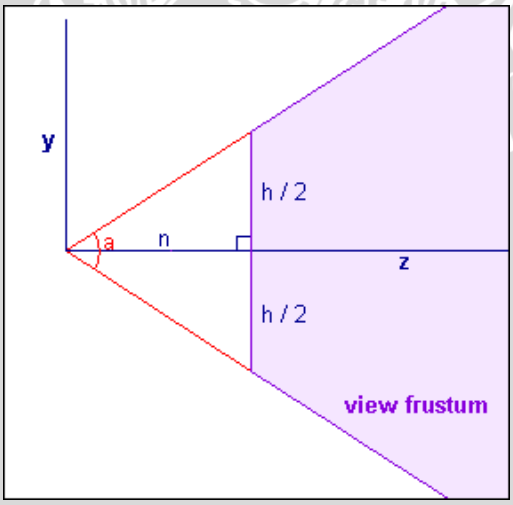
Sehingga didapatkan

$$z'z = \frac{f}{f-n}z - \frac{fn}{f-n}$$

Dengan koordinat homogen yang biasanya bernilai $w' = 1$, apabila persamaan mempunyai faktor pengali z, maka agar nantinya bisa dilakukan *homogenizing* digunakan nilai $w'z = z$. Sehingga didapatkan matriks proyeksi berupa

$$M_{proyeksi} = \begin{bmatrix} \frac{2n}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2n}{t-b} & 0 & 0 \\ -\frac{r+l}{r-l} & -\frac{t+b}{t-b} & \frac{f}{f-n} & 1 \\ 0 & 0 & -\frac{fn}{f-n} & 0 \end{bmatrix}$$

Dalam DirectX Graphics koordinat ternormalisasi berada di sumbu pusat $(0,0,0)$, sehingga didapatkan $r = -l$, $t = -b$, dengan $r - l$ merupakan lebar w dan $t - b$ merupakan tinggi h. Dengan penentuan nilai-nilai tersebut maka persamaan matriks proyeksi di atas dapat disederhanakan.



Gambar 2.12 Medan pandang atau *field of view*

Dari gambar diatas terdapat sudut terhadap ruang pandang yang biasa disebut sudut medan pandang atau sudut *field of view*. Dengan persamaan trigonometri dapat diperoleh:

$$cot\frac{a}{2} = \frac{2n}{h}$$

Perbandingan antara lebar dan panjang disebut rasio aspek atau *aspect ratio*. Aspect ratio bisa dinyatakan dengan perbandingan antara lebar dan tinggi

$$Rasio\ Aspek = \frac{w}{h}$$

$$\frac{2n}{w} = \frac{2n}{rh} = \frac{1}{Rasio\ Aspek} cot\frac{a}{2}$$

Hasil akhir dari penurunan dan penyederhanaan persamaan yang telah disebutkan, didapatkan matriks proyeksi yang dapat diberi masukan berupa medan pandang atau *field of view*, batas proyeksi *near plane* dan *far plane* serta rasio aspek media penampil.

$$M_{proyeksi} = \begin{bmatrix} \frac{1}{Rasio\ Aspek} cot\frac{a}{2} & 0 & 0 & 0 \\ 0 & cot\frac{a}{2} & 0 & 0 \\ 0 & 0 & \frac{f}{f-n} & 1 \\ 0 & 0 & -\frac{fn}{f-n} & 0 \end{bmatrix}$$

2.9.4 Transformasi Layar *(Screen Space Transformation)*

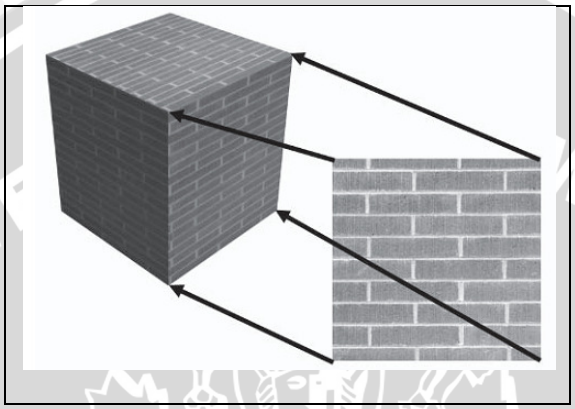
Transformasi layar atau *screen space transformation* adalah transformasi yang bertujuan untuk merubah nilai koordinat objek 3D yang berada dalam koordinat ternormalisasi menjadi nilai koordinat pada layar media penampil. Transformasi layar memerlukan pengkonversian dari koordinat ternomalisasi dengan range antara -1 hingga 1 menjadi koresponden dengan lebar dan tinggi layar media penampil semisal layar monitor. Transformasi layar bisa didapatkan dengan persamaan

$$posisi\ layar\ x = \frac{hasil\ proyeksi\ x \times lebar\ layar}{2} + \frac{lebar\ layar}{2}$$

$$posisi\ layar\ y = \frac{hasil\ proyeksi\ y \times lebar\ layar}{2} + \frac{tingggi\ layar}{2}$$

2.10 Pemetaan Tekstur *(Texture Mapping)*

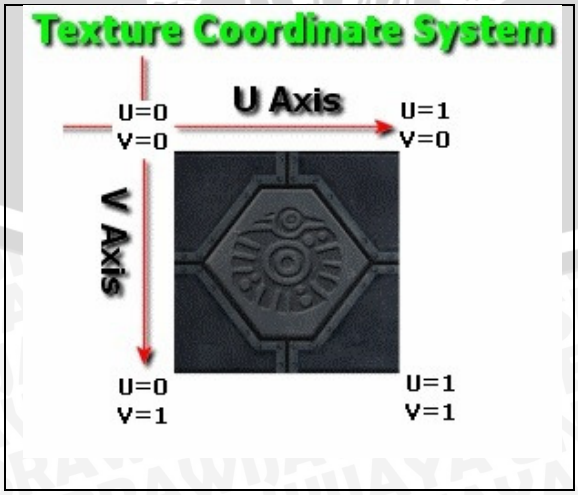
Pemetaan tekstur adalah pemberian gambar dua dimensi kepada verteks-verteks atau titik-titik yang membentuk poligon pada sistem koordinat 3D. Tekstur merupakan gambar bitmap atau gambar raster yang nantinya akan dipetakan atau disisipkan kepada objek 3D. Pemetaan tekstur bisa diterapkan pada objek 3D karena pada umumnya objek 3D tersusun dari banyak poligon.



Gambar 2.13 Contoh dari pemetaan tekstur

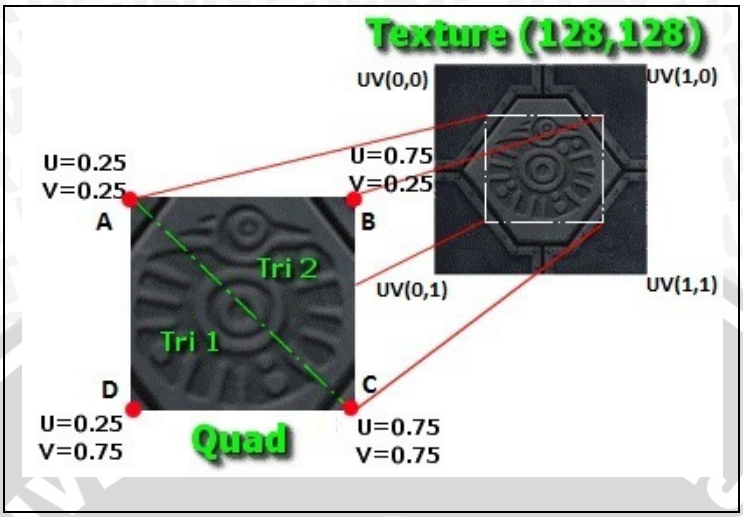
2.11 Koordinat Tekstur

Koordinat tekstur adalah koordinat yang digunakan sebuah poligon sebagai acuan mengenai bagian atau porsi gambar yang akan dipetakan atau disisipkan pada poligon tersebut. Koordinat tekstur dinyatakan dengan skala unit antara (0,0) sampai (0,1). Dengan sumbu u yang menyatakan porsi horisontal sebuah gambar dan sumbu v yang menyatakan porsi vertikal.



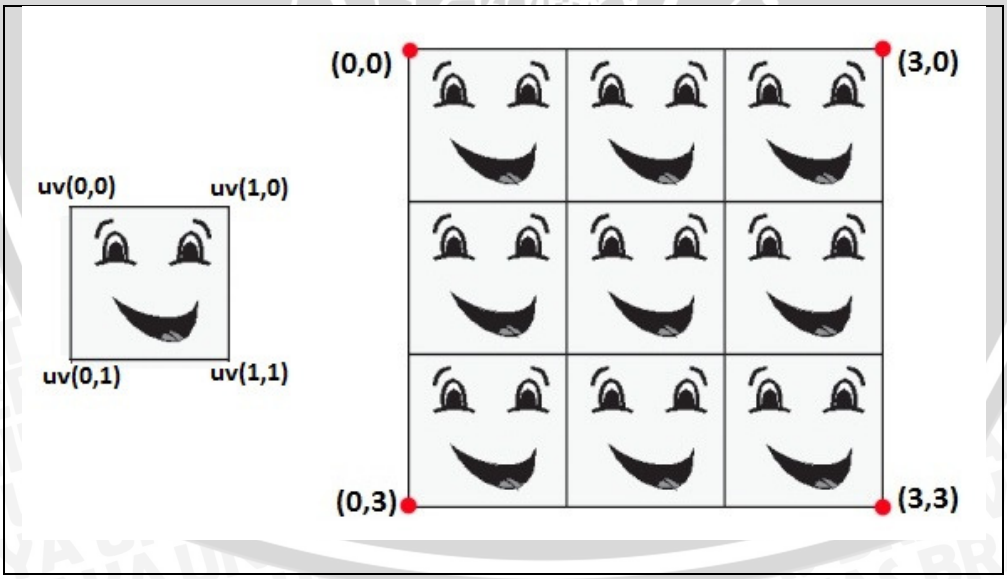
Gambar 2.14 Sistem koordinat tekstur

Apabila sebuah poligon mempunyai nilai koordinat tekstur yaitu $0 < u < 1$ dan u $0 < v < 1$ maka tekstur yang akan dipetakan hanya sebagian saja.



Gambar 2.15 Hasil dari pemberian koordinat tekstur terhadap objek polygon

Apabila sebuah poligon mempunyai nilai koordinat tekstur yang melebihi range (0,0) dan (1,1) maka gambar akan ditampilkan secara repetitif atau berulang.



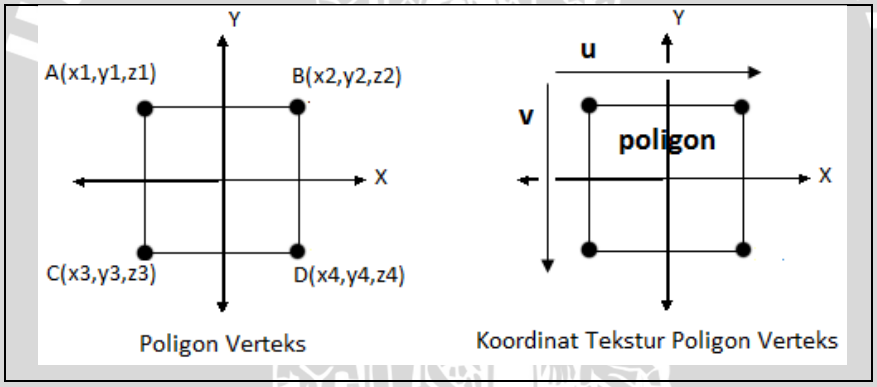
Gambar 2.16 Poligon dengan nilai koordinat tekstur melebihi skala unit

2.12 Metode Pengalamatan Tekstur

Metode pengalamatan tekstur adalah cara pemberian atau cara pendistribusian nilai-nilai koordinat tekstur pada verteks-verteks penyusun sebuah objek 3D. Metode pengalamatan tekstur sangat beragam dan fleksibel tergantung kebutuhan pemodel. Ada tiga metode pengalamatan yang umum dijumpai saat ini yaitu metode pengalamatan planar, metode pengalamatan spheris, dan metode pengalamatan silindris.

2.12.1 Metode Pengalamatan Planar

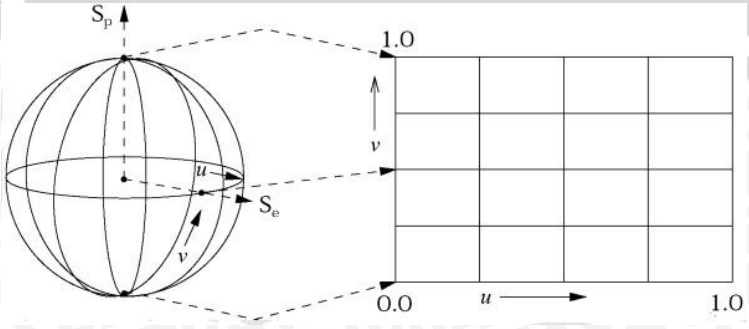
Metode pengalamatan planar adalah cara pendistribusian nilai-nilai koordinat tekstur dimana nilai-nilai koordinat tekstur u dan v diberikan secara linier atau sebanding dengan sumbu x dan y pada verteks-verteks poligon.



Gambar 2.17 Metode Pengalamatan Planar

2.12.2 Metode Pengalamatan Spheris

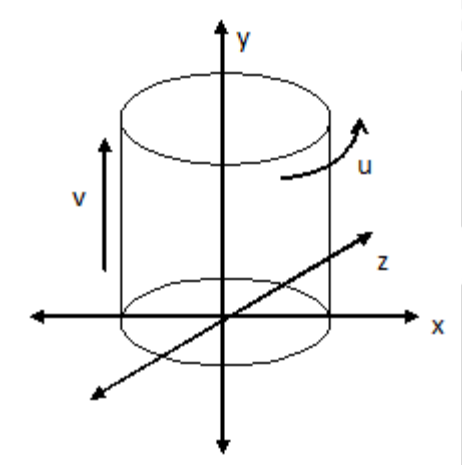
Metode pengalamatan spheris adalah cara pendistribusian nilai-nilai koordinat tekstur dimana nilai-nilai koordinat tekstur u dan v diberikan secara sebanding dengan sudut phi dan theta milik objek yang berbentuk spheris.



Gambar 2.18 Metode Pengalamatan Spheris

2.12.3 Metode Pengalamatan Silindris

Metode pengalamatan silindris adalah cara pendistribusian nilai-nilai koordinat tekstur dimana nilai-nilai koordinat tekstur u diberikan secara sebanding dengan sudut phi milik objek berbentuk silindris sedangkan nilai-nilai koordinat v diberikan secara linier sebanding dengan tinggi objek yang berbentuk silindris.



Gambar 2.19 Metode Pengalamatan Silindris

2.13 3DSMAX

3DSMAX merupakan sebuah perangkat lunak pemodel grafis 3D *(3D software modelling)* yang dapat digunakan sebagai alat bantu dalam mendefinisikan posisi sekaligus urutan verteks dan indeks dalam memodelkan atau membentuk sebuah objek 3D. Pemodelan objek 3D untuk objek yang kompleks menjadi lebih mudah karena 3DSMAX menyediakan antarmuka berupa tampilan verteks secara visual. Informasi mengenai posisi sekaligus urutan daftar verteks dan indeks kemudian dapat digenerasi dan disimpan ke dalam format file DirectX yaitu format file .x berupa metadata yang bersifat struktural maupun deskriptif sehingga nantinya dapat diakses dan digunakan dalam program aplikasi yang menggunakan implementasi DirectX.

BAB III
METODOLOGI PENELITIAN

Bab ini menjelaskan mengenai langkah-langkah yang dilakukan untuk membuat aplikasi. Langkah-langkah yang diperlukan antara lain studi literatur, perancangan, implementasi, pengujian dan analisis, serta pengambilan kesimpulan dan saran.

3.1 Studi Literatur

Berupa kajian pustaka yang mendukung pembuatan aplikasi yang berupa :

1. Kajian pustaka mengenai COM dan DirectX.
2. Kajian pustaka mengenai dasar-dasar perhitungan 3D pada pemrograman grafis.
3. Kajian pustaka mengenai geometri dalam pemrograman 3D.
4. Kajian pustaka mengenai tahapan-tahapan transformasi*(transformation pipeline).*
5. Kajian pustaka mengenai pemetaan tekstur, koordinat tekstur, dan metode pengalamatan tekstur.

3.2 Perancangan

Secara garis besar perancangan meliputi :

3.2.1 Analisis Kebutuhan

Analisis kebutuhan terdiri dari:

1. Diagram Konteks

   Diagram konteks digunakan untuk mengetahui kebutuhan dasar yang utama dalam perancangan aplikasi serta relasi dengan entitas luar yaitu pengguna aplikasi.

2. Tabel Kebutuhan

   Tabel kebutuhan digunakan untuk mengetahi daftar kebutuhan yang diperlukan sebagai elemen dasar dalam pembuatan program aplikasi. Tabel kebutuhan difokuskan pada kebutuhan objek antarmuka COM

DirectX sebagai antarmuka antara pemrograman Win32 dengan HAL*(Hardware Abstraction Layer).*

3. COM Diagram

   COM diagram digunakan untuk mengetahui hirarki dan relasi antar sesama objek antarmuka COM. COM diagram digunakan untuk membantu menentukan prosedur atau urut-urutan inisialisasi objek COM DirectX.

3.2.2 Perancangan Sistem

Perancangan sistem terdiri dari :

1. Perancangan Kerangka Utama

   Perancangan kerangka utama bertujuan untuk memberikan gambaran mengenai aliran sekuensial sistem secara keseluruhan dari tahap awal dijalankannya aplikasi hingga akhir akhir aplikasi.

2. Perancangan Fungsi *Main Loop* dan Fungsi Penanganan Pesan

   Perancangan *main loop* dan penanganan pesan ditujukan supaya program bersifat reaktif dan memberikan keluaran yang sinkron dan responsif terhadap masukan.

3. Perancangan Fungsi Input

   Perancangan fungsi input bertujuan agar sistem dapat mengenali dan mendapatkan masukan atau *event* masukan yang diberikan oleh entitas luar atau *user*.

4. Perancangan Fungsi Logika

   Perancangan fungsi logika bertujuan untuk menentukan respon sistem terhadap pengaruh masukan. Fungsi logika bertugas menentukan objek 3D maupun 2D yang akan dirender oleh sistem. Selain itu fungsi logika juga menentukan nilai dari tahapan-tahapan transformasi.

5. Perancangan Fungsi Render

   Perancangan fungsi render bertujuan untuk menentukan perintah render terhadap objek yang akan dilukis pada layar media penampil

3.2.3 Pemodelan Objek Primitif

Pemodelan objek primitif bertujuan untuk merancang dan membuat objek-objek berbentuk bangun relatif sederhana yang dikategorikan mempunyai sifat planar, spheris, dan silindris. Pembuatan objek primitif ditujukan untuk menerapkan metode-metode pengalamatan yang akan digunakan dalam menentukan alamat tekstur pada objek-objek dengan sifat planar, spheris, dan silindris sebelum menerapkan metode-metode pengalamatan tersebut pada objek kompleks berupa pakaian. Penggunaan objek primitif sebagai media awal pemetaan tekstur bertujuan untuk menentukan keberhasilan metode-metode pengalamatan tekstur.

3.2.4 Perancangan Metode Pengalamatan Tekstur

Metode pengalamatan tekstur bertujuan untuk mendistribusikan atau memberikan nilai-nilai koordinat tekstur pada objek 3D yang akan dijadikan media pemetaan gambar tekstur. Ada tiga metode pengalamatan yang akan dirancang dan digunakan dalam pembuatan program aplikasi yaitu :

a. Metode Pengalamatan Planar

Metode pengalamatan planar yaitu pendistribusian nilai-nilai koordinat tekstur dimana nilai-nilai koordinat tekstur u dan v diberikan secara linier atau sebanding dengan sumbu x dan y. Metode pengalamatan planar bertujuan untuk memberikan metode pengalamatan pada objek 3D yang mempunyai sifat planar atau mempunyai bentuk seperti bidang datar. Perhitungan yang digunakan untuk menentukan nilai-nilai koordinat tekstur dengan pengalamatan secara planar didasarkan atas persamaan

$$koordinat\ tekstur\ u = \frac{x}{x\ max\ -\ x\ min} + 0{,}5$$

$$koordinat\ tekstur\ v = -\frac{y}{y\ max\ -\ y\ min} + 0{,}5$$

b. Metode Pengalamatan Spheris

Metode pengalamatan spheris yaitu cara pendistribusian nilai-nilai koordinat tekstur dimana nilai-nilai koordinat tekstur u dan v diberikan secara sebanding dengan sudut phi dan theta milik objek yang berbentuk spheris. Metode pengalamatan spheris bertujuan untuk memberikan metode pengalamatan pada objek 3D yang mempunyai sifat lengkung dengan orientasi vertikal dan horisontal.

Perhitungan yang digunakan untuk menentukan nilai-nilai koordinat tekstur secara spheris dicari berdasarkan persamaan koordinat bola dan hasil sudut dot product.

Untuk koordinat tekstur vertikal atau koordinat tekstur v

$$koordinat\ tekstur\ v = \frac{arcos({}^{y}/_{r})}{\pi}$$

Untuk koordinat tekstur horisontal atau koordinat tekstur u

Jika $N_Z \geq 0$

$$koordinat\ tekstur\ u = \frac{\arccos\ (A \cdot \vec{i})}{2\pi}$$

Jika $N_Z < 0$

$$koordinat\ tekstur\ u = 1 - \frac{\arccos\ (A \cdot \vec{i})}{2\pi}$$

Perhitungan kedua yang digunakan untuk menentukan nilai-nilai koordinat tekstur secara spheris didapat berdasarkan normal vektor milik verteks terhadap posisi x dan y dengan orientasi arah dari titik pusat objek.

$$koordinat\ tekstur\ u = \frac{\text{asin}\ (N_x)}{\pi} + 0{,}5$$

$$koordinat\ tekstur\ v = -\frac{\text{asin}\ (N_y)}{\pi} + 0{,}5$$

c. Metode Pengalamatan Silindris

Metode pengalamatan silindris yaitu cara pendistribusian nilai-nilai koordinat tekstur dimana nilai-nilai koordinat tekstur u diberikan secara sebanding dengan sudut phi milik objek yang berbentuk silinder sedangkan koordinat tekstur v diberikan secara linier atau sebanding dengan tinggi silinder. Metode pengalamatan silindris bertujuan untuk memberikan metode pengalamatan pada objek 3D yang mempunyai sifat lengkung dengan orientasi horisontal dan mempunyai sifat planar dengan orientasi vertikal.

Perhitungan yang digunakan untuk menentukan nilai-nilai koordinat tekstur secara silindris dicari berdasarkan persamaan koordinat silinder yaitu :

Untuk koordinat tekstur vertikal atau koordinat tekstur v

$$koordinat\ tekstur\ v = -\frac{y}{y\ max\ -\ y\ min} + 0{,}5$$

Untuk koordinat tekstur horisontal atau koordinat tekstur u

Jika $N_Z \geq 0$

$$koordinat\ tekstur\ u = \frac{arcos\left(\frac{x}{r}\right)}{2\pi}$$

Jika $N_Z < 0$

$$koordinat\ tekstur\ u = 1 - \frac{arcos\left(\frac{x}{r}\right)}{2\pi}$$

Perhitungan kedua yang digunakan untuk menentukan nilai-nilai koordinat tekstur secara silindris didapat berdasarkan normal vektor milik verteks terhadap posisi x dengan orientasi arah dari titik pusat objek. Sedangkan untuk koordinat tekstur vertikal didapat secara linier berdasarkan pada tinggi silinder.

$$koordinat\ tekstur\ u = \frac{\text{asin}\ (N_x)}{\pi} + 0{,}5$$

$$koordinat\ tekstur\ v = -\frac{y}{y\ max\ -\ y\ min} + 0{,}5$$

3.2.5 Pemodelan Objek Pakaian

Pemodelan objek kompleks berupa pakaian dilakukan dengan menggunakan alat bantu berupa perangkat lunak pemodelan yaitu 3DSMAX. Dalam 3DSMAX pendefinisian daftar verteks dan indeks dilakukan dengan antarmuka visual sehingga memudahkan dalam hal pemodelan.

Hasil dari pemodelan secara visual menggunakan 3DSMAX kemudian dikonversi menjadi daftar verteks dan daftar indeks sesuai dengan *template* atau *metadata* struktural dan deskriptif milik *file* berekstensi .x dengan menggunakan program *exporter* yang ditambahkan pada 3DSMAX.

3.3 Implementasi

Penerapan hasil perancangan perangkat lunak yang telah dibuat ke dalam kode program *(coding)* sesuai dengan sintaks dari bahasa pemrograman yang digunakan yaitu menerapkan bahasa pemrograman C++ dan mengimplementasikan objek-objek COM milik DirectX.

3.4 Pengujian dan Analisis

Pengujian aplikasi yaitu untuk mengetahui kesesuaian analisis kebutuhan yang dibuat dengan implementasi aplikasi. Analisis sistem dilakukan dengan membandingkan sistem aplikasi yang dengan teori yang ada sehingga didapatkan suatu kesimpulan.

3.4.1 Pengujian Subjektif Hasil Proyeksi

Pengujian subjektif hasil proyeksi dilakukan dengan pengamatan dan pembandingan hasil proyeksi yang didapatkan dengan perkalian matriks dan hasil proyeksi yang berasal dari fungsi proyeksi dari DirectX Graphics.

3.4.2 Pengujian Objektif Hasil Proyeksi

Pengujian objektif hasil proyeksi dilakukan dengan perhitungan nilai ekstrim *(extreme value)* terhadap titik-titik yang akan diproyeksikan dengan hasil proyeksi pada bidang proyeksi berupa koordinat ternormalisasi.

3.4.3 Pengujian Subjektif Metode Pengalamatan

Pengujian subjektif terhadap metode pengalamatan dilakukan untuk mengamati hasil dari metode pengalamatan tekstur pada objek-objek primitif yang mempunyai sifat planar, spheris, dan silindris yaitu berupa kubus, bola, dan silinder. Pengujian subjektif kemudian dilakukan pada objek kompleks berbentuk pakaian. Pengujian secara subjektif dilakukan terhadap masing-masing metode pengalamatan yaitu metode pengalamatan planar, metode pengalamatan spheris, dan metode pengalamatan silindris.

3.4.4 Pengujian Objektif Metode Pengalamatan

Pengujian secara objektif terhadap metode pengalamatan dilakukan untuk mendapatkan nilai-nilai koordinat tekstur u dan v pada objek-objek primitif kubus, bola, dan silinder untuk tiap-tiap metode pengalamatan.

3.4.5 Pengujian Manipulasi Koordinat Tekstur

Pengujian manipulasi koordinat tekstur dilakukan untuk mengamati hasil dari pengubahan nilai-nilai koordinat tekstur pada objek-objek primitif kubus, bola, dan silinder kemudian pada objek kompleks berbentuk pakaian. Pengujian dilakukan terhadap masing-masing metode pengalamatan dengan mengunakan faktor skalasi yang sekaligus berperan sebagai representasi dari faktor translasi dan kemiringan *(shearing).*

3.5 Pengambilan Kesimpulan dan Saran

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan, implementasi, dan pengujian sistem aplikasi telah selesai dilakukan dan didasarkan pada kesesuaian antara teori dan praktek. Kesimpulan diambil untuk menjawab rumusan masalah yang telah ditetapkan sebelumnya. Tahap yang paling akhir adalah pengambilan saran terhadap penelitian yang selanjutnya,

sehingga bisa menyempurnakan kekurangan yang ada dan mengembangkan pada tingkat pokok kajian yang lebih lanjut.

BAB IV
PERANCANGAN

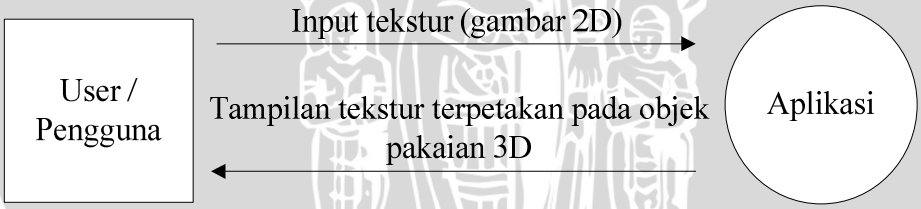
Bab ini membahas mengenai perancangan perangkat lunak. Perancangan yang dilakukan meliputi analisis kebutuhan, perancangan sistem, pemodelan objek 3D, metode pengalamatan tekstur, dan pemodelan objek pakaian.

4.1 Analisis Kebutuhan

Analisis kebutuhan merupakan tahapan awal dalam perancangan sistem. Analisis kebutuhan memberikan informasi mengenai kebutuhan minimalis yang dibutuhkan oleh sistem. Ada tiga hal yang diperlukan dalam analisis kebutuhan dalam perancangan sitem yaitu diagram konteks, tabel kebutuhan, dan diagram COM*(Component Objek Model).*

4.1.1 Diagram Konteks

Diagram konteks merupakan sebuah diagram sederhana yang menggambarkan hubungan dengan entitas luar, masukan dan keluaran dari sistem.



Gambar 4.1 Diagram Konteks

Dari diagram konteks dapat dilihat bahwa pelaku atau *user* memberikan masukan tekstur berupa gambar 2D (gambar bitmap atau gambar raster) kemudian aplikasi memberikan keluaran keluaran berupa tampilan tekstur terpetakan terhadap objek 3D yang berbentuk pakaian.

4.1.2 Tabel Kebutuhan

Tabel kebutuhan merupakan tabel yang digunakan untuk membuat daftar elemen yang dibutuhkan oleh sistem. Menurut analisis kebutuhan, sistem

memerlukan daftar kebutuhan mengenai objek-objek antarmuka COM yang akan digunakan oleh sistem. Terdapat dua *subset* atau bagian dari COM DirectX yang dibutuhkan oleh sistem. Bagian pertama adalah daftar COM yang menangani antarmuka perihal penanganan grafis yaitu DirectX Graphics. Sedangkan bagian yang kedua adalah daftar COM yang menangani penanganan input output yaitu DirectInput.

Tabel 4.1 Tabel Kebutuhan Objek Antarmuka DirectX Graphics yang Diperlukan

IDirect3D9	Objek COM Direct3D yang berfungsi sebagai antarmuka pemrograman untuk mendapatkan informasi tentang perangkat keras *display adapter* pada sistem sehingga dapat dibuat objek antarmuka IDirect3DDevice9 yang optimal sesuai dengan kapabilitas perangkat keras display adapter pada sistem komputer.
IDirect3DDevice9	Objek COM Direct3D yang berfungsi sebagai virtualisasi display adapter semisal kartu grafis dan berguna sebagai antarmuka pemrograman terhadap *display adapter.*
ID3DXSprite	Objek COM Direct3D yang berfungsi sebagai antarmuka pemrograman perihal penanganan grafis yang sifatnya permukaan datar dua dimensi seperti gambar bitmap.
ID3DXFont	Objek COM Direct3D yang berfungsi sebagai antarmuka pemrograman perihal pembuatan font (tulisan) pada *display adapter* dengan media rektangular.
IDirect3DVertexBuffer9	Objek COM Direct3D yang berfungsi sebagai antarmuka pemrograman untuk menyimpan data verteks pada memori display adapter (video RAM) ataupun memori sistem.
IDirect3DIndexBuffer9	Objek COM Direct3D yang berfungsi sebagai antarmuka pemrograman untuk menyimpan data indeks pada memori display adapter (video RAM) ataupun memori sistem.

IDirect3DTexture9	Objek COM Direct3D yang berfungsi sebagai antarmuka pemrograman untuk menyimpan data pixel milik gambar bitmap yang nantinya akan dipetakan pada verteks-verteks yang membentuk polygon.
ID3DXMesh	Objek COM Direct3D yang berfungsi sebagai antarmuka pemrograman untuk menyimpan data mesh atau model 3D yang terdiri dari daftar verteks, daftar indeks, dan daftar koordinat tekstur.
ID3DXBuffer	Objek COM Direct3D yang berfungsi sebagai antarmuka pemrograman sebagai buffer penyimpan sementara data mesh seperti menahan data atrbut warna milik kumpulan verteks yang membentuk mesh.

Tabel 4.2 Tabel Kebutuhan Objek Antarmuka DirectInput yang Diperlukan

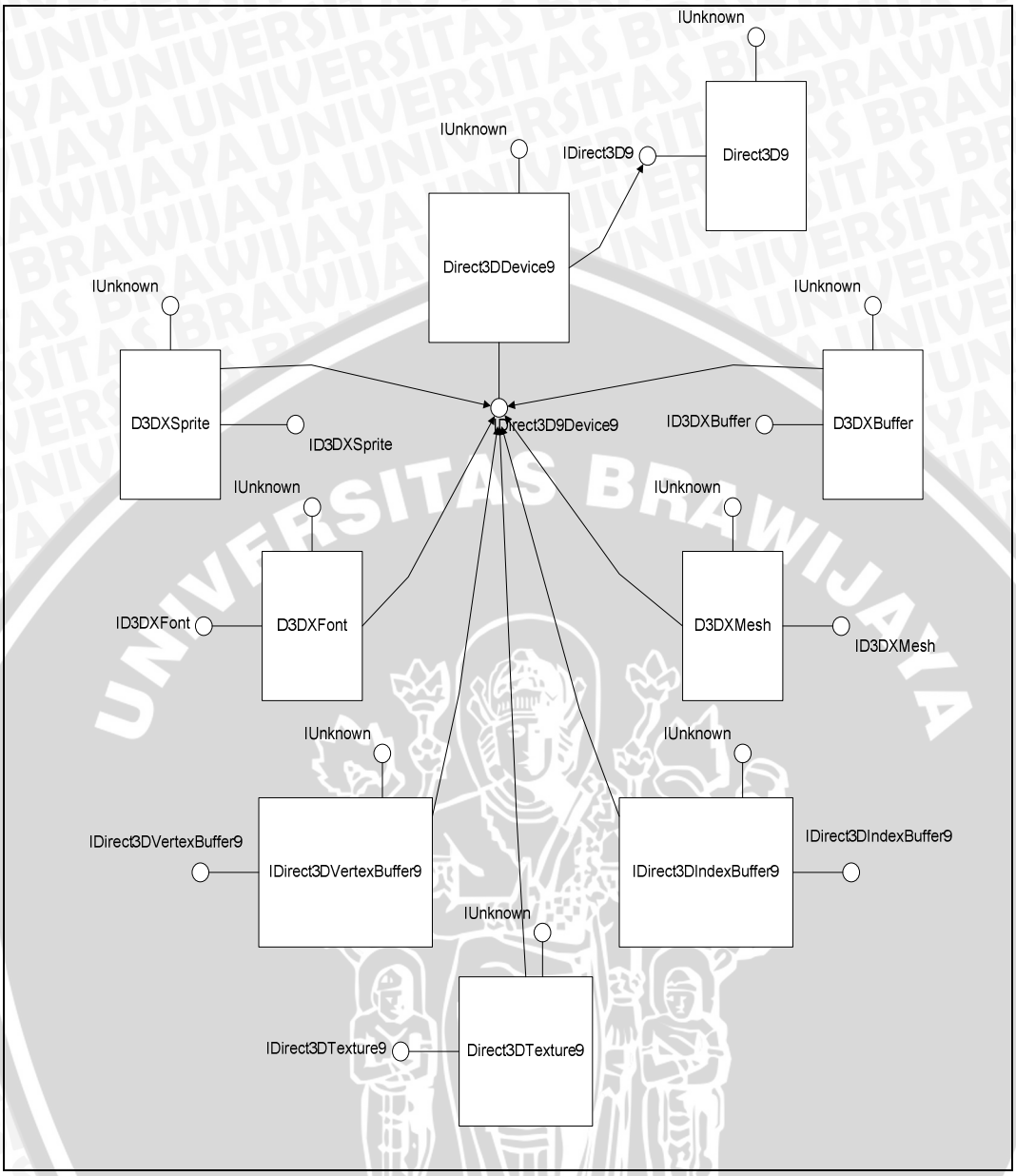
IDirectInput8	Objek COM DirectInput yang berfungsi sebagai antarmuka pemrograman untuk mendapatkan status atau informasi tentang perangkat keras masukan seperti mouse dan keyboard.
IDirectInputDevice8	Objek COM Direct3D yang berfungsi sebagai virtualisasi perangkat keras masukan seperti kartu grafis dan berguna sebagai antarmuka pemrograman terhadap perangkat keras masukan.

4.1.3 COM Diagram

COM diagram merupakan diagram yang menggambarkan relasi antar sesama objek antarmuka COM. Objek COM disimbolkan dengan memiliki minimal satu antarmuka atau *interface* bernama IUnknown. Meskipun objek-objek COM sebenarnya bersifat independen, terdapat hirarki di dalam pendefinisian objek antarmuka. Ada beberapa objek COM yang menjadi referensi dalam DirectX. Objek COM yang menjadi referensi biasanya didefinisikan dengan fungsi *non*-COM yang disediakan oleh *library*. Objek COM

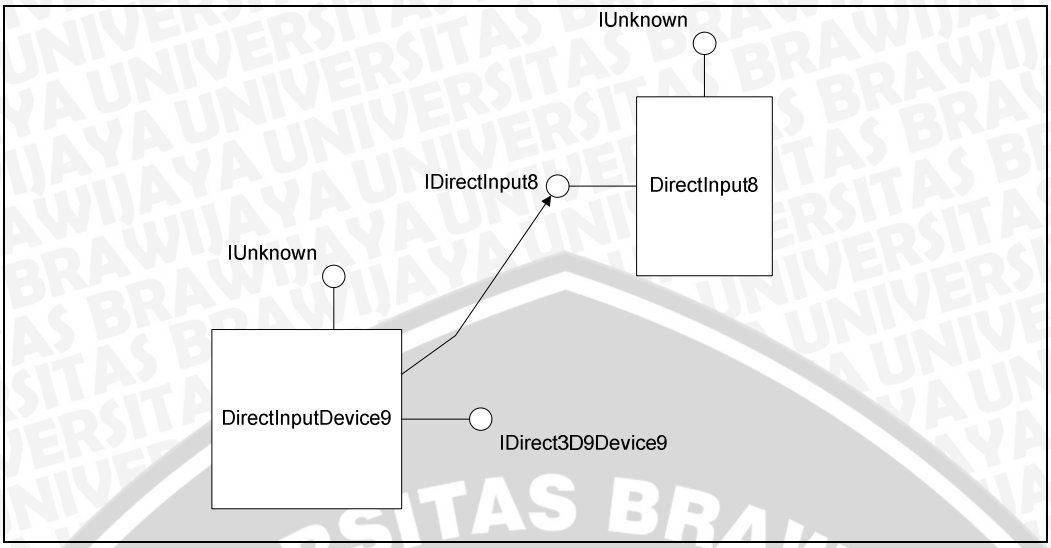
yang dijadikan referensi merupakan objek COM yang mempunyai peran vital atau berisi mengenai informasi fisik dari sistem.

Menurut analisis kebutuhan, sistem memerlukan dua COM diagram. Diagram yang pertama menunjukkan relasi antar objek COM dalam subset DirectX yang khusus menangani permasalahan grafis yaitu DirectX Graphics. Tanda panah menunjukkan bahwa salah satu dari objek COM menjadi objek referensi dari objek COM yang lain. Objek COM Direct3D yang berisi informasi mengenai perangkat keras *display adapter* dibutuhkan oleh Objek COM Direct3DDevice untuk membuat objek antarmuka yang berperan virtual dalam pemerintahan *display adapter*. Objek Direct3DDevice yang diposisikan sebagai representasi dari kartu grafis menjadi referensi bagi objek COM lain yang memerlukan akses kerja dari kartu grafis atau *display adapter* yang ada di dalam sistem.



Gambar 4.2 COM Diagram untuk DirectX Graphics

Diagram yang kedua menunjukkan relasi antar objek COM dalam subset DirectX yang khusus menangani permasalahan input output yaitu DirectInput. Sama seperti halnya objek COM Direct3D, objek antarmuka DirectInput8 juga berfungsi untuk mendapatkan informasi fisik mengenai perangkat keras masukan dan keluaran. Di dalam DirectInput8 terdapat *method* yang berfungsi untuk membuat objek antarmuka DirectInputDevice8.



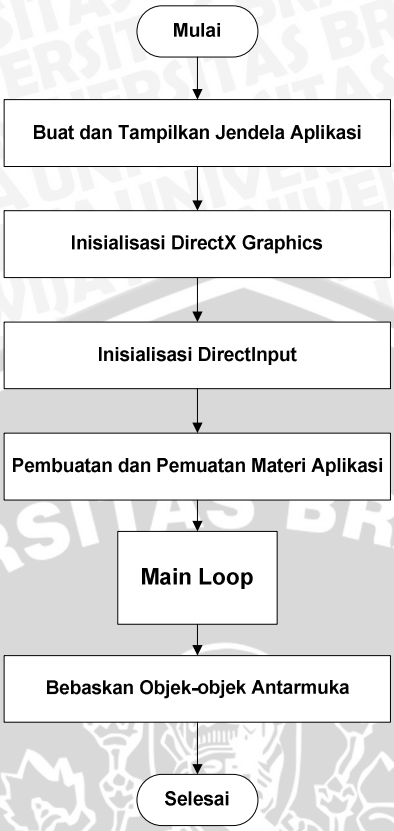
Gambar 4.3 COM Diagram untuk DirectInput

4.2 Perancangan Sistem

Perancangan sistem ditujukan untuk mengetahui gambaran utama kerja sistem secara keseluruhan. Perancangan sistem perlu dilakukan sebelum menginjak kepada bahasa pemrograman. Perancangan sistem menentukan aliran proses di dalam sebuah aplikasi yang akan dibuat. Perancangan sistem akan digambarkan menggunakan diagram alir atau *flowchart* untuk mengetahui fase-fase yang dilalui dalam algoritma aplikasi.

4.2.1 Perancangan Kerangka Utama Program

Perancangan kerangka utama program bertujuan untuk memberikan gambaran mengenai aliran sekuensial sistem secara keseluruhan dari tahap awal dijalankannya aplikasi hingga akhir akhir aplikasi.



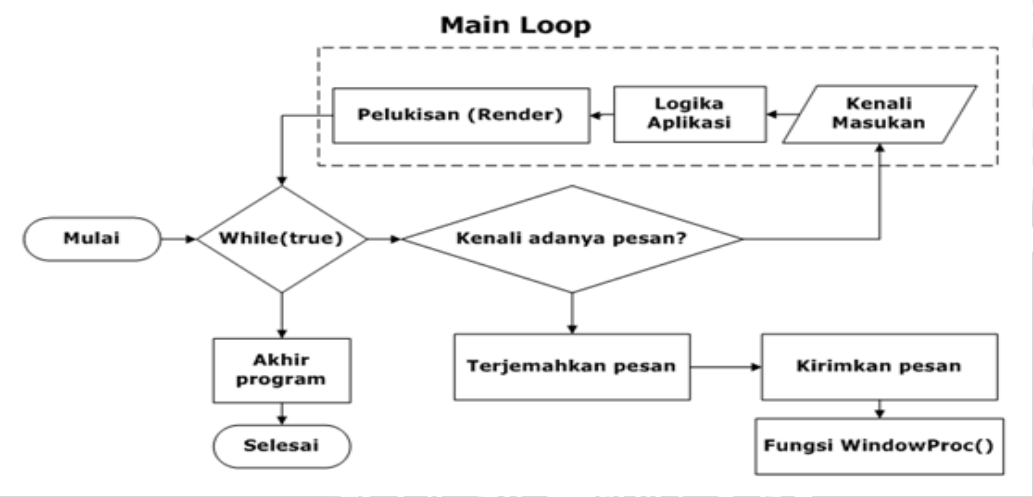
Gambar 4.4 Diagram Alir untuk Kerangka Utama Program

Pada diagram alir ditunjukkan bahwa proses pertama yang harus dilakukan oleh sistem adalah membuat dan menampilkan jendela aplikasi sebagai media penampil aplikasi. Tahapan berikutnya adalah inisialisasi objek-objek COM DirectX Graphics dan DirectInput yang akan digunakan oleh sistem. Setelah selesai melakukan inisialisasi, fase berikutnya merupakan pembuatan dan pemuatan materi aplikasi dalam hal ini berupa objek atau model 3D.

Main Loop merupakan tahapan utama di dalam sistem. Pada tahapan ini aplikasi memberikan logika utama sistem terhadap masukan dari *user* kemudian melakukan render. Setelah aplikasi masuk ke tahap Main Loop, proses perulangan terjadi terus-menerus sampai *user* memutuskan untuk menghentikan aplikasi. Setelah keluar dari tahapan Main Loop, sistem melakukan pembebasan objek-objek antarmuka yang sebelumnya dijadikan referensi maupun meminta alokasi ruang di memori.

4.2.2 Perancangan Main Loop dan Penanganan Pesan *(Event Handling)*

Pemrograman aplikasi Win32 merupakan pemrograman yang bebasis *event* atau *event driven. Event* merupakan kejadian-kejadian seperti *user* menekan *keyboard* atau menekan *mouse*. Perancangan Main Loop dan penanganan pesan ditujukan supaya program bersifat reaktif dan memberikan keluaran yang sinkron dan responsif terhadap masukan.



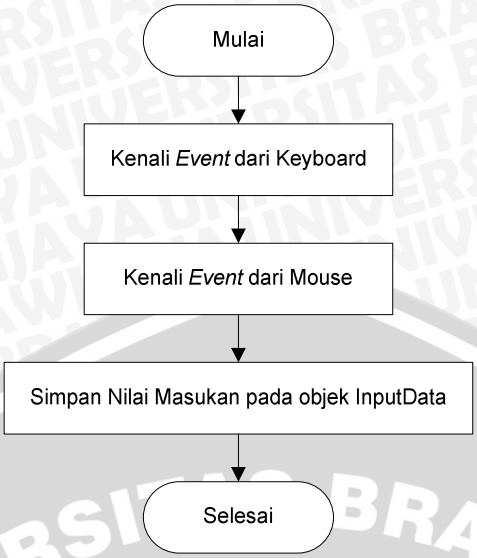
Gambar 4.5 Diagram Alir untuk MainLoop

Dari diagaram alir terdapat prosedur penanganan pesan atau *event handling*. *Event handling* bertujuan mengenali atau mendapatkan pesan yang diterima oleh program. Selama tidak ada *quit* atau *break* maka prosedur penanganan pesan akan menunggu adanya pesan sambil melakukan looping

Main Loop berisikan tiga fungsi utama yaitu input, logika, dan render. Ketiga fungsi ini merupakan elemen terpenting dalam program aplikasi grafis.

4.2.3 Perancangan Fungsi Input

Perancangan fungsi input bertujuan agar sistem dapat mengenali dan mendapatkan masukan atau *event* masukan yang diberikan oleh entitas luar atau *user*.



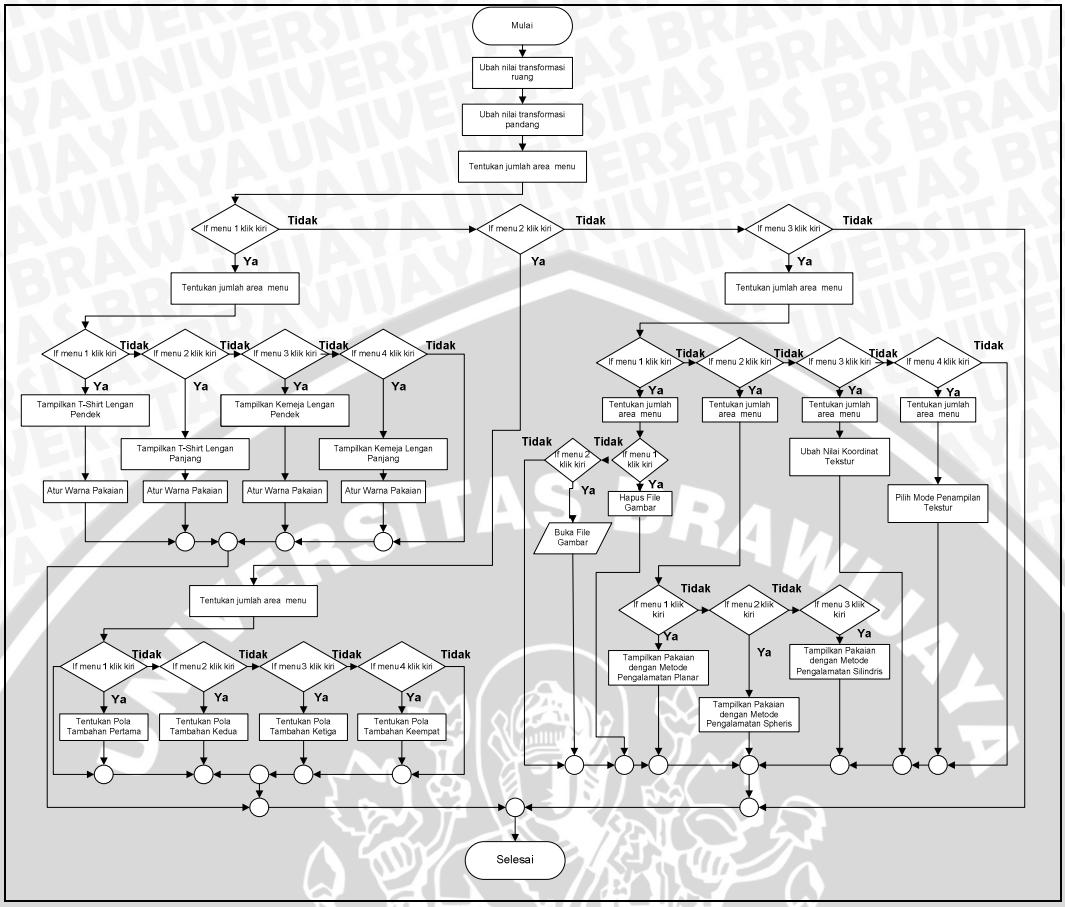
Gambar 4.6 Diagram Alir untuk Fungsi Input

Setelah melakukan pengenalan masukan baik itu merupakan masukan dari *keyboard* ataupun *mouse*, nilai masukan yang didapatkan kemudian disimpan di dalam sebuah objek InputData untuk selanjutnya memberikan pengaruh terhadap fungsi logika.

4.2.4 Perancangan Fungsi Logika

Fungsi logika merupakan fungsi yang dipengaruhi oleh fungsi input dan kemudian mempengaruhi jalannya fungsi render. Fungsi logika bertujuan untuk menentukan respon sistem terhadap pengaruh masukan. Fungsi logika bertugas menentukan objek 3D maupun 2D yang akan dirender oleh sistem. Selain itu fungsi logika juga menentukan nilai dari tahapan-tahapan transformasi.

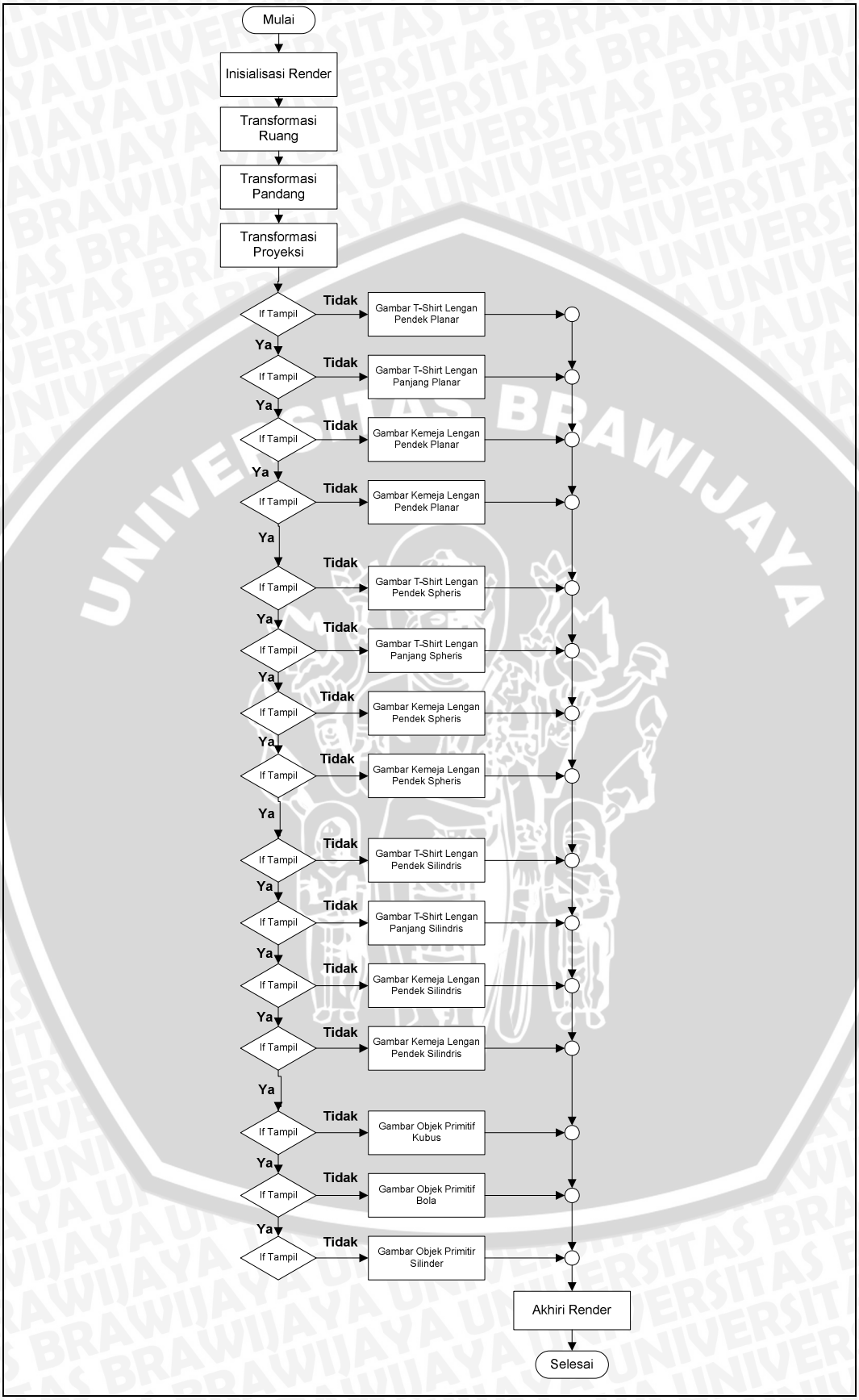
Penentuan nilai koordinat tekstur yang akan diubah nilainya juga dilakukan oleh fungsi logika sebagai bentuk dari manipulasi tekstur.



Gambar 4.7 Diagram Alir untuk Fungsi Logika

4.2.5 Perancangan Fungsi Render

Fungsi render bertujuan untuk memerintahkan perangkat keras grafis atau kartu grafis maupun *display adapter* untuk melakukan *render* atau pelukisan pada layar media penampil atau monitor. Pada fungsi render terdapat objek 3D berupa primitif dan objek 3D berupa model mesh yang akan ditampilkan pada layer media penampil. Seleksi tampil dari objek-objek tersebut ditentukan dalam fungsi logika karena program aplikasi dirancang untuk menampilkan satu objek setiap kali fungsi render dijalankan. Pada fungsi render terdapat tahapan-tahapan transformasi *(transformation pipeline)* sebelum objek 3D yang berada dalam koordinat 3D bisa ditampilkan pada layar media penampil. Fungsi render diawali dengan fungsi pengawalan render dan fungsi pengakhiran render.



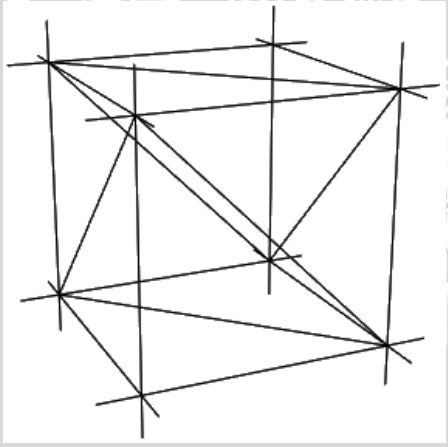
Gambar 4.8 Diagram Alir untuk Fungsi Render

4.3 Pemodelan Objek 3D

Tahapan pemodelan objek 3D pada perancangan bertujuan untuk melakukan rancang bangun atau pemodelan secara primitif terhadap objek-objek sederhana yang nantinya akan digunakan untuk melakukan metode pengalamatan tekstur sebelum akhirnya diterapkan pada objek-objek mesh yang lebih kompleks selain objek primitif.

4.3.1 Pemodelan Objek Kubus

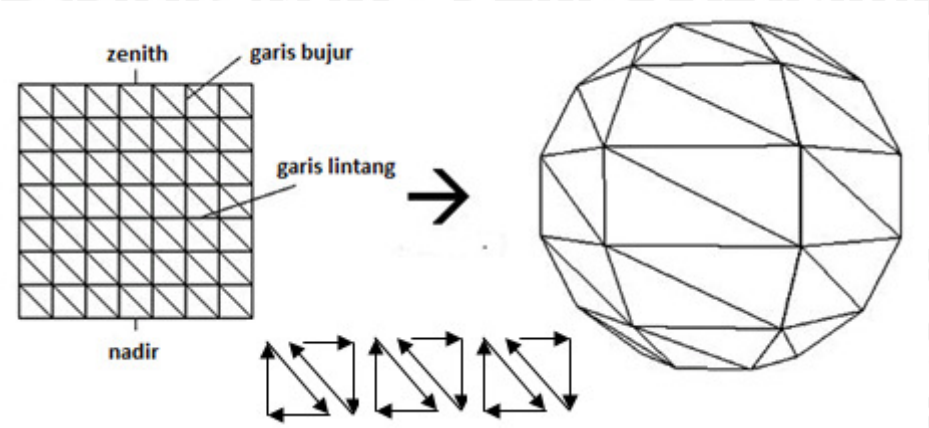
Pembuatan objek kubus bertujuan untuk menerapkan salah satu metode pengalamatan yang sifatnya planar atau linier. Objek kubus didefinisikan dengan jumlah verteks senilai 8 buah dengan jumlah indeks senilai 36 buah. Pendefinisian secara primitif dilakukan dengan menggunakan daftar segitiga atau *triangle list* sebanyak 12 buah, dengan rincian dua segitiga untuk masing-masing sisi. Pendefinisian verteks dan indeks secara primitif harus berurutan dan mengikuti aturan *winding order* yaitu searah jarum jam. Masing-masing titik atau verteks diberikan atribut normal vektor dan koordinat tekstur.



Gambar 4.9 Pemodelan Objek Kubus secara Primitif

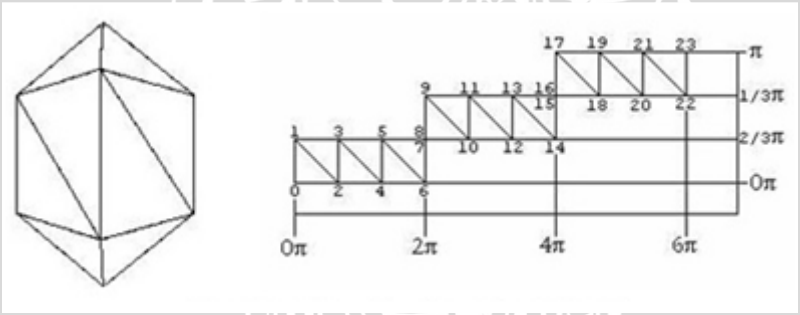
4.3.2 Pemodelan Objek Bola

Pembuatan objek bola bertujuan untuk menerapkan salah satu metode pengalamatan yang bersifat spheris atau melingkar seperti bola. Pembuatan objek bola secara primitif dilakukan dengan mendefinisikan verteks dan indeks dengan ketentuan sesuai *winding order* atau searah jarum.



Gambar 4.10 Pemodelan Objek Bola secara Primitif

Pembuatan model bola disederhanakan dengan membuat lembaran yang nantinya mempunyai orientasi posisi terhadap sudut phi dan theta. Kumpulan daftar verteks yang berada di masing-masing kutub nantinya diposisikan berhimpit sehingga menjadi berada pada satu titik yang disebut titik zenith untuk bagian atas dan titik nadir pada bagian bawah lembaran.



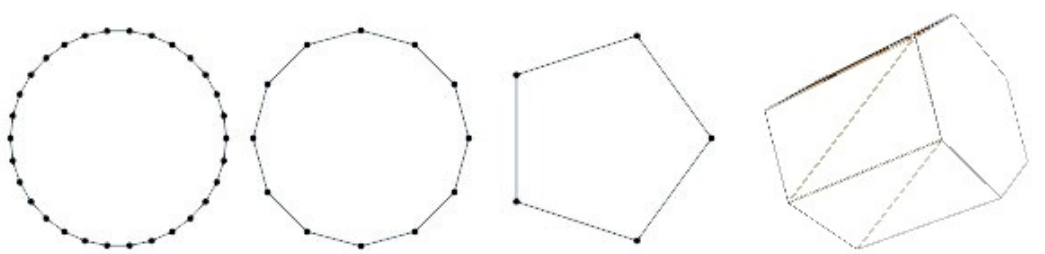
Gambar 4.11 Pengurutan indeks terhadap kumpulan verteks

Pendaftaran *(listing)* indeks dilakukan pada masing-masing segmen baris dengan mencari nilai jari-jari untuk tiap-tiap baris kemudian memposisikannya secara berdekatan dengan baris berikutnya. Nilai kebulatan *(round value)* ditentukan dengan nilai banyak sisi yang akan dimasukkan terhadap pendefinisian objek bola.

4.3.3 Pemodelan Objek Silinder

Pembuatan objek silinder bertujuan untuk menerapkan salah satu metode pengalamatan yang bersifat silindris atau berbentuk seperti silinder. Pembuatan

objek silinder secara primitif dilakukan dengan mendefinisikan verteks dan indeks dengan membaginya menjadi tiga bagian.



Gambar 4.12 Pembuatan Objek Silinder secara Primitif

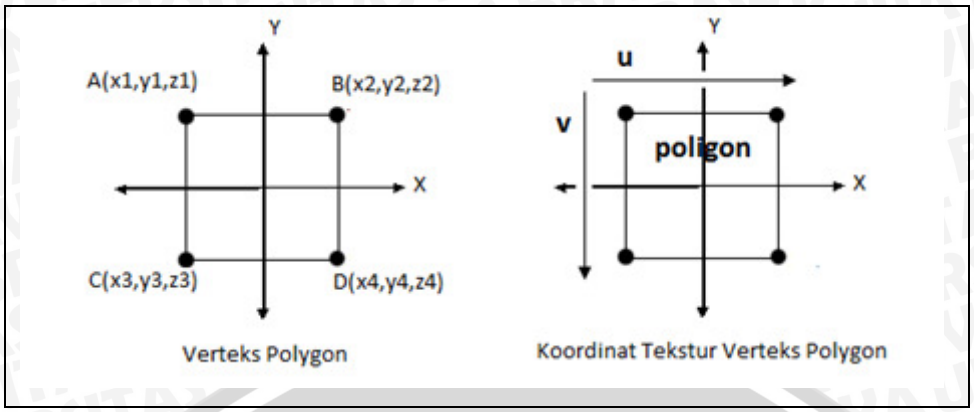
Tiga bagian yang akan menyusun bangun silinder adalah bagian *cap* atau penutup dengan jumlah dua buah dan bagian badan silinder. Urut-urutan verteks dan indeks yang pertama didefinisikan secara sejajar untuk masing-masing penutup kemudian didefinisikan bagian badan silinder dengan urutan verteks yang bersesuaian dengan masing-masing penutup.

4.4 Metode Pengalamatan Tekstur

Metode pengalamatan tekstur adalah cara pemberian atau cara pendistribusian nilai koordinat tekstur pada objek 3D agar sebuah objek 3D mempunyai orientasi terhadap sebuah gambar 2D yang dialamati. Ada tiga metode utama yang akan digunakan dalam perancangan yaitu metode pengalamatan secara planar atau linier, metode pengalamatan secara spheris, dan metode pengalamatan secara silindris.

4.4.1 Metode Pengalamatan Planar

Metode pengalamatan planar yang akan dilakukan dalam perancangan adalah dengan memberikan nilai u dan v yang sebanding dengan masing-masing sumbu x dan y.



Gambar 4.13 Metode pengalamatan planar

Dengan orientasi gambar yang berada pada titik tengah maka nilai koordinat u dan v dapat diperoleh dengan menggunakan persamaan

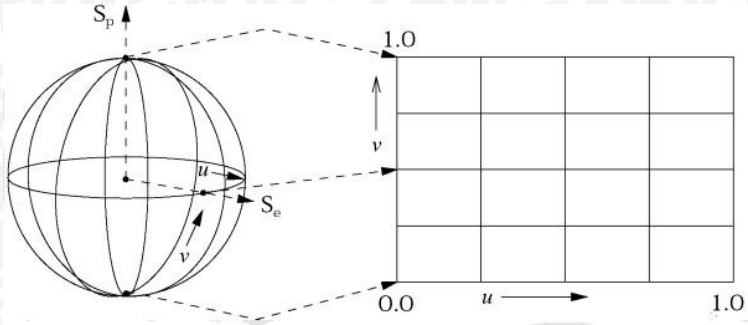
$$koordinat\ tekstur\ u = \frac{x}{x\ max\ -\ x\ min} + 0{,}5$$

$$koordinat\ tekstur\ v = -\frac{y}{y\ max\ -\ y\ min} + 0{,}5$$

Apabila nilai x dan y pada verteks poligon berada pada titik tengah maka akan diperoleh nilai koordinat tekstur (0,5,0,5). Nilai y pada verteks poligon dikalikan negatif karena unit koordinat tekstur v mempunyai orientasi dari atas ke bawah seperti halnya layar monitor.

4.4.2 Metode Pengalamatan Spheris

Metode pengalamatan spheris yang akan dilakukan dalam perancangan adalah dengan memberikan koordinat tekstur u dan v yang sebanding dengan sudut phi dan theta. Pada perancangan akan digunakan dua metode dalam menerapkan pengalamatan spheris yaitu dengan menggunakan persamaan koordinat bola ditambah sudut hasil dot product, dan dengan menggunakan normal vektor.



Gambar 4.14 Metode pengalamatan spheris

4.4.2.1 Berdasarkan Persamaan Koordinat Bola dan sudut dot product

Pada persamaan koordinat bola terdapat nilai sudut θ yang berkisar antara 0 hingga π yang sebanding dengan nilai koordinat tekstur v. Sedangkan sudut φ yang berkisar antara 0 hingga 2π sebanding dengan nilai koordinat tekstur u. Dengan menggunakan persamaan koordinat bola, bisa diperoleh nilai u dan v yang sebanding dengan nilai sudut φ dan θ.

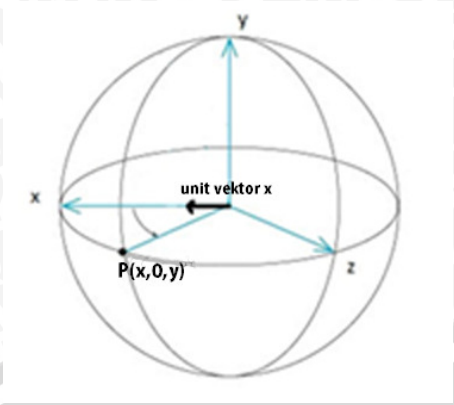
x = r sin(v π) cos(u 2 π)
z = r sin(v π) sin(u 2 π)
y = r cos(v π)

Dengan melakukan penurunan persamaan maka akan diperoleh penentuan nilai koordinat tekstur yaitu

$$koordinat\ tekstur\ v = \frac{arcos(y/r)}{\pi}$$

$$koordinat\ tekstur\ u = \frac{arcos\left(\frac{x}{r\ sin(v\ \pi)}\right)}{2\pi}$$

Karena pada persamaan koordinat tekstur u terdapat penyebut yang dapat bernilai 0 maka persamaan tersebut tidak dapat dipakai. Untuk menentukan nilai koordinat tekstur u bisa diperoleh dengan menggunakan sudut hasil dot product.



Gambar 4.15 Pencarian sudut menggunakan dot product

Yang dimaksud dengan sudut hasil dot product adalah sudut antara vektor posisi verteks setelah dihilangkan elemen y semisal vektor $\mathrm{P} = (x, 0, z)$ dengan unit vektor senilai (1,0,0). Unit vektor senilai (1,0,0) dinotasikan dengan simbol $\vec{\imath}$. Sehingga didapatkan persamaan baru koordinat tekstur u yaitu

$$koordinat\ tekstur\ u = \frac{\arccos\left(\mathrm{P} \cdot \vec{\imath}\right)}{2\pi}$$

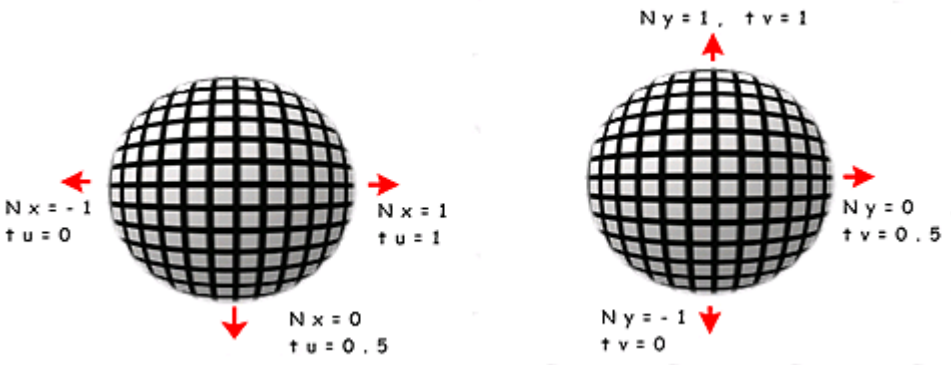
Nilai-nilai sudut kosinus pada kuadran I dan II bernilai sama dengan nilai-nilai sudut kosinus pada kuadran III dan IV. Agar tidak terjadi duplikasi koordinat tekstur u pada verteks-verteks yang berada pada kuadran III dan IV maka nilai-nilai u diperoleh dengan pengurangan skala unit koordinat tekstur u. Untuk menentukan apakah posisi verteks berada pada kuadran I dan II atau pada kuadran III dan IV dapat menggunakan nilai normal vektor z yang menunjukkan arah terhadap titik pusat.

$$koordinat\ tekstur\ u = 1 - \frac{\arccos\left(\mathrm{P} \cdot \vec{\imath}\right)}{2\pi}$$

4.4.2.2 Berdasarkan Normal Vektor

Pengalamatan pada bidang spheris dapat ditentukan menggunakan normal vektor x dan normal vektor y. Nilai koordinat tekstur u dapat dicari dengan menggunakan normal vektor x yang mempunyai nilai antara -1 hingga 1

sedangkan nilai koordinat tekstur v dapat dicari dengan menggunakan normal vektor y yang juga berkisar antara −1 hingga 1.'



Gambar 4.16 Metode pengalamatan spheris berdasarkan normal vektor

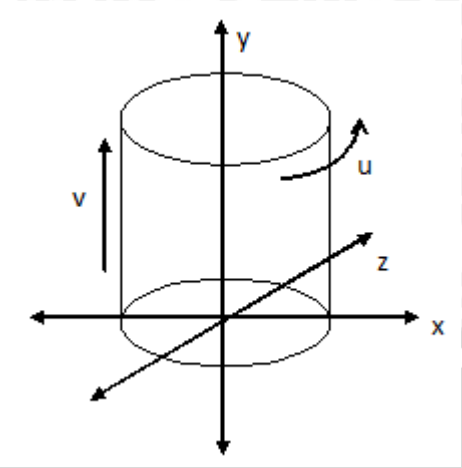
Karena nilai normal vektor yang bersifat linier terhadap tiap-tiap sudut maka nilai-nilai koordinat tekstur u dan v dapat dicari dengan menggunakan orientasi terhadap titik tengah sehingga didapat persamaan

$$koordinat\ tekstur\ u = \frac{\text{asin}\ (N_x)}{\pi} + 0{,}5$$

$$koordinat\ tekstur\ v = -\frac{\text{asin}\ (N_y)}{\pi} + 0{,}5$$

4.4.3 Metode Pengalamatan Silindris

Metode pengalamatan sinder yang akan dilakukan dalam perancangan adalah dengan memberikan koordinat tekstur u yang sebanding dengan sudut φ sedangkan untuk koordinat tekstur v didapatkan secara linier atau sebanding dengan sumbu y. Pada perancangan akan digunakan dua metode dalam menerapkan pengalamatan silinder yaitu dengan menggunakan persamaan koordinat silinder, dan dengan menggunakan normal vektor.



Gambar 4.17 Metode pengalamatan silindris

4.4.3.1 Berdasarkan Persamaan Koordinat Silinder

Pada persamaan koordinat silinder terdapat sudut φ yang berkisar antara 0 hingga 2π sebanding dengan nilai koordinat tekstur u. Dengan menggunakan persamaan koordinat silinder, bisa diperoleh nilai u yang nilainya berbanding lurus dengan besar sudut φ.

$$x = r\cos\theta = r\cos(u\,2\pi)$$
$$z = r\sin\theta = r\,sin(u\,2\pi)$$
$$y = y$$

Dengan melakukan penurunan persamaan maka akan diperoleh penentuan nilai koordinat tekstur u yang serupa dengan persamaan koordinat bola yaitu

$$koordinat\ tekstur\ u = \frac{arcos\left(\frac{x}{r}\right)}{2\pi}$$

Agar tidak terjadi duplikasi koordinat tekstur u pada verteks-verteks yang berada pada kuadran III dan IV maka nilai-nilai u diperoleh dengan pengurangan skala unit koordinat tekstur u untuk nilai normal vektor z yang mempunyai nilai atau arah negatif terhadap titik tengah objek.

Jika $N_Z < 0$

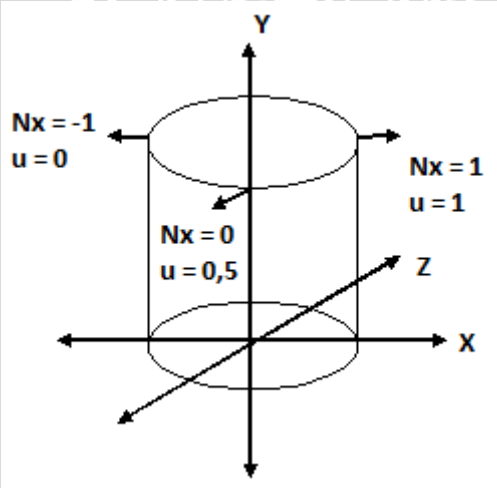
$$koordinat\ tekstur\ u = 1 - \frac{arcos\left(\frac{x}{r}\right)}{2\pi}$$

Untuk penentuan koordinat tekstur v dapat diperoleh dengan menggunakan pengalamatan planar yang sesuai dengan bidang datar atau linier sesuai dengan tinggi yang dimiliki silinder.

$$koordinat\ tekstur\ v = -\frac{y}{y\ max\ -\ y\ min} + 0{,}5$$

4.4.3.2 Berdasarkan Normal Vektor

Pengalamatan pada bidang silindris dapat ditentukan menggunakan normal vektor x dan normal vektor y yang mempunyai nilai antara -1 hingga 1.



Gambar 4.18 Pengalamatan silindris berdasarkan normal vektor

Karena nilai normal vektor yang bersifat linier terhadap sudut φ maka nilai-nilai koordinat tekstur u dapat dicari dengan menggunakan orientasi terhadap titik tengah sehingga didapat persamaan

$$koordinat\ tekstur\ u = \frac{\text{asin}\ (N_x)}{\pi} + 0{,}5$$

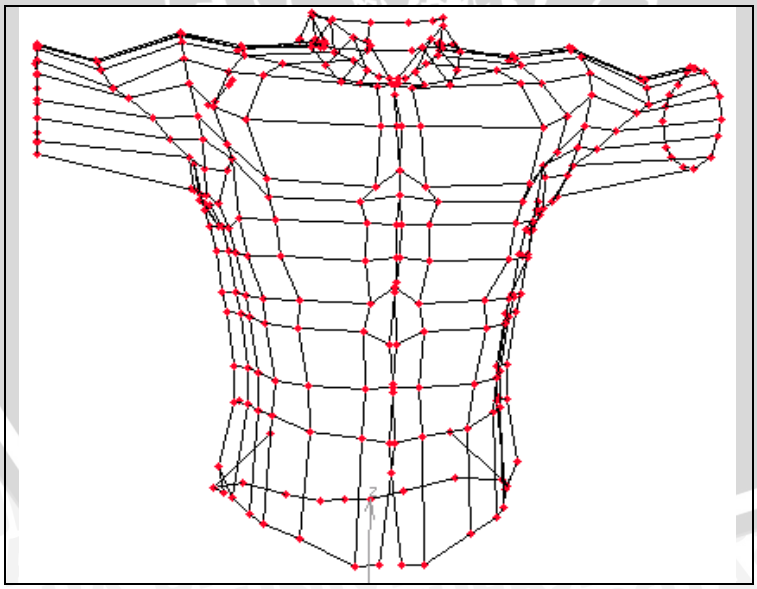
Untuk penentuan koordinat tekstur v dapat diperoleh dengan menggunakan pengalamatan planar yang sesuai dengan bidang datar atau linier sesuai dengan tinggi yang dimiliki silinder.

$$koordinat\ tekstur\ v = -\frac{y}{y\ max\ -\ y\ min} + 0{,}5$$

4.5 Pemodelan Objek Pakaian

Pemodelan objek kompleks berupa pakaian dilakukan dengan menggunakan alat bantu berupa perangkat lunak pemodelan yaitu 3DSMAX. Pendefinisian posisi dan urutan koneksi verteks dilakukan dengan antarmuka visual sehingga memudahkan dalam hal pemodelan.

Pemodelan objek pakaian dilakukan dengan teknik simetris yaitu hanya sebagian daftar verteks yang didefinisikan, selebihnya menggunakan alat bantu berupa refleksi simetris *(mirroring)* untuk menggandakan verteks dengan posisi yang berseberangan. Untuk memperhalus *(smoothing)* objek pakaian digunakan alat bantu berupa meshsmooth pada 3DSMAX dengan prosedur aktual berupa penyisipan sejumlah verteks pada verteks-verteks yang sudah didefinisikan sebelumnya sehinga banyak verteks menjadi berlipat ganda sehingga memperoleh kesan halus atau absennya segmentasi.



Gambar 4.19 Pendefisian verteks dengan antarmuka visual menggunakan perangkat lunak pemodelan 3DSMAX

Proses selanjutnya dalam pemodelan objek pakaian adalah mengkonversi verteks-verteks hasil pendefinisian secara visual menggunakan alat bantu

3DSMAX menjadi daftar verteks dan daftar indeks sesuai dengan *template* atau *metadata* struktural dan deskriptif milik *file* berekstensi .x. Pengkonversian dari data 3DSMAX menjadi data .x dapat dilakukan secara otomatis dengan menggunakan program *exporter* atau komponen yang ditambahkan pada perangkat lunak 3DSMAX disesuaikan dengan kebutuhan *template* yang akan digunakan oleh penulis. Penulis menggunakan komponen milik perangkat lunak QUEST3D untuk mengkonversi data 3DSMAX atau data vektor menjadi data array di dalam file berekstensi .x.

Template yang akan digunakan untuk menyimpan data verteks dan indeks merupakan template yang tersusun atas :

a. Material berupa daftar material yaitu nilai-nilai warna verteks
b. Jumlah verteks dan nilai posisi tiap-tiap verteks dalam bentuk daftar verteks
c. Jumlah permukaan *(surface/triangle list)* dan daftar indeks (referensi terhadap verteks yang telah didefinisikan) penyusun permukaan
d. Jumlah material (banyaknya warna verteks sesuai daftar material) dan daftar indeks yang menggunakan material
e. Nilai-nilai normal verteks atau daftar normal verteks untuk menentukan kuantitas cahaya yang diterima oleh verteks
f. Daftar indeks yang mengacu pada daftar verteks yang menggunakan nilai-nilai normal verteks.

Untuk memperjelas perihal *template* yang digunakan dalam perancangan objek pakaian, diberikan contoh ilustrasi *pseudocode* pendefinisian sebuah objek 3D berupa kubus yang disimpan dalam struktur format file .x sebagai berikut

```
Material WarnaHijau {
0.000000;1.000000;0.000000;1.000000;;     // R = 0.0, G = 1.0, B = 0.0
0.000000;
0.000000;0.000000;0.000000;;
0.000000;0.000000;0.000000;;
}
// objek 3D dengan jumlah verteks 8 dan jumlah permukaan 12 (segitiga-
segitiga).
// informasi objek 3D berupa material, posisi verteks atau daftar verteks,
//dafar indeks, daftar segitiga(triangle list), dan normal verteks,
Mesh Kotak {
8;                                // jumlah verteks
1.000000;1.000000;-1.000000;,     // verteks ke-0
-1.000000;1.000000;-1.000000;,    // verteks ke-1
-1.000000;1.000000;1.000000;,
```

```
1.000000;1.000000;1.000000;,
1.000000;-1.000000;-1.000000;,
-1.000000;-1.000000;-1.000000;,
-1.000000;-1.000000;1.000000;,
1.000000;-1.000000;1.000000;;     // verteks ke-7

12;                          // jumlah permukaan
3;0,1,2;,                    // setiap permukaan terdiri dari 3 verteks
3;0,2,3;,
3;0,4,5;,
3;0,5,1;,
3;1,5,6;,
3;1,6,2;,
3;2,6,7;,
3;2,7,3;,
3;3,7,4;,
3;3,4,0;,
3;4,7,6;,
3;4,6,5;;

MeshMaterialList {
1;                    // jumlah material yang digunakan
12;                   // material digunakan untuk setiap permukaan
0,                    // material pertama (material urutan ke-0) digunakan-
0,                    // oleh semua permukaan objek 3D
0,
0,
0,
0,
0,
0,
0,
0,
0,
0;;
{Warna Hijau}   // referensi material yang digunakan
}
MeshNormals {
8;                  // pendefinisian normal vektor untuk masing-masing verteks
0.333333;0.666667;-0.666667;,
-0.816497;0.408248;-0.408248;,
-0.333333;0.666667;0.666667;,
0.816497;0.408248;0.408248;,
0.666667;-0.666667;-0.333333;,
-0.408248;-0.408248;-0.816497;,
-0.666667;-0.666667;0.333333;,
0.408248;-0.408248;0.816497;;
12;                 // tentukan normal vektor untuk tiap-tiap permukaan
3;0,1,2;,
3;0,2,3;,
3;0,4,5;,
3;0,5,1;,
3;1,5,6;,
3;1,6,2;,
3;2,6,7;,
3;2,7,3;,
3;3,7,4;,
3;3,4,0;,
3;4,7,6;,
3;4,6,5;;
}
```

Kode ilustrasi *pseudocode* dari *template* yang digunakan dalam pemodelan objek 3D

BAB V
IMPLEMENTASI

Pada bab ini dibahas mengenai implementasi perangkat lunak berdasarkan hasil yang telah didapatkan dari perancangan perangkat lunak yang telah dibuat. Implementasi merupakan proses transformasi hasil perancangan perangkat lunak ke dalam kode program *(coding)* sesuai dengan sintaks dari bahasa pemrograman yang digunakan. Pembahasan terdiri dari penjelasan tentang Lingkungan Implementasi, Algoritma Implementasi, dan Implementasi Antarmuka Aplikasi.

5.1 Lingkungan Implementasi

Sistem dibuat dengan menggunakan bahasa permrograman C++ dengan menggunakan IDE*(Integrated Development Environment)* Microsoft Visual Studio 2008, dan SDK*(Software Development Kit)* DirectX SDK yang dipublikasikan pada bulan Juni 2010. Sistem diimplementasikan dengan menggunakan spesifikasi sebagai berikut:

5.1.1 Spesifikasi Perangkat Keras

- CPU : Intel Core i3 2330-M (2.2 GHz)
- *Memory* RAM : 2 GB RAM
- *Hard Disk Drive* : 160 GB
- *Graphics Adapter* : ATI Radeon HD 6730M 2GHz

5.1.2 Spesifikasi Perangkat Lunak

- Sistem operasi : Windows 7
- IDE : Microsoft Visual Studio 2008
- SDK : DirectX SDK (Juni 2010)
-

5.2 Implementasi Algoritma Sistem

Implementasi algoritma sistem merupakan baris-baris kode program atau manifestasi kode program yang digunakan dalam pembuatan perangkat lunak. Implementasi algoritma sistem dibuat berdasarkan tahap perancangan.

5.2.1 Implementasi Kerangka Utama Program

Fungsi utama merupakan prosedur utama dari program. Kerangka utama program menentukan aliran program dari awal dijalankannya aplikasi hingga aplikasi diakhiri. Kerangka utama program berjalan di dalam pemrograman win32 yang berarti mempunyai fungsi penanganan pesan yang disebut dengan WindowProc();

```
1   int WINAPI WinMain(HINSTANCE hInstance,
2                      HINSTANCE hPrevInstance,
3                      LPSTR lpCmdLine,
4                      int nCmdShow)
5   {
6       GAMEWINDOW gw;
7       InitDirect3D(&gw);
8       DisplayWindow(&gw, hInstance, nCmdShow);
9       InitDirectInput(hInstance, &gw);
10      LoadGraphics();
11      MainLoop(gw);
12      CloseDirect3D();
13      CloseDirectInput();
14
15      return 0;
16  }
17  LRESULT CALLBACK WindowProc(HWND hWnd, UINT message, WPARAM wParam,
18  LPARAM lParam)
19  {
20      switch(message)
21      {
22                  case WM_KEYDOWN:
23                      {
24                          switch( wParam )
25                              {
26                   case VK_ESCAPE:
27                   PostQuitMessage(0);
28                   return 0;
29                                  break;
30                                  }
31                              }break;
32          case WM_DESTROY:
33              {
34                  PostQuitMessage(0);
35                  return 0;
36              } break;
37      }
38      return DefWindowProc (hWnd, message, wParam, lParam);
39  }
```

Kode implementasi fungsi utama

Pada kode implementasi dari baris ke-6 hingga baris ke-13 diperlihatkan urut-urutan fungsi yang dijalankan oleh sistem. Dimulai dengan pembuatan jendela aplikasi untuk menampilkan program kemudian dilanjutkan dengan inisialisasi objek antarmuka DirectX dan DirectInput. Setelah selesai dilakukannya inisialisasi objek antarmuka, kemudian sistem memuat materi berupa gambar 2D maupun objek 3D.

Setelah proses pemuatan selesai kemudian program masuk ke dalam fungsi MainLoop() yang merupakan tahapan utama dari aplikasi. Jika tidak ada pesan *quit* atau *break* program akan terus melakukan *looping* pada MainLoop sambil menunggu adanya masukan dari *user*. Setelah adanya proses *quit* atau *break*, selanjutnya aliran program keluar dari MainLoop() kemudian dilakukan pembebasan memori dan objek-objek antarmuka.

5.2.2 Implementasi Inisialisasi DirectX Graphics

Pada proses inisialisasi DirectX Graphics, dilakukan pembuatan objek-objek COM yang diperlukan oleh sistem dalam melakukan penanganan grafis. Pembuatan objek-objek COM harus mengikuti COM diagram karena dalam DirectX terdapat aturan mengenai objek-objek yang menjadi referensi dan objek-objek yang mereferensi.

```
1   LPDIRECT3D9 d3d = NULL;
2   LPDIRECT3DDEVICE9 d3ddev = NULL;
3   D3DPRESENT_PARAMETERS d3dpp;
4   if(FAILED(d3d = Direct3DCreate9(D3D_SDK_VERSION)))
5       {
6           MessageBox(NULL, "Gagal Membuat Antarmuka Direct3D", "Error",
7   MB_OK);
8           exit(0);
9       }
10
11  if(FAILED(d3d-
12  >GetAdapterDisplayMode(D3DADAPTER_DEFAULT,&ModeTampilanDesktop)))
13      {
14          MessageBox(NULL, "Gagal Mengambil Informasi Tampilan Desktop",
15  "Error", MB_OK);
16          exit(0);
17      }
18  if(FAILED(d3d-
19  >GetAdapterDisplayMode(D3DADAPTER_DEFAULT,&ModeTampilanDesktop)))
20      {
21          MessageBox(NULL, "Gagal Mengambil Informasi Tampilan Desktop",
22  "Error", MB_OK);
23          exit(0);
24      }
25          gw->Width = ModeTampilanDesktop.Width;
26
27      ZeroMemory(&d3dpp, sizeof(d3dpp));
28      d3dpp.SwapEffect = D3DSWAPEFFECT_DISCARD;
29      d3dpp.BackBufferFormat = D3DFMT_X8R8G8B8;
30      d3dpp.Windowed = gw->Windowed;
31      d3dpp.BackBufferWidth = gw->Width;
32      d3dpp.BackBufferHeight = gw->Height;
33      d3dpp.EnableAutoDepthStencil = TRUE;
34      d3dpp.AutoDepthStencilFormat = D3DFMT_D16;
35
36  if(FAILED(d3d->CreateDevice(D3DADAPTER_DEFAULT,
37                              D3DDEVTYPE_HAL,
38                              gw->hWnd,
39                              D3DCREATE_SOFTWARE_VERTEXPROCESSING,
40                              &d3dpp,
41                              &d3ddev)))
42      {
43       MessageBox(NULL, "Gagal Membuat Antarmuka Devais Kartu Grafis",
44  "Error", MB_OK);
```

```
45          exit(0);
46      }
47  if(FAILED(d3d-
48  >GetAdapterIdentifier(D3DADAPTER_DEFAULT,0,&InfoKartuGrafis)))
49      {
50        MessageBox(NULL, "Gagal Mengambil Informasi Kartu Grafis Anda",
51  "Error", MB_OK);
52      }
53          if(FAILED(D3DXCreateSprite(d3ddev, &d3dspt)))
54      {
55          MessageBox(NULL, "Gagal Membuat Antarmuka Sprite 2D", "Error",
56  MB_OK);
57          exit(0);
58      }
59      d3ddev->SetRenderState(D3DRS_LIGHTING, TRUE);
60      d3ddev->SetRenderState(D3DRS_ZENABLE, TRUE);
61      d3ddev->SetRenderState(D3DRS_AMBIENT, D3DCOLOR_XRGB(50, 50, 50));
```

Kode Implementasi Inisialisasi DirectX Graphics

Pada kode implementasi ditunjukkan bahwa IDirect3D9 merupakan objek COM pertama yang harus dibuat karena berisikan mengenai informasi dari perangkat keras grafis di dalam sistem. Pembuatan objek COM IDirect3D9 dilakukan dengan fungsi *non*-COM yaitu Direct3DCreate9(). Setelah objek IDirect3D9 dibuat, dibutuhkan sebuah *struct* untuk menahan informasi yang didapat dari objek IDirect3D9 yaitu *struct* D3DPRESENT_PARAMETERS. Setelah objek IDirect3D9 dan informasi mengenai perangkat keras grafis disimpan, selanjutnya dibuat objek COM IDirect3DDevice9 yang berfungsi sebagai perangkat keras grafis virtual yang akan digunakan sebagai antarmuka dalam melakukan akses terhadap perangkat keras grafis.

5.2.3 Implementasi Inisialisasi DirectInput

Pada proses inisialisasi DirectInput, dilakukan pembuatan objek-objek COM yang diperlukan oleh sistem perihal perangkat keras input output. Pembuatan objek-objek COM DirectInput juga mengikuti COM diagram.

```
1   LPDIRECTINPUT8 din;
2   LPDIRECTINPUTDEVICE8 dinkeyboard;
3   LPDIRECTINPUTDEVICE8 dinmouse;
4   void InitDirectInput(HINSTANCE hInstance, GAMEWINDOW* gw)
5   {
6           if(FAILED(     DirectInput8Create(hInstance,
7                                  DIRECTINPUT_VERSION,
8                                  IID_IDirectInput8,
9                                  (void**)&din,
10                                 NULL)))
11      {
12          MessageBox(NULL, "Gagal Membuat Antarmuka DirectInput", "Error",
13  MB_OK);
14          exit(0);
15      }
16      tulisbaris("Inisialisasi Antarmuka DirectInput berhasil  .  .  .
17  OK");
18
```

```
19          if(FAILED(din->CreateDevice(GUID_SysKeyboard, &dinkeyboard, NULL)))
20      {
21          MessageBox(NULL, "Gagal Membuat Antarmuka Devais KEYBOARD",
22  "Error", MB_OK);
23          exit(0);
24      }
25      tulisbaris("Inisialisasi Antarmuka Devais KEYBOARD berhasil  .  .  .
26  OK");
27          if(FAILED(din->CreateDevice(GUID_SysMouse, &dinmouse, NULL)))
28      {
29          MessageBox(NULL, "Gagal Membuat Antarmuka Devais MOUSE", "Error",
30  MB_OK);
31          exit(0);
32      }
33      tulisbaris("Inisialisasi Antarmuka Devais MOUSE berhasil  .  .  .
34  OK");
35
36      dinkeyboard->SetDataFormat(&c_dfDIKeyboard);
37      dinmouse->SetDataFormat(&c_dfDIMouse);
38
39      dinkeyboard->SetCooperativeLevel(gw->hWnd, DISCL_NONEXCLUSIVE |
40  DISCL_BACKGROUND);
41      dinmouse->SetCooperativeLevel(gw->hWnd, DISCL_NONEXCLUSIVE |
42  DISCL_BACKGROUND);
43
44      dinmouse->Acquire();
45
46      return;
47  }
```

Kode Implementasi Inisialisasi DirectInput

Pada kode implementasi, diperlihatkan bahwa IDirectInput8 merupakan objek COM yang menjadi referensi bagi objek COM lain sehingga harus dibuat terlebih dahulu. Objek COM IDirectInput8 dibuat dengan menggunakan fungsi *non*-COM DirectInput8Create(). Setelah pembuatan objek COM IDirectInput8 berhasil, kemudian dibuat objek antarmuka IDirectInputDevice8 untuk masing-masing perangkat keras masukan *keyboard* dan *mouse*. Bagian akhir dari kode implementasi merupakan pengaturan format data perangkat keras masukan dan prioritas sistem operasi untuk perangkat keras masukan terhadap program aplikasi lain yang berjalan pada sistem operasi yang sama.

5.2.4 Implementasi Fungsi MainLoop

Fungsi MainLoop merupakan tahapan utama dalam aplikasi. Di dalam fungsi main loop terdapat fungsi input, fungsi logika, dan fungsi render yang mempengaruhi respon dari aplikasi terhadap masukan oleh *user*.

```
1   void MainLoop(GAMEWINDOW gwo)
2   {
3   InitAplikasi(
4                   LogikaObject,
5                   TulisObject,
6                   TransformObject,
7                   KameraObject,
8
```

```
                MappingObject,
                ModelObject,
                TextureObject,
                AnimObject
                );

while(HandleMessages())
{
        Input(&InputData);
        Logic(
                LogikaObject,
                TulisObject,
                TransformObject,
                KameraObject,
                MappingObject,
                ModelObject,
                TextureObject,
                AnimObject,
                gwo,
                &InputData
                );
        Render(
                LogikaObject,
                TulisObject,
                TransformObject,
                KameraObject,
                MappingObject,
                ModelObject,
                TextureObject,
                AnimObject
                );
}
}

```

Kode Implementasi Fungsi MainLoop

Pada fungsi MainLoop, tahapan pertama yang dilakukan adalah pengaturan variabel awal aplikasi pada fungsi initAplikasi() sebelum melakukan *looping*. Setelah masuk ke dalam tahapan *looping* maka fungsi input, fungsi logika, dan fungsi render dijalankan secara sekuensial dan berulang-ulang. Pada fungsi logika dan fungsi render terdapat objek-objek yang bersifat *public* yang berisikan sebagian besar atribut logika program dan digunakan sebagai parameter fungsi.

5.2.5 Implementasi Fungsi Penanganan Pesan *(Event Handling)*

Pembuatan fungsi penanganan pesan bertujuan supaya program bersifat reaktif dan memberikan keluaran yang sinkron dan responsif terhadap masukan.

```
bool HandleMessages()
{
static MSG msg;

if (PeekMessage(&msg, NULL, 0, 0, PM_REMOVE))
```

```
{
if (msg.message == WM_QUIT)
return FALSE;

TranslateMessage(&msg);
DispatchMessage(&msg);
}

return TRUE;
}
```

Kode Implementasi Fungsi Penanganan Pesan

Setelah fungsi input, fungsi logika, dan fungsi render dijalankan sekali, program kemudian memeriksa adanya pesan melalui fungsi HandleMessages() yang berperan sebagai operator *while* yang bertipe *boolean*. Fungsi yang digunakan untuk memeriksa atau mengenali adanya pesan adalah fungsi PeekMessage(). Fungsi PeekMessage() dipilih sebagai pemeriksa adanya pesan pada *event queue* supaya program bersifat lebih reaktif dibandingkan dengan menggunakan fungsi GetMessage(). Jika program mengenali adanya pesan, selanjutnya pesan diterjemahkan dan dikirimkan ke fungsi WindowProc() untuk segera diproses.

5.2.6 Implementasi Pengenalan Masukan dari *Keyboard*

Pengenalan masukan dari *keyboard* dengan menggunakan DirectInput dipilih karena pemrograman win32 tidak menyediakan fungsi yang mengenali tombol asli yang ditekan *(actual keystroke)* dan memberikan *delay* ketika memfilter penekanan yang berulang. DirectInput memberikan data asli atau mentah *(raw data)* sehingga didapatkan sinyal ketika tombol spesifik ditekan.

```
void GetKeys(BYTE* KeyState)
{
    dinkeyboard->Acquire();

    dinkeyboard->GetDeviceState(256, (LPVOID)KeyState);

    for(int Index = 0; Index < 256; Index++)
        *(KeyState + Index) &= 0x80;

    return;
}
```

Kode Implementasi Pengenalan Masukan dari *Keyboard*

Pada kode implementasi, fungsi acquire() digunakan untuk meminta akses perangkat keras masukan supaya tidak digunakan oleh program aplikasi lain yang berjalan pada sistem yang sama. Fungsi GetDeviceState() berguna untuk

memperoleh informasi perangkat keras *keyboard* beserta elemen masukannya serta menyimpannya sebagai *array* ke dalam memori. Untuk mendapatkan status *keyboard* yang ditekan, dilakukan perkalian dengan MSB karena KeyState yang didapatkan bertipe data SHORT bukan BOOL dan pada bagian tertinggi merupakan status *keyboar*d ketika ditekan.

5.2.7 Implementasi Pengenalan Masukan dari *Mouse*

Pengenalan masukan dari *mouse* bertujuan untuk mengenali masukan dari perangkat keras *mouse*. Pada koordinat layar monitor, objek *mouse* yang dibuat dari DirectInput dapat memberikan masukan berupa posisi absolut*(absolute position)* dan posisi relatif*(relative position).* Posisi absolut bekerja berdasarkan posisi koordinat pada layar monitor sedangkan posisi relatif bekerja berdasarkan perubahan posisi *mouse* atau kursor.

```
1   void GetMouse(DIMOUSESTATE* MouseState)
2   {
3       dinmouse->Acquire();
4       dinmouse->GetDeviceState(sizeof(DIMOUSESTATE), (LPVOID)MouseState);
5       MouseState->rgbButtons[0] &= 0x80;
6       MouseState->rgbButtons[1] &= 0x80;
7       MouseState->rgbButtons[2] &= 0x80;
8       MouseState->rgbButtons[3] &= 0x80;
9       return;
10  }
```

Kode Implementasi Pengenalan Masukan dari *Mouse*

Seperti halnya kode implementasi pada pengenalan masukan berupa *keyboard*, fungsi acquire() digunakan untuk meminta akses perangkat keras masukan supaya tidak digunakan oleh program aplikasi lain yang berjalan pada sistem yang sama. Fungsi GetDeviceState() berguna untuk memperoleh informasi perangkat keras *mouse* beserta elemen masukannya serta menyimpannya sebagai *array* ke dalam memori. Untuk pengalokasian memori disesuaikan dengan ukuran *struct* DIMOUSESTATE yang berguna untuk menahan informasi data masukan. Jika pada keyboard hanya terdapat nilai lepas *(up)* dan tekan *(down),* pada mouse terdapat tiga elemen tambahan perihal perubahan posisi kursor pada layar. Nilai rgbButtons 0,1,2 masing-masing berkoresponden dengan klik kiri pada mouse, klik kanan, dan klik tengah.

5.2.8 Implementasi Fungsi Input

Fungsi input bertujuan untuk mengenali tombol masukan yang ditekan oleh user kemudian merubahnya sebagai logika pada sistem. Fungsi input bersifat membatasi tombol mana saja yang akan digunakan oleh logika sistem supaya pemrograman menjadi lebih efisien.

```
1     ZeroMemory(InputData, sizeof(INPUTDATA));
2
3      if(Keys[DIK_LEFT])
4          InputData->k_kiri = true;
5      if(Keys[DIK_RIGHT])
6          InputData->k_kanan = true;
7      if(Keys[DIK_UP])
8          InputData->k_atas = true;
9      if(Keys[DIK_DOWN])
10         InputData->k_bawah = true;
11     if(Keys[DIK_HOME])
12         InputData->k_HOME = true;
13     if(Keys[DIK_END])
14         InputData->k_END = true;
15     if(Keys[DIK_PRIOR])
16         InputData->k_PRIOR = true;
17     if(Keys[DIK_NEXT])
18         InputData->k_NEXT = true;
19
20        InputData->m_scroll = Mouse.lZ;
21        InputData->m_geserx = Mouse.lX;
22        InputData->m_gesery = Mouse.lY;
23
24        if (Mouse.rgbButtons[0])
25                {
26               InputData->m_klik_kiri = true;
27                }
28        else {
29         InputData->m_klik_kiri = false;
30                }
31
32        if      (Mouse.rgbButtons[1])
33                {
34               InputData->m_klik_kanan = true;
35               }
36        else {
37         InputData->m_klik_kanan = false;
38                }
```

Kode Implementasi Fungsi Input

Pada kode implementasi ditentukan tombol-tombol yang akan digunakan oleh logika sistem. Tombol-tombol keyboard DIK_LEFT, DIK_RIGHT, DIK_UP, DIK DOWN nantinya akan digunakan sebagai transformasi posisi objek 3D sedangkan DIK_HOME, DIK_END, DIK_PRIOR, dan DIK_NEXT digunakan sebagai transformasi ruang dan transformasi pandang sebagai orientasi kamera terhadap objek.

Mouse.lX dan Mouse.lY digunakan sebagai pengaruh perubahan posisi kursor horisontal dan vertikal sedangkan Mouse.lZ digunakan untuk arah *scroll*.

Mouse.rgbButtons[0] dan Mouse.rgbButtons[1] digunakan untuk menentukan klik kiri dan klik kanan. Fungsi klik kiri digunakan sebagian besar sebagai penentu logika sesuai dengan gambar sprite (gambar 2D yang berperan sebagai GUI).

5.2.9 Implementasi Pemuatan dan Penyimpanan Gambar Tekstur

Gambar tekstur adalah gambar bitmap 2D yang akan digunakan dalam pemetaan gambar pada objek 3D. Implementasi pemuatan dan penyimpanan tekstur pada memori sangat sederhana karena DirectX Graphics sudah menyediakan fungsi untuk melakukan penanganan perihal pemuatan dan penyimpanan gambar tekstur.

```
1       if(FAILED(D3DXCreateTextureFromFileEx(d3ddev
2                                             ,File
3                                             ,D3DX_DEFAULT
4                                             ,D3DX_DEFAULT
5                                             ,D3DX_DEFAULT
6                                             ,NULL,D3DFMT_A8R8G8B8
7                                             ,D3DPOOL_MANAGED
8                                             ,D3DX_DEFAULT
9                                             ,D3DX_DEFAULT
10                                            ,NULL
11                                            ,NULL,NULL
12                                            ,&pSprite->tex)))
13      {
14                  sprintf(TextSementara,"Gagal Memuat File Gambar %s",File);
15              MessageBox(NULL, TextSementara, "Error", MB_OK);
16                  exit(0);
17      }
```

Kode Implementasi Pemuatan dan Penyimpanan Gambar Tekstur

Salah satu fungsi yang disediakan oleh DirectX adalah fungsi D3DXCreateTextureFromFileEx. Fungsi ini bertugas untuk memuat dan menyimpan gambar tekstur ke dalam sistem memori. Parameter yang dibutuhkan antara lain adalah objek COM IDirect3DDevice9, karena gambar tekstur membutuhkan ruang memori grafis dan perlu adanya pengecekan format data tekstur yang sesuai dengan kapabilitas perangkat keras grafis. Parameter yang kedua merupakan pointer ke tipe data bertipe string untuk mengenali nama file gambar. Parameter ketiga hingga keenam merupakan penentu lebar dan tinggi gambar yang akan disimpan serta sampling filter. Parameter ketujuh merupakan format dari teksel yang akan digunakan untuk menyimpan tekstur. Parameter kedelapan menentukan sumberdaya memori yang akan digunakan oleh tekstur. Parameter sembilan dan sepuluh merupakan filter tambahan yang akan

diterapkan pada tekstur. Tiga parameter selanjutnya adalah penentu warna *mask* atau warna yang tidak digunakan dan informasi mengenai gambar asli. Parameter terakhir merupakan paramter terpenting yaitu objek COM IDirect3DTexture9 yang akan diposisikan sebagai representasi gambar yang bisa dipanggil pada saat pemerintahan objek devais IDirect3DDevice9.

5.2.10 Implementasi Pemuatan dan Penyimpanan Objek 3D

Pemuatan dan penyimpanan objek 3D bertujuan untuk memuat daftar verteks dan indeks penyusun objek 3D atau model mesh yang berada di dalam file berekstensi .x. Pendefinisian daftar verteks dan indeks secara primitif pada pembuatan model yang sangat kompleks hampir mustahil dilakuan tanpa adanya bantuan dari perangkat lunak yang dapat mengenerasi kumpulan verteks dan indeks.

```
1   //LPD3DXMESH Model->Mesh;
2   //D3DXMATERIAL*Model->Material;
3   //DWORD Model->numMaterials;
4   LPD3DXBUFFER bufMaterial;
5       if(FAILED(D3DXLoadMeshFromX(File
6                                   ,D3DXMESH_SYSTEMMEM
7                                   ,d3ddev
8                                   ,NULL
9                                   ,&bufMaterial
10                                  ,NULL
11                                  ,&Model->numMaterials
12                                  ,&Model->Mesh)))
13      {
14                  sprintf(TextSementara,"Gagal Memuat File Mesh %s",File);
15          MessageBox(NULL, TextSementara, "Error", MB_OK);
16          exit(0);
17      }
18      D3DXMATERIAL* tempMaterials = (D3DXMATERIAL*)bufMaterial-
19  >GetBufferPointer();
20      Model->Material = new D3DMATERIAL9[Model->numMaterials];
21      Model->Texture = new LPDIRECT3DTEXTURE9[Model->numMaterials];
22
23      for(DWORD index = 0; index < Model->numMaterials; index++)
24      {
25          Model->Material[index] = tempMaterials[index].MatD3D;
26          Model->Material[index].Ambient = Model->Material[index].Diffuse;
27          if(FAILED(D3DXCreateTextureFromFile(d3ddev,
28
29  tempMaterials[index].pTextureFilename,
30                                                  &Model->Texture[index])))
31              Model->Texture[index] = NULL;
32      }
33
34
35
36
37
38
```

Kode Implementasi Pemuatan dan Penyimpanan Objek 3D

Sebelum melakukan proses pemuatan model 3D perlu didefinisikan terlebih dahulu sebuah objek COM ID3DXBuffer yang berfungsi sebagai penahan informasi pada saat proses pemuatan. Sama seperti halnya tekstur, DirectX Graphics juga menyediakan fungsi untuk melakukan pemuatan dan penyimpanan model 3D yaitu D3DXLoadMeshFromX(). Parameter parameter yang dibutuhkan oleh fungsi pemuatan adalah pointer bertipe data string sebagai pemanggilan nama file, enum yang menentukan jenis memori yang digunakan untuk penyimpanan, objek COM devais grafis yang digunakan, pointer terhadap objek COM buffer yang akan digunakan untuk menyimpan material, pointer dengan nilai keluaran jumlah material, dan parameter terakhir adalah objek antarmuka COM ID3DXMesh. Objek COM ID3DXMesh berfungsi untuk menyimpan data-data mengenai model 3D dan diposisikan sebagai representasi dari objek 3D yang bisa dipanggil pada saat pemerintahan objek devais IDirect3DDevice9.

Setelah selesai melakukan pemanggilan fungsi D3DXLoadMeshFromX(). Objek antarmuka ID3DXMesh belum mempunyai informasi mengenai material penyusun objek 3D. material adalah atribut-atribut verteks selain koordinat posisi seperti warna dan koordinat tekstur. Sebuah pointer dedefinisikan untuk mengalamati buffer yang menahan data material. Setelah informasi mengenai material kembali didapatkan, pointer yang menunjuk ke tipe data D3DXMATERIAL di-*heap* untuk menunjuk pada tipe data D3DXMATERIAL sejumlah banyak material yang dimiliki oleh model mesh. Proses yang sama pada material juga dilakukan terhadap objek antarmuka IDirect3DTexture9.

Setelah informasi mengenai data-data milik model mesh didapatkan kembali melalui pointer, maka ID3DXMesh diatur parameternya sesuai dengan objek atau model mesh yang dimuat. ID3DXMesh nantinya menjadi representasi objek 3D pada saat melakukan pemrograman.

5.2.11 Implementasi Penentuan Jumlah dan Deteksi Area Menu

Penentuan jumlah dan deteksi area pilihan menu bertujuan untuk menentukan banyaknya jumlah dan mendeteksi area aktif objek sprite yang akan

digunakan oleh *user* sebagai GUI*(Graphical User Interface*) dalam berinteraksi dengan program aplikasi.

```
1   void  TentukanTepiGambar2(OBJECTSPRITE*  go,int  baris,int  kolom,float
2   lebar,float tinggi,int geserx,int gesery,float jedax,float jeday)
3   {
4   int left[50][50],top[50][50],right[50][50],bottom[50][50],menuke[50][50];
5   int tipe=0,jumkolom=kolom,jumbaris=baris;
6   float ukuranlebar=lebar,ukurantinggi=tinggi;
7           go->dalamarea = false;
8           go->tepikursorx = 2000;
9           go->tepikursory = 2000;
10          for (int a=0; a<jumbaris;a++)
11          {
12                  for (int b=0; b<jumkolom;b++)
13                  {
14                  left[a][b] = (tipe%jumkolom)*ukuranlebar;
15                  top[a][b] = (tipe/jumkolom)*ukurantinggi+15;
16                  left[a][b] += (tipe%jumkolom)*jedax;
17                  top[a][b] += (tipe/jumkolom)*jeday;
18                  right[a][b] = left[a][b]+ukuranlebar;
19                  bottom[a][b] = top[a][b]+ukurantinggi;
20                  tipe++;
21                  menuke[a][b] = tipe;
22                  }
23          }
24          for (int c=0; c<jumbaris;c++)
25          {
26                  for (int d=0; d<jumkolom;d++)
27                  {
28                          if  (
29                                  (MousePos.x > (left[c][d]+geserx) )&&
30                                 (MousePos.x < (right[c][d]+geserx) )&&
31                                 (MousePos.y > (top[c][d]+gesery) ) &&
32                                 (MousePos.y < (bottom[c][d]+gesery))
33                                 )
34                                 {
35                                 go->dalamarea = true;
36                                 go->tepikursorx = left[c][d]+geserx;
37                                 go->tepikursory = top[c][d]+gesery;
38                                 go->submenu = menuke[c][d];
39                                 }
40                  }
41          }
42          go->lebarby = ukuranlebar;
43          go->tinggiby= ukurantinggi;
44          go->jumlahmenu = tipe;
45  return ;
46  }
47
```

Kode Implementasi Penentuan Jumlah dan Deteksi Area Menu

Baris ke-4 hingga baris ke-23 pada kode implementasi bertujuan untuk menentukan banyaknya jumlah pilihan menu serta menentukan batas-batas area objek sprite yang akan digambar. Hal ini dilakukan dengan menentukan jumlah baris dan kolom yang akan digunakan untuk melukis objek sprite beserta jarak untuk setiap baris dan kolom. Baris ke-24 hingga baris ke-44 pada kode implementasi bertujuan untuk mendeteksi area aktif yang ditunjuk oleh kursor serta menentukan posisi tepian objek sprite yang akan digambar sebagai GUI.

5.2.12 Implementasi Penggambaran Objek Sprite sebagai Antarmuka

Penggambaran objek sprite sebagai antarmuka bertujuan untuk menggambar objek sprite yang telah ditentukan orientasi tepi gambar serta batas-batas penggambaran untuk digunakan oleh user sebagai antarmuka dalam memberikan masukan kepada program aplikasi.

```
1   RECT part;
2   SetRect(&part,0,0,lebarby,tinggiby );
3   D3DXVECTOR3 center(0.0f, 0.0f, 0.0f);
4   D3DXVECTOR3 position(tepix, tepiy, 0.1f);
5   d3dspt->Draw(pSprite->tex, &part, &center, &position, warna);
```

Kode Implementasi Penggambaran Objek Sprite sebagai Antarmuka

Pada kode implementasi ditunjukkan bahwa objek sprite digambar pada media rektangular yang ditentukan oleh batas atas, bawah, kiri, dan kanan. Kemudian ditentukan orientasi titik tengah objek sprite yang akan digambar. Pada kode implementasi ditentukan orientasi titik tengah berada pada tepi kiri atas seperti halnya ujung kursor. Hal terakhir yang dilakukan adalah menentukan posisi gambar sprite terhadap ujung tepi kiri atas dari layar media penampil.

5.2.13 Implementasi Transformasi Ruang

Transformasi ruang adalah transformasi yang merubah posisi kedudukan verteks pada ruang *world* setelah objek-objek 3D ditransformasi dari koordinat lokal ke koordinat ruang.

```
1   D3DXMATRIX MatSkalasi;
2
3   MatSkalasi._11 = 1.0f; MatSkalasi._12 = 0.0f; MatSkalasi._13 = 0.0f;
4   MatSkalasi._14 = 0.0f;
5   MatSkalasi._21 = 0.0f; MatSkalasi._22 = 1.0f; MatSkalasi._23 = 0.0f;
6   MatSkalasi._24 = 0.0f;
7
8   MatSkalasi._31 = 0.0f; MatSkalasi._32 = 0.0f; MatSkalasi._33 = 1.0f;
9   MatSkalasi._34 = 0.0f;
10  MatSkalasi._41 = 0.0f; MatSkalasi._42 = 0.0f; MatSkalasi._43 = 0.0f;
11  MatSkalasi._44 = 1.0f;
12
13  D3DXMATRIX MatRotasiSumbuY;
14
15  MatRotasiSumbuY._11=  cosf(trao->transformy);
16  MatRotasiSumbuY._12 = 0.0f;
17  MatRotasiSumbuY._13 = -sinf(trao->transformy);
18  MatRotasiSumbuY._14 = 0.0f;
19
20  MatRotasiSumbuY._21 = 0.0f;     MatRotasiSumbuY._22 = 1.0f;
21  MatRotasiSumbuY._23 = 0.0f;MatRotasiSumbuY._24 = 0.0f;
22
23  MatRotasiSumbuY._31 = sinf(trao->transformy);
24  MatRotasiSumbuY._32 = 0.0f;
25  MatRotasiSumbuY._33 = cosf(trao->transformy);
26  MatRotasiSumbuY._34 = 0.0f;
```

```
27
28  MatRotasiSumbuY._41 = 0.0f; MatRotasiSumbuY._42 = 0.0f;
29  MatRotasiSumbuY._43 = 0.0f; MatRotasiSumbuY._44 = 1.0f;
30
31  D3DXMATRIX MatRotasiSumbuX;
32
33  MatRotasiSumbuX._11=  1.0f;   MatRotasiSumbuX._12 = 0.0f;
34  MatRotasiSumbuX._13 = 0.0f; MatRotasiSumbuX._14 = 0.0f;
35
36  MatRotasiSumbuX._21 = 0.0f;
37  MatRotasiSumbuX._22 = cosf(trao->transformx);
38  MatRotasiSumbuX._23 = sinf(trao->transformx);
39  MatRotasiSumbuX._24 = 0.0f;
40
41  MatRotasiSumbuX._31 = 0;
42  MatRotasiSumbuX._32 = -sinf(trao->transformx);
43  MatRotasiSumbuX._33 = cosf(trao->transformx);
44  MatRotasiSumbuX._34 = 0.0f;
45
46  MatRotasiSumbuX._41 = 0.0f; MatRotasiSumbuX._42 = 0.0f;
47  MatRotasiSumbuX._43 = 0.0f; MatRotasiSumbuX._44 = 1.0f;
48  d3ddev->SetTransform(D3DTS_WORLD,
49  &(MatSkalasi*MatRotasiSumbuY*MatRotasiSumbuX));
50
51
```

Kode Implementasi Transformasi Ruang

Pada kode implementasi ditentukan nilai matriks untuk melakukan transformasi geometri translasi dan transformasi geometri rotasi. Ada dua matriks rotasi yang digunakan untuk mentransformasi objek secara rotasi, yaitu rotasi berdasarkan sumbu x dan rotasi berdasarkan sumbu y. Proses transformasi ruang dilakukan dengan rentetan perkalian matriks *(matrix concatenation)*. Proses akhir dari transformasi ruang dilakukan dengan memanggil objek COM IDirect3DDevice9 yang merupakan representasi dari perangkat keras grafis agar melakukan perkalian matriks yang telah didefinisikan terhadap tiap-tiap verteks penyusun objek 3D.

5.2.14 Implementasi Transformasi Pandang

Transformasi pandang atau *view transformation* adalah transformasi yang bertujuan untuk merubah nilai-nilai verteks penyusun sebuah objek 3D yang berada dalam koordinat ruang menjadi mempunyai nilai-nilai di dalam koordinat pandang atau *view space*. Koordinat pandang atau *view space* bisa dianalogikan seperti koordinat kamera virtual karena koordinat pandang bersifat menentukan sudut pandang atau *point of view* relatif terhadap objek-objek yang berada di dalam koordinat ruang.

```
1  D3DXMATRIX MatPandang;
2  D3DXVECTOR3 posisi_kamera = D3DXVECTOR3(a,b,c);
3  D3DXVECTOR3 kamera_lihat = D3DXVECTOR3(x,y,z);
```

```
4   D3DXVECTOR3 up_kamera = D3DXVECTOR3 (0.0f, 1.0f, 0.0f);
5
6   D3DXVECTOR3 orientasi_kamera = -(posisi_kamera - kamera_lihat);
7   D3DXVec3Normalize(&orientasi_kamera,&orientasi_kamera);
8
9   D3DXVECTOR3 kamera_look = orientasi_kamera;
10
11  D3DXVECTOR3 hasil_cross_yz;
12  D3DXVec3Cross(&hasil_cross_yz,&up_kamera,&kamera_look);
13  D3DXVec3Normalize(&hasil_cross_yz,&hasil_cross_yz);
14  D3DXVECTOR3 kamera_right = hasil_cross_yz;
15
16  D3DXVECTOR3 hasil_cross_zx;
17  D3DXVec3Cross(&hasil_cross_zx,&kamera_look,&kamera_right);
18  D3DXVec3Normalize(&hasil_cross_zx,&hasil_cross_zx);
19  D3DXVECTOR3 kamera_up = hasil_cross_zx;
20
21
22  MatPandang._11=  kamera_right.x; MatPandang._12 = kamera_up.x;
23  MatPandang._13 = kamera_look.x; MatPandang._14 = 0.0f;
24
25  MatPandang._21 = kamera_right.y; MatPandang._22 = kamera_up.y;
26  MatPandang._23 = kamera_look.y; MatPandang._24 = 0.0f;
27
28  MatPandang._31 = kamera_right.z; MatPandang._32 = kamera_up.z;
29  MatPandang._33 =       kamera_look.z; MatPandang._34 = 0.0f;
30
31  MatPandang._41 = -D3DXVec3Dot( &posisi_kamera, &kamera_right ) ;
32  MatPandang._42 = -D3DXVec3Dot( &posisi_kamera, &kamera_up);
33  MatPandang._43 = -D3DXVec3Dot( &posisi_kamera, &kamera_look ) ;
34  MatPandang._44 = 1.0f;
35
36  d3ddev->SetTransform(D3DTS_VIEW, &MatPandang);
37
38
39
```

Kode Implementasi Transformasi Pandang

Pada kode implementasi, nilai dari transformasi pandang ditentukan dari posisi koordinat lokal kamera atau mata kamera, arah permukaan bidang datar y atau orientasi tegak milik koordinat lokal kamera atau up kamera, serta kemana arah kamera menghadap*(point of view)*. Transformasi pandang memerlukan nilai *right*, *up*, dan *look* milik koordinat lokal kamera atau arah permukaan bidang datar milik koordinat lokal kamera yang senilai unit vektor $\vec{i}$, $\vec{j}$, dan $\vec{k}$. Tahapan pertama yang dilakukan pada implementasi adalah pencarian nilai *look* koordinat lokal kamera dengan cara mengurangkan posisi koordinat lokal kamera dengan posisi titik atau verteks yang akan ditransformasi nilainya. Hasil dari pengurangan ini kemudian dikalikan negatif karena *look* merupakan orientasi arah pandang kamera terhadap objek dan bukan sebaliknya. Setelah nilai *look* atau unit vektor searah sumbu z milik koordinat lokal kamera diketahui, selanjutnya dicari nilai *right* atau unit vektor searah sumbu x milik koordinat lokal kamera dengan melakukan perkalian *cross product* antara unit vektor $\vec{j}$

milik kamera yang didefinisikan di awal dengan nilai *look* kamera. Hasil dari perkalian cross product tersebut menghasilkan nilai *right* kamera atau unit vektor $\vec{i}$ milik kamera. Setelah unit vektor $\vec{i}$ dan $\vec{j}$ milik koordinat lokal kamera ditemukan, selanjutnya dicari nilai *up* kamera atau unit vektor $\vec{k}$ milik kamera dengan melakukan perkalian cross product antara *look* kamera dengan *right* kamera.

Perkalian dengan cross product dilakukan menurut kaidah tangan kiri karena DirectX Graphics bekerja pada sistem koordinat kartesian menurut kaidah tangan kiri *(left-handed Cartesian coordinate system).* Setelah nilai *right*, *up*, dan *look* milik koordinat lokal kamera ditemukan, selanjutnya dimasukkan ke dalam persamaan matriks yang disebut matriks pandang. Tahap akhir yang dilakukan adalah dengan memanggil objek COM IDirect3DDevice9 yang merupakan representasi dari perangkat keras grafis agar melakukan perkalian matriks yang telah didefinisikan terhadap tiap-tiap verteks penyusun objek 3D.

5.2.15 Implementasi Transformasi Proyeksi Perspektif

Transformasi proyeksi perspektif atau *perspective projection transformation* adalah transformasi yang bertujuan untuk merubah nilai-nilai verteks penyusun sebuah objek 3D yang berada dalam koordinat lokal kamera atau *view space* menjadi koordinat 2D yang disebut koordinat proyeksi atau ruang proyeksi.

```
1   D3DXMATRIX MatProyeksi;
2   float near_plane,far_plane;
3   near_plane = 1.0f;
4   far_plane = 3000.0f;
5
6   MatProyeksi._11= (FLOAT)740/(FLOAT)(ModeTampilanDesktop.Width-412)*(
7   1/cosf(D3DXToRadian(45/2)));
8   MatProyeksi._12 = 0.0f;
9   MatProyeksi._13 = 0.0f;
10  MatProyeksi._14 = 0.0f;
11
12  MatProyeksi._21 = 0.0f;
13  MatProyeksi._22 = 1/cosf(D3DXToRadian(45/2));
14  MatProyeksi._23 = 0.0f;
15  MatProyeksi._24 = 0.0f;
16
17  MatProyeksi._31 = 0.0f;
18  MatProyeksi._32 = 0.0f;
19  MatProyeksi._33 = far_plane/(far_plane-near_plane);
20  MatProyeksi._34 = 1.0f;
21
22  MatProyeksi._41 = 0.0f;
23  MatProyeksi._42 = 0.0f;
24  MatProyeksi._43 = -(far_plane)/(far_plane-near_plane);
25  MatProyeksi._44 = 0.0f;
```

```
26
27  d3ddev->SetTransform(D3DTS_PROJECTION, &MatProyeksi);
28
```

Kode Implementasi Transformasi Proyeksi Perspektif

Transformasi proyeksi perspektif pada implementasi ditentukan dengan besar nilai sudut proyeksi, *near plane*, *far plane*, serta rasio aspek yang digunakan oleh media penampil. Setelah nilai-nilai tersebut ditentukan dan dimasukkan ke dalam persamaan matriks, dilakukan pemanggilan objek COM IDirect3DDevice9 yang merupakan representasi dari perangkat keras grafis agar melakukan perkalian matriks yang telah didefinisikan terhadap tiap-tiap verteks penyusun objek 3D, kemudian menyimpan hasilnya ke dalam memori untuk selanjutnya ditampilkan pada layar media penampil.

5.2.16 Implementasi Manipulasi Tekstur

Manipulasi tekstur diimplementasikan dengan melakukan perubahan terhadap nilai koordinat tekstur. pengubahan nilai koordinat tekstur dapat dilakukan dengan menggunakan perkalian matriks. Elemen matriks m_{11}dan m_{22} merupakan faktor skalasi dari koordinat tekstur, m_{31}dan m_{32} merupakan faktor translasi, dan m_{12}dan m_{21} merupakan faktor kemiringan*(shearing).*

```
1  mat._11 = mapo->b1k1;
2  mat._12 = mapo->b1k2;
3  mat._21 = mapo->b2k1;
4  mat._22 = mapo->b2k2;
5  mat._31 = mapo->b3k1;
6  mat._32 = mapo->b3k2;
7
8  d3ddev->SetTexture(0, Model->Texture);
9  d3ddev->SetTransform ( D3DTS_TEXTURE0 , &mat);
```

Kode Implementasi Manipulasi Tekstur

Setelah nilai matriks pengali ditentukan, program akan meminta IDirect3DDevice9 agar memetakan tekstur tersebut pada objek 3D. Selanjutnya objek devais diminta untuk melakukan perkalian matriks terhadap tiap-tiap verteks penyusun objek 3D.

5.2.17 Implementasi Mulai Render

Proses render atau pelukisan sebuah *frame* pada layar media penampil harus diawali dengan pernyataan konfirmasi agar devais perangkat keras grafis mengunci ruang di memori sebelum dilakukan proses render. Proses penguncian *(locking)* ini memberikan program aplikasi reservasi esklusif terhadap memori video ataupun memori sistem selama dijalankannya operasi render.

```
1   void StartRenderViewportKanan()
2   {
3   D3DVIEWPORT9 rightViewPort;
4   rightViewPort.X      = 412;
5   rightViewPort.Y      = 0;
6   rightViewPort.Width  = (FLOAT)(ModeTampilanDesktop.Width-412);
7   rightViewPort.Height = 740;
8   rightViewPort.MinZ   = 0.0f;
9   rightViewPort.MaxZ   = 1.0f;
10
11  d3ddev->SetViewport( &rightViewPort );
12  d3ddev->Clear(0, NULL, D3DCLEAR_TARGET, D3DCOLOR_XRGB(10, 10, 10), 1.0f,
13  0);
14  d3ddev->Clear(0, NULL, D3DCLEAR_ZBUFFER, D3DCOLOR_XRGB(0, 0, 0), 1.0f, 0);
15  d3ddev->BeginScene();
16      return;
17  }
```

Kode Implementasi Mulai Render

Pada implementasi, awalan render dimulai dengan membagi terlebih dahulu layar media render menjadi dua bagian yang disebut viewport sesuai dengan lebar resolusi dari layar media render. Setelah dilakukan pengaturan viewport, selanjutnya program meminta objek COM IDirect3DDevice9 yang berperan sebagai devais grafis virtual untuk membersihkan *back buffer* yaitu buffer memori berbentuk permukaan *(surface)* yang berperan menyimpan data array sebelum dirender pada *front buffer* atau layar monitor. Method BeginScene() merupakan perintah kepada devais untuk mulai menjalankan proses render.

5.2.18 Implementasi Render Objek Primitif

Penggambaran atau pelukisan objek primitif bertujuan untuk menggambar atau me-*render* objek geometri sederhana, maupun geometri triangular yang telah didefinisikan sebelumnya sesuai dengan daftar urutan verteks, indeks, dan daftar segitiga.

```
1   d3ddev->SetFVF(CUSTOMFVF);
2   d3ddev->SetStreamSource(0, Model->v_buffer, 0, sizeof(CUSTOMVERTEX));
3   d3ddev->SetIndices(Model->i_buffer);
4   d3ddev->SetTexture(0, Model->texturems);
```

```
5  d3ddev->DrawIndexedPrimitive(D3DPT_TRIANGLELIST, 0, 0, 4*(Model->sisi+1), 0,
6  4*Model->sisi);
```

Kode Implementasi Render Objek Primitif

Tahapan pertama pada penggambaran objek primitif adalah pengaturan tipe data verteks. Tipe data verteks adalah penentuan atribut-atribut yang dimiliki oleh verteks. Dengan memerintahkan devais agar menggunakan FVF(Flexible Vertex Format) berarti verteks yang akan digunakan bersifat *custom* atau sesuai dengan kebutuhan pemrogram. Tahapan selanjutnya adalah menentukan verteks buffer dan indeks buffer yang akan dijadikan referensi dalam penggambaran objek primitif. Setelah pemilihan buffer selesai kemudian objek devais diperintahkan untuk melakukan pelukisan atau render objek primitif sesuai dengan daftar segitiga *(triangle list)* yang dimiliki oleh objek primitif.

5.2.19 Implementasi Render Objek Mesh

Penggambaran atau pelukisan objek mesh dilakukan dengan pemanggilan kembali objek COM antarmuka ID3DXMesh yang sebelumnya digunakan dalam pemuatan model yang berasal dari file .x yaitu file yang berisi informasi mengenai daftar verteks dan indeks.

```
1   for(DWORD i = 0; i < numMaterials; i++)
2      {
3       d3ddev->SetMaterial(&material[i]);
4   Model->Material[index].Diffuse.r = go->warnamerah;
5   Model->Material[index].Diffuse.g = go->warnahijau;
6   Model->Material[index].Diffuse.b = go->warnabiru;
7   Model->Material[index].Ambient = Model->Material[index].Diffuse;
8
9           Model->Mesh->DrawSubset(i);
10     }
```

Kode Implementasi Render Objek Mesh

Tahapan awal implementasi pada pelukisan model mesh 3D adalah perhitungan jumlah material yang dimiliki oleh mesh. Material merupakan elemen milik mesh yang mempunyai masing-masing daftar verteks dan indeks secara terpisah dengan elemen mesh yang lain. Setelah didapatkan jumlah material, objek COM ID3DXMesh diperintahkan untuk menggambar tiap-tiap *subset* atau bagian-bagian mesh sesuai dengan jumlah material. Nilai *diffuse* dan *ambient* pada material merupakan warna dari verteks yang terpengaruh refleksi warna

cahaya. Pada implementasi, warna *diffuse* dan ambient digunakan sebagai manipulasi warna terhadap objek pakaian.

5.2.20 Implementasi Akhiri Render

Implementasi akhir render digunakan sebagai penanda bahwa proses render untuk satu kali frame sudah selesai dilakukan.

```
1  void EndRenderViewportKanan()
2  {
3      d3ddev->EndScene();
4      d3ddev->Present(NULL, NULL, NULL, NULL);
5      return;
6  }
```

Kode Implementasi Akhiri Render

Pada implementasi untuk akhir dari proses render, devais perangkat keras grafis diberikan perintah untuk membebaskan memori yang sebelumnya direservasi pada tahap awal pelukisan atau render. Baris kedua merupakan perintah untuk membuat pertukaran *(swap) multiple frame buffer* maupun *multiple window* jika nantinya ada dan dibutuhkan.

5.2.21 Implementasi Pemodelan Kubus

Pemodelan kubus secara primitif bertujuan untuk menerapkan salah satu metode pengalamatan tekstur yaitu metode pengalamatan planar agar nanti hasilnya sesuai dan dapat diamati. Pemodelan kubus diposisikan di titik tengah koordinat 3D supaya memudahkan dalam pemodelan.

```
1  struct CUSTOMVERTEX {FLOAT X, Y, Z; D3DVECTOR NORMAL; FLOAT U, V;};
2  CUSTOMVERTEX vertices[] =
3      {
4          { -3.0f, -3.0f, 3.0f,  0.0f, 0.0f, 1.0f,  1.0f, 1.0f, },
5          { 3.0f, -3.0f, 3.0f,  0.0f, 0.0f, 1.0f,  0.0f, 1.0f, },
6          { -3.0f, 3.0f, 3.0f,  0.0f, 0.0f, 1.0f,  1.0f, 0.0f, },
7          { 3.0f, 3.0f, 3.0f,  0.0f, 0.0f, 1.0f,  0.0f, 0.0f, },
8
9          { -3.0f, -3.0f, -3.0f,  0.0f, 0.0f, -1.0f,  0.0f, 1.0f, },
10         { -3.0f, 3.0f, -3.0f,  0.0f, 0.0f, -1.0f,  0.0f, 0.0f, },
11         { 3.0f, -3.0f, -3.0f,  0.0f, 0.0f, -1.0f,  1.0f, 1.0f, },
12         { 3.0f, 3.0f, -3.0f,  0.0f, 0.0f, -1.0f,  1.0f, 0.0f, },
13
14         { -3.0f, 3.0f, -3.0f,  0.0f, 1.0f, 0.0f,  1.0f, 0.0f, },    //
15 atas
16         { -3.0f, 3.0f, 3.0f,  0.0f, 1.0f, 0.0f,  1.0f, 1.0f, },
17         { 3.0f, 3.0f, -3.0f,  0.0f, 1.0f, 0.0f,  0.0f, 0.0f, },
18         { 3.0f, 3.0f, 3.0f,  0.0f, 1.0f, 0.0f,  0.0f, 1.0f, },
19
20         { -3.0f, -3.0f, -3.0f,  0.0f, -1.0f, 0.0f,  0.0f, 0.0f, },    //
21 bawah
22         { 3.0f, -3.0f, -3.0f,  0.0f, -1.0f, 0.0f,  1.0f, 0.0f, },
23         { -3.0f, -3.0f, 3.0f,  0.0f, -1.0f, 0.0f,  0.0f, 1.0f, },
```

```
24          { 3.0f, -3.0f, 3.0f,  0.0f, -1.0f, 0.0f,  1.0f, 1.0f, },
25
26          { 3.0f, -3.0f, -3.0f,  1.0f, 0.0f, 0.0f,  0.0f, 1.0f, },
27          { 3.0f, 3.0f, -3.0f,  1.0f, 0.0f, 0.0f,  0.0f, 0.0f, },
28          { 3.0f, -3.0f, 3.0f,  1.0f, 0.0f, 0.0f,  1.0f, 1.0f, },
29          { 3.0f, 3.0f, 3.0f,  1.0f, 0.0f, 0.0f,  1.0f, 0.0f, },
30
31          { -3.0f, -3.0f, -3.0f,  -1.0f, 0.0f, 0.0f,  1.0f, 1.0f, },
32          { -3.0f, -3.0f, 3.0f,  -1.0f, 0.0f, 0.0f,  0.0f, 1.0f, },
33          { -3.0f, 3.0f, -3.0f,  -1.0f, 0.0f, 0.0f,  1.0f, 0.0f, },
34          { -3.0f, 3.0f, 3.0f,  -1.0f, 0.0f, 0.0f,  0.0f, 0.0f, },
35      };
36
37      d3ddev->CreateVertexBuffer(24*sizeof(CUSTOMVERTEX),
38                                 0,
39                                 CUSTOMFVF,
40                                 D3DPOOL_MANAGED,
41                                 &v_buffer,
42                                 NULL);
43      VOID* pVoid;
44
45      v_buffer->Lock(0, 0, (void**)&pVoid, 0);
46      memcpy(pVoid, vertices, sizeof(vertices));
47      v_buffer->Unlock();
48      short indices[] =
49      {
50          0, 1, 2,    // sisi 1
51          2, 1, 3,
52          4, 5, 6,    // sisi 2
53          6, 5, 7,
54          8, 9, 10,   // sisi 3
55          10, 9, 11,
56          12, 13, 14, // sisi 4
57          14, 13, 15,
58          16, 17, 18, // sisi 5
59          18, 17, 19,
60          20, 21, 22, // sisi 6
61          22, 21, 23,
62      };
63
64      d3ddev->CreateIndexBuffer(36*sizeof(short),
65                                0,
66                                D3DFMT_INDEX16,
67                                D3DPOOL_MANAGED,
68                                &i_buffer,
69                                NULL);
70
71      i_buffer->Lock(0, 0, (void**)&pVoid, 0);
72      memcpy(pVoid, indices, sizeof(indices));
73      i_buffer->Unlock();
```

Kode Implementasi Pemodelan Kubus

Hal pertama yang dilakukan dalam pemodelan kubus adalah menentukan tipe data verteks yang akan digunakan dalam pendefinisian. Pada implementasi, penulis menggunakan atribut-atribut berupa posisi verteks yang dinyatakan dengan tipe data float, normal vektor yang dinyatakan dengan tipe data objek D3DVECTOR, dan koordinat tekstur yang dinyatakan dengan tipe data float. Tipe data objek D3DVECTOR merupakan struct yang berisikan tiga bilangan float x,y,z dengan kelebihan pada operator overloading.

Pada pemodelan kubus digunakan jumlah verteks sebanyak 24 buah, 6 buah untuk tiap-tiap sisi pada kubus. Pendefisian verteks-verteks dalam pemodelan kubus membutuhkan sebuah objek COM DirectX yaitu IDirect3DVertexBuffer yang berfungsi untuk menyimpan data verteks pada memori sistem atau memori video RAM. Setelah verteks-verteks selesai didefinisi, verteks buffer perlu melakukan penguncian *(lock)* terhadap ruang memori agar data verteks dapat dimasukkan ke dalam verteks buffer. Setelah perintah memcpy digunakan untuk menyimpan data verteks ke dalam verteks buffer, verteks buffer bisa melepas penguncian pada memori agar bisa diakses oleh program atau instruksi yang lain.

Pemodelan kubus selain menggunakan verteks juga menggunakan indeks. Indeks digunakan untuk menyatakan urut-urutan verteks yang menyusun objek kubus. Aturan pengurutan data verteks harus sesuai dengan aturan *winding order* yaitu searah jarum jam untuk bisa merender depan permukaan. Selain mempermudah pemodelan, indeks juga berfungsi dalam penghematan array. Sama seperti verteks, indeks juga memerlukan objek antarmuka COM sebagai penyimpan data indeks. Proses penyimpanan data indeks ke dalam indeks buffer serupa dengan proses lock unlock pada verteks buffer.

5.2.22 Implementasi Pemodelan Bola

Pemodelan bola secara primitif bertujuan untuk menerapkan salah satu metode pengalamatan tekstur yaitu metode pengalamatan spheris agar nanti hasilnya sesuai dan dapat diamati. Pemodelan bola diposisikan di titik tengah koordinat 3D supaya memudahkan dalam pemodelan.

```
1   float sudut;
2   sudut = 2*D3DX_PI  /sisi;
3   float nilaiY,nilaiY2;
4   float Rhor,Rhor2;
5
6   int jumv = (sisi+1); int jumvb = 0;
7
8   CUSTOMVERTEX vertices[5000] ;
9
10      for (int a = 0; a < (sisi/2); a++)
11          {
12                  nilaiY = (float)cos( a*sudut ) * jari;
13                  Rhor = (float)sin( a*sudut )*jari;
14                  nilaiY2 = (float)cos( (a+1)*sudut ) * jari;
15                  Rhor2 = (float)sin( (a+1)*sudut )*jari;
16
17                  for (int i = 0; i <= sisi; i++)
18                          {
19                          vertices[jumvb+i].Y = nilaiY;
```

```
20                          vertices[jumvb+i].X = (float)cos( i*sudut ) * Rhor;
21                          vertices[jumvb+i].Z = (float)sin( i*sudut ) * Rhor;
22                          }
23                  for (int i = 0; i <= sisi; i++)
24                          {
25                          vertices[jumvb+i+jumv].Y = nilaiY2;
26                          vertices[jumvb+i+jumv].X = (float)cos( i*sudut ) *
27  Rhor2;
28                          vertices[jumvb+i+jumv].Z = (float)sin( i*sudut ) *
29  Rhor2;
30                          }
31
32                  jumvb = jumvb +((sisi+1)*2);
33          }
34
35  ..................
36
37          short indices[5000];
38          int n;
39          int juminb;
40          int nilaib = (sisi+1)*2;
41          int q;
42      for (int a = 0; a < (sisi/2); a++)
43          {
44            juminb = a*(6*sisi);
45            q = a*nilaib;
46                  for (int i = 0; i < sisi; i++)
47                  {
48                  n = (i*6)+juminb;
49                  indices[n]= i+q;
50                  indices[n+1] = i+1+q;
51                  indices[n+2] = i+sisi+1+q;
52
53                  indices[n+3]= i+1+q;
54                  indices[n+4] = i+sisi+2+q;
55                  indices[n+5] = i+sisi+1+q;
56                  }
57          }
```

Kode Implementasi Pemodelan Bola

Pemodelan bola dilakukan dengan membuat daftar verteks dan indeks yang berurutan sesuai dengan konsep lembaran atau bidang datar. Urut-urutan verteks yang berada pada bagian paling atas dan bagian paling nantinya diposisikan berhimpit pada satu titik yang disebut titik zenith dan titik nadir.

Pada pemodelan bola ditentukan dulu jumlah sisi yang akan menyusun objek bola. Banyak jumlah sisi pada bola berpengaruh pada nilai kebulatan *(round value).* Semakin banyak sisi yang digunakan semakin bulat bola, semakin sedikit sisi yang digunakan semakin tersegmentasi sebuah objek bola.

Cara pengurutan indeks pada objek bola yang digunakan adalah dengan mencari nilai titik di posisi y kemudian mencari jari-jari untuk setiap tinggi y. Nilai y dapat dicari dengan menggunakan persamaan koordinat silinder yaitu $y = rcos\varphi$, sedangkan untuk jari-jari di setiap sumbu $y = n$ diperoleh dengan persamaan $r_{hor} = rsin\varphi$. Pendaftaran indeks dilakukan setiap nilai y dan nilai

$y + 1$ agar daftar segitiga-segitiga*(triangle list)* bisa saling terkoneksi. Prosedur penyimpanan data verteks dan indeks pada pemodelan bola serupa dengan prosedur penyimpanan data verteks dan indeks pada pemodelan kubus.

5.2.23 Implementasi Pemodelan Silinder

Pemodelan silinder secara primitif bertujuan untuk menerapkan salah satu metode pengalamatan tekstur yaitu metode pengalamatan silindris agar nanti hasilnya sesuai dan dapat diamati. Pemodelan silinder diposisikan di titik tengah koordinat 3D supaya memudahkan dalam pemodelan.

```
1   float sudut;
2   sudut = 2*D3DX_PI  /sisi;
3   sisi=36;
4   CUSTOMVERTEX vertices[2000] ;
5       for (int i = 0; i < sisi; i++)
6           {
7           vertices[i].X = (float)cos( i*sudut ) * jari;
8                   vertices[i].Y = tinggi;
9                   vertices[i].Z= (float)sin( i*sudut ) * jari;
10                  vertices[i].NORMAL = D3DXVECTOR3(0.0f, 1.0f, 0.0f);
11          }
12          vertices[sisi].X = 0.0f;
13          vertices[sisi].Y= tinggi;
14                  vertices[sisi].Z = 0.0f;
15                  vertices[sisi].NORMAL = D3DXVECTOR3(0.0f, 1.0f, 0.0f);
16
17          int balik;
18      for (int c = (sisi+1); c < ((2*sisi)+1); c++)
19          {
20                  balik = c-(sisi+1);
21          vertices[c].X = (float)cos( balik*sudut ) * jari;
22                  vertices[c].Y = -tinggi;
23          vertices[c].Z= (float)sin( balik*sudut ) * jari;
24
25                  vertices[c].NORMAL = D3DXVECTOR3(0.0f, -1.0f, 0.0f);
26          }
27
28          vertices[((2*sisi)+1)].X = 0.0f;
29                  vertices[((2*sisi)+1)].Y = -tinggi;
30          vertices[((2*sisi)+1)].Z= 0.0f;
31
32                  vertices[((2*sisi)+1)].NORMAL = D3DXVECTOR3(0.0f, -1.0f,
33  0.0f);
34
35  ////////////////////////////////////////////////////
36
37      for (int c = ((2*sisi)+2); c < ((3*sisi)+2); c++)
38          {
39                  balik = c-((2*sisi)+2);
40          vertices[c].X = (float)cos( balik*sudut ) * jari;
41                  vertices[c].Y = tinggi;
42          vertices[c].Z= (float)sin( balik*sudut ) * jari;
43
44                  D3DXVECTOR3 veknom;
45                  veknom = D3DXVECTOR3(vertices[c].X,0.0f, vertices[c].Z);
46
47                  D3DXVec3Normalize(&veknom,&veknom);
48                  vertices[c].NORMAL = veknom;
49          }
50          vertices[((3*sisi)+2)].X = 0.0f;
51          vertices[((3*sisi)+2)].Y= tinggi;;
52                  vertices[((3*sisi)+2)].Z = 0.0f;
53                  vertices[((3*sisi)+2)].NORMAL = D3DXVECTOR3(0.0f, 0.0f,
```

```
54  1.0f);
55
56          for (int c = ((3*sisi)+3); c < ((4*sisi)+3); c++)
57            {
58                  balik = c-((3*sisi)+3);
59           vertices[c].X = (float)cos( balik*sudut ) * jari;
60                  vertices[c].Y = -tinggi;
61           vertices[c].Z= (float)sin( balik*sudut ) * jari;
62
63                  D3DXVECTOR3 veknom2;
64                  veknom2 = D3DXVECTOR3(vertices[c].X,0.0f, vertices[c].Z);
65                  D3DXVec3Normalize(&veknom2,&veknom2);
66                  vertices[c].NORMAL = veknom2;
67          }
68           vertices[((4*sisi)+3)].X = 0.0f;
69                  vertices[((4*sisi)+3)].Y = -tinggi;
70           vertices[((4*sisi)+3)].Z= 0.0f;
71          vertices[((4*sisi)+3)].NORMAL = D3DXVECTOR3(0.0f, 0.0f, -1.0f);
72
73          short indices[1000];
74          int n;
75      for (int i = 0; i < sisi; i++)
76          {
77                  if(i==(sisi-1))
78                  {
79                  n = i*3;
80                  indices[n]= i;
81                  indices[n+1] = sisi;
82                  indices[n+2] = 0;
83                  }
84                  else
85                  {
86                  n = i*3;
87                  indices[n]= i;
88                  indices[n+1] = sisi;
89                  indices[n+2] = i+1;
90                  }
91          }
92          int  p = sisi;
93          int q = sisi +1;
94      for (int i = 0; i < sisi; i++)
95          {
96                  if(i==(sisi-1))
97                  {
98                  n = p*3;
99                  indices[n]= q;
100                 indices[n+1] = sisi+1;
101                 indices[n+2] = ((2*sisi)+1);
102                 }
103                 else
104                 {
105                 n = p*3;
106                 indices[n]= q;
107                 indices[n+1] = q+1;
108                 indices[n+2] = ((2*sisi)+1);
109                 }
110                 p++;
111                 q++;
112         }
113         p = 2*sisi;
114         q = 0;
115     for (int i = 0; i < sisi; i++)
116         {
117         n = (p*3) + (3*i);
118         if(i==(sisi-1))
119         {
120         indices[n]= 2*sisi+2+q;
121         indices[n+1] = 3*sisi+3;
122         indices[n+2] = 3*sisi+3+q;
123         indices[n+3]= 2*sisi+2;
124         indices[n+4] = 3*sisi+3;
125         indices[n+5] = 2*sisi+2+q;
126         }
127         else
```

```
128         {
129         indices[n]= 2*sisi+2+q;
130         indices[n+1] = 3*sisi+4+q;
131         indices[n+2] = 3*sisi+3+q;
132         indices[n+3]= 2*sisi+3+q;
133         indices[n+4] = 3*sisi+4+q;
134         indices[n+5] = 2*sisi+2+q;
135         }
136         p++;
137         q++;
138
139 }
```

Kode Implementasi Pemodelan Silinder

Pada implementasi pemodelan silinder, dua bagian penutup didefinisikan terlebih dahulu daftar verteks-verteksnya untuk masing-masing penutup. Pendaftaran verteks-verteks pada bagian penutup dicari dengan menetapkan satu titik tengah sebagai penghubung segitiga kemudian bagian melingkar yang didapatkan dengan persamaan $x = rcos\varphi$ dan $z = rsin\varphi$. Setelah selesai dibuat bagian penutup, dibuat lagi daftar indeks dengan metode yang sama seperti bagian penutup hanya saja kali ini digunakan sebagai penghubung badan silinder. Persamaan pendaftaran indeks ke dalam array didapatkan dengan metode empiris yaitu dengan memberikan nilai urut-urutan verteks yang sifatnya tidak terlalu banyak kemudian mengamati pola urut-urutan agar dapat dicari persmaan untuk daftar indeks ke dalam array.

5.2.24 Implementasi Metode Pengalamatan Planar

Metode pengalamatan planar digunakan untuk mendistribusikan nilai-nilai koordinat tekstur secara linier atau sebanding dengan bidang datar.

```
1   int numVerts=texMesh->GetNumVertices();
2   float fMinY,fMaxY,fTengahY;
3   fMinY = pVerts->pos.y;
4   fMaxY = pVerts->pos.y;
5
6    for (int i=0;i<numVerts;i++)
7    {
8           if(pVerts->pos.y < fMinY)
9                   fMinY = pVerts->pos.y;
10          if(pVerts->pos.y > fMaxY)
11                  fMaxY = pVerts->pos.y;
12          pVerts++;
13
14   }
15   pVerts -= numVerts;
16    fTengahY = (fMaxY+fMinY)/2;
17
18  float fMinX,fMaxX,fTengahX;
19  fMinX = pVerts->pos.x;
20  fMaxX = pVerts->pos.x;
21
22   for (int i=0;i<numVerts;i++)
```

```
 {
        if(pVerts->pos.x < fMinX)
                fMinX = pVerts->pos.x;
        if(pVerts->pos.x > fMaxX)
                fMaxX = pVerts->pos.x;
        pVerts++;
 }
 pVerts -= numVerts;
  fTengahX = (fMaxX+fMinX)/2;

float fMinZ,fMaxZ,fTengahZ;
fMinZ = pVerts->pos.z;
fMaxZ = pVerts->pos.z;
 for (int i=0;i<numVerts;i++)
 {
        if(pVerts->pos.z < fMinZ)
                fMinZ = pVerts->pos.z;
        if(pVerts->pos.z > fMaxZ)
                fMaxZ = pVerts->pos.z;
        pVerts++;
 }
 pVerts -= numVerts;
  fTengahZ = (fMaxZ+fMinZ)/2;
D3DXVECTOR3 vMin,vMax,vTengah;
vMin = pVerts->pos;
vMax = pVerts->pos;
 for (int i=0;i<numVerts;i++)
 {
        if(pVerts->pos < vMin)
                vMin = pVerts->pos;
        if(pVerts->pos > vMax)
                vMax = pVerts->pos;
        pVerts++;
 }
 pVerts -= numVerts;
vTengah = D3DXVECTOR3(fTengahX,fTengahY,fTengahZ);
for (int i=0;i<numVerts;i++) {
        pVerts->tv = -(pVerts->pos.y-fTengahY)/(fMaxY-fMinY+epsilon) +
0.5f;

        pVerts->tu =  (pVerts->pos.x-fTengahX)/(fMaxX-fMinX+epsilon) +
0.5f;

            pVerts++;
         }
```

Kode Implementasi Metode Pengalamatan Planar

Tahapan awal proses pendistribusian alamat koordinat tekstur adalah dengan mencari nilai posisi verteks yang berada di tengah-tengah objek. Pencarian nilai tengah didapatkan dengan enumerasi posisi semua verteks-verteks sehingga didapatkan nilai x maks, x min, y maks, y min, z maks, dan z min yang menjadi penentu nilai titik tengah. Setelah didapatkan nilai titik tengah pada objek mesh 3D kemudian dimasukkan persamaan yang didapat di dalam tahap perancangan yaitu

$$koordinat\ tekstur\ u = \frac{x}{x\ max\ -\ x\ min} + 0{,}5$$

$$koordinat\ tekstur\ v = -\frac{y}{y\ max\ -\ y\ min} + 0{,}5$$

5.2.25 Implementasi Metode Pengalamatan Spheris berdasarkan Persamaan Koordinat Bola dan Sudut Hasil Dot Product

Metode pengalamatan spheris digunakan untuk mendistribusikan nilai-nilai koordinat tekstur yang berbanding lurus dengan sudut φ dan θ.

```
for (int i=0;i<numVerts;i++) {
                D3DXVECTOR3 vJarak = pVerts->pos - vTengah;
                D3DXVECTOR3 vJarakxz = D3DXVECTOR3(vJarak.x,0.0f,vJarak.z);
                D3DXVec3Normalize(&vJarakxz,&vJarakxz);
                D3DXVECTOR3 normalacuan = D3DXVECTOR3(1.0f,0.0f, 0.0);
                float dotprod = D3DXVec3Dot( &normalacuan, &vJarakxz ) ;
                float sudutuv = acosf(dotprod);
                float uv = sudutuv/(2*D3DX_PI);

                pVerts->tv = acosf(vJarak.y)/D3DX_PI ;
                if(vJarak.z>=0)
                        pVerts->tu =  uv;
                else
                        pVerts->tu =  1.0f-uv;
          pVerts++;
        }
```

Kode Implementasi Metode Pengalamatan Spheris Berdasarkan Persamaan Koordinat Bola dan Sudut Hasil Dot Product

Nilai koordinat tekstur v dicari dengan menggunakan persamaan koordinat bola sedangkan nila koordinat tekstur u dicari dengan menggunakan sudut dot product antara posisi verteks setelah dikurangi elemen y dengan unit vektor searah sumbu x yaitu $\vec{\imath}$. Supaya tidak terjadi duplikasi koordinat tekstur u pada verteks-verteks yang berada pada kuadran III dan IV maka nilai-nilai u diperoleh dengan pengurangan skala unit koordinat tekstur u. Penentuan posisi kuadran ditentukan dengan nilai normal z. Normal z merupakan arah posisi verteks dari titik pusat setelah dilakukan normalisasi. Setelah didapatkan nilai titik tengah pada objek mesh 3D kemudian dimasukkan persamaan yang didapat di dalam tahap perancangan yaitu

$$koordinat\ tekstur\ v = \frac{arcos\left({}^{y}/_{r}\right)}{\pi}$$

$$koordinat\ tekstur\ u = \frac{\arccos\left(\mathrm{P}\cdot\vec{\imath}\right)}{2\pi}$$

Jika $N_z < 0$

$$koordinat\ tekstur\ u = 1 - \frac{\arccos\left(\mathrm{P}\cdot\vec{\imath}\right)}{2\pi}$$

5.2.26 Implementasi Metode Pengalamatan Spheris berdasarkan Normal Vektor

Metode pengalamatan spheris berdasarkan normal vektor digunakan untuk mendistribusikan nilai-nilai koordinat tekstur yang berbanding lurus dengan sudut φ dan θ dengan nilai sudut φ dan θ berkisar antara 0 hingga π.

```
1   for (int i=0;i<numVerts;i++)
2   {
3         D3DXVECTOR3 vJarak = pVerts->pos - vTengah;
4
5    D3DXVec3Normalize(&vJarak,&vJarak);
6         pVerts->tv =  -(asinf(vJarak.y)/D3DX_PI)+0.5 ;
7         pVerts->tu =  (asinf(vJarak.x)/D3DX_PI)+0.5 ;
8
9    pVerts++;
10         }
```

Kode Implementasi Metode Pengalamatan Spheris Berdasarkan Normal Vektor

Karena nilai normal vektor yang bersifat linier terhadap tiap-tiap sudut maka nilai-nilai koordinat tekstur u dan v dapat dicari dengan menggunakan orientasi terhadap titik tengah sesuai dengan persamaan yang didapatkan pada perancangan yaitu

$$koordinat\ tekstur\ u = \frac{\text{asin}\ (N_x)}{\pi} + 0{,}5$$

$$koordinat\ tekstur\ v = -\frac{\text{asin}\ (N_y)}{\pi} + 0{,}5$$

5.2.27 Implementasi Metode Pengalamatan Silindris berdasarkan Persamaan Koordinat Silinder

Metode pengalamatan silindris digunakan untuk memberikan koordinat tekstur u yang sebanding dengan sudut φ sedangkan untuk koordinat tekstur v didapatkan secara linier atau sebanding dengan sumbu y.

```
1   for (int i=0;i<numVerts;i++) {
2
3   D3DXVECTOR3 vJarak = pVerts->pos - vTengah;
4   D3DXVECTOR3 vNormalxz = D3DXVECTOR3(vJarak.x,0.0f,vJarak.z);
5   D3DXVec3Normalize(&vNormalxz,&vNormalxz);
6
7   D3DXVECTOR3 vCariJari = pVerts->pos - vTengah;
8   float jarikuadrat= (pVerts->pos.x*pVerts->pos.x)+(pVerts->pos.z*pVerts-
9   >pos.z);
10  float jari= sqrtf(jarikuadrat);
11
12  pVerts->tv = -(pVerts->pos.y-fTengahY)/(fMaxY-fMinY) + 0.5f;
13
14  if(vNormalxz.z >=0)
15  pVerts->tu = pVerts->tu = acosf(pVerts->pos.x/jari) / (2*D3DX_PI);
16  else
```

```
17    pVerts->tu = pVerts->tu = 1.0f-acosf(pVerts->pos.x/jari) / (2*D3DX_PI);
18        pVerts++;
19    }
```

Kode Implementasi Metode Pengalamatan Silindris berdasarkan Persamaan Koordinat Silinder

Kode implementasi pada pengalamatan silindris didapat dari persamaan koordinat silinder yaitu $x = rcos\varphi$ untuk nilai koordinat tekstur u, sedangkan untuk koordinat tekstur v didapatkan secara linier sebanding dengan tinggi silinder. Setelah didapatkan nilai titik tengah pada objek mesh 3D kemudian dimasukkan persamaan yang didapat di dalam tahap perancangan yaitu

$$koordinat\ tekstur\ v = -\frac{y}{y\ max\ -\ y\ min} + 0{,}5$$

$$koordinat\ tekstur\ u = \frac{arcos\left(\frac{x}{r}\right)}{2\pi}$$

Jika $N_z < 0$

$$koordinat\ tekstur\ u = 1 - \frac{arcos\left(\frac{x}{r}\right)}{2\pi}$$

5.2.28 Implementasi Metode Pengalamatan Silindris berdasarkan Normal Vektor

Metode pengalamatan silindris berdasarkan normal vektor digunakan untuk mendistribusikan nilai-nilai koordinat tekstur yang berbanding lurus dengan sudut φ yang berkisar antara 0 hingga π.

```
1     for (int i=0;i<numVerts;i++)
2     {
3       D3DXVECTOR3 vJarak = pVerts->pos - vTengah;
4
5       D3DXVec3Normalize(&vJarak,&vJarak);
6
7            pVerts->tv = -(pVerts->pos.y-fTengahY)/(fMaxY-fMinY) + 0.5f;
8            pVerts->tu =  asinf(vJarak.x)/D3DX_PI+ 0.5 ;
9           pVerts++;
10        }
```

Kode Implementasi Metode Pengalamatan Silindris berdasarkan Normal Vektor

Karena nilai normal vektor x yang bersifat linier terhadap sudut φ maka nilai-nilai koordinat tekstur u dicari dengan menggunakan orientasi terhadap titik tengah sesuai dengan persamaan yang didapatkan pada perancangan yaitu

$$koordinat\ tekstur\ u = \frac{\text{asin}\ (N_x)}{\pi} + 0{,}5$$

Sedangkan untuk koordinat tekstur v didapatkan secara linier sebanding dengan tinggi silinder

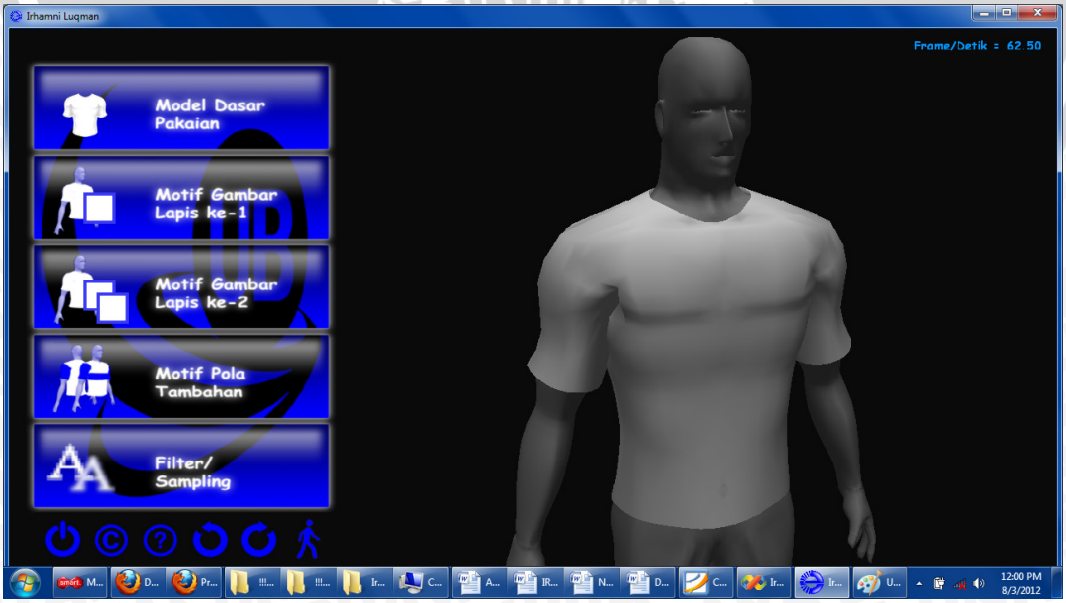
$$koordinat\ tekstur\ v = -\frac{y}{y\ max\ -\ y\ min} + 0{,}5$$

5.3 Implementasi Tampilan Antarmuka Program Penampil Motif Pakaian

Implementasi tampilan antarmuka yang akan digunakan oleh *user* dalam melakukan kontrol atau berinteraksi dengan program aplikasi diantaranya adalah sebagai berikut:

5.3.1 Tampilan Antarmuka Menu Utama

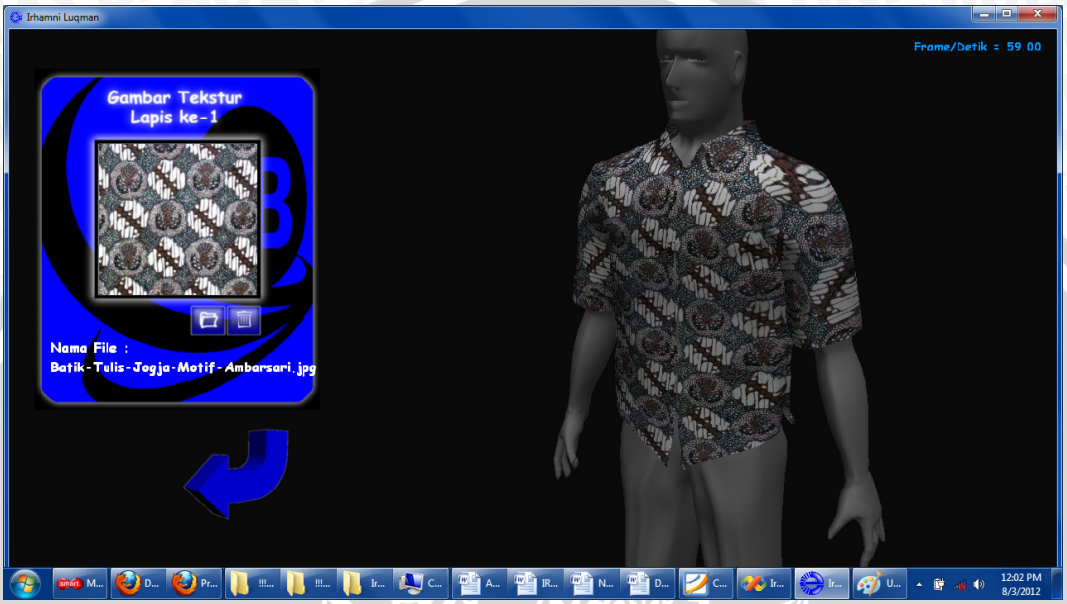
Menu utama adalah tampilan awal dijalankannya aplikasi setelah program berhasil melakukan tahapan-tahapan inisialisasi. Menu utama berisikan menu untuk memilih model dasar pakaian, menu untuk memilih motif pakaian lapis pertama, menu untuk memilih motif pakaian lapis kedua, menu untuk memilih pola tambahan, dan menu untuk memilih tipe filter yang akan digunakan untuk pengambilan gambar. Pada bagian bawah pilihan menu merupakan pilihan untuk merotasi objek secara berkala dan menjalankan animasi model berjalan serta pilihan untuk keluar dari aplikasi.



Gambar 5.1 Tampilan Antarmuka Menu Utama

5.3.2 Tampilan Antarmuka Pengambilan Gambar

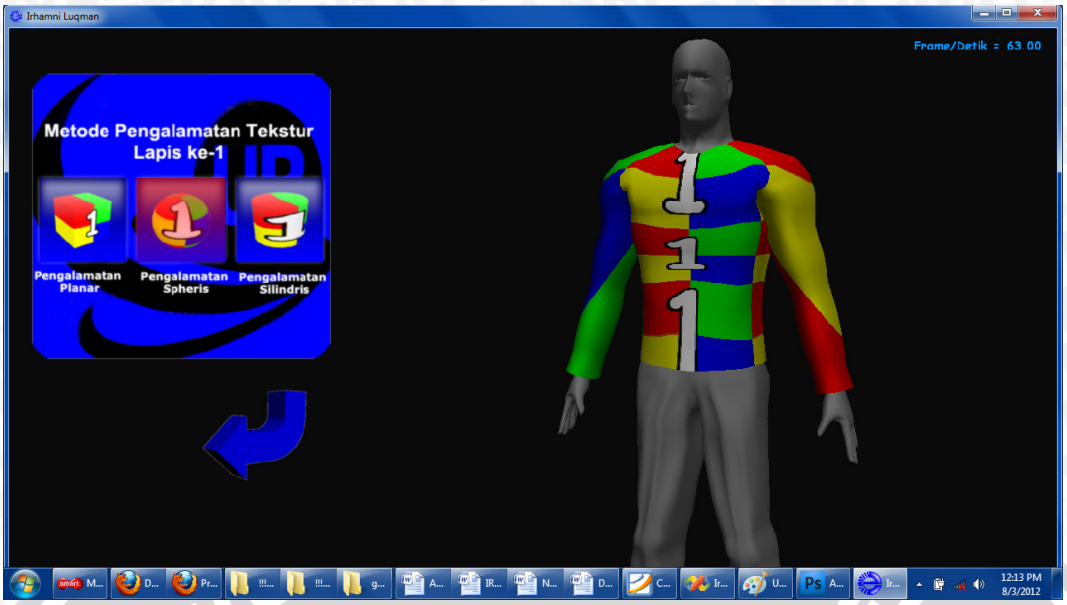
Menu pengambilan gambar digunakan untuk memuat gambar tekstur dari direktori file yang ada di dalam sistem. Menu pengambilan gambar dilengkapi dengan Open Dialog Box untuk mempermudah pengambilan gambar serta pilihan untuk menghapus gambar tekstur.



Gambar 5.2 Tampilan Antarmuka Pengambilan Gambar

5.3.3 Tampilan Antarmuka Metode Pengalamatan

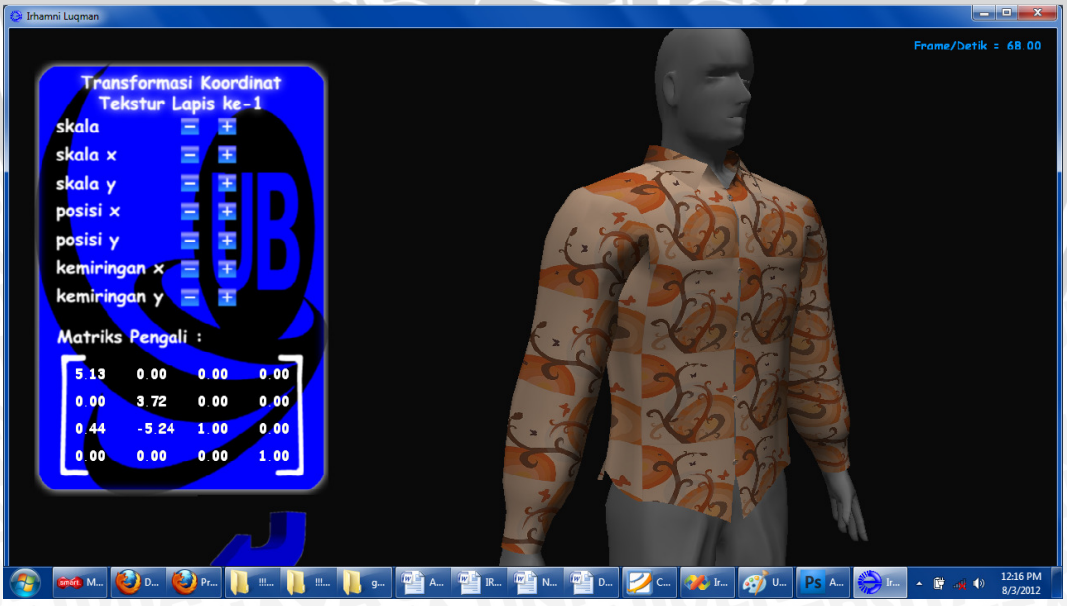
Menu pilihan metode pengalamatan digunakan untuk menampilkan model yang telah diberi koordinat tekstur sesuai dengan masing-masing metode pengalamatan.



Gambar 5.3 Tampilan Antarmuka Pilih Metode Pengalamatan

5.3.4 Tampilan Antarmuka Manipulasi Koordinat Tekstur

Menu manipulasi koordinat tekstur digunakan untuk mengubah nilai koordinat tekstur yang dimiliki model sesuai dengan nilai pengubah yang direpresentasikan dengan matriks pengali berdimensi 4×4.



Gambar 5.4 Tampilan Antarmuka Manipulasi Koordinat Tekstur

BAB VI
PENGUJIAN DAN ANALISIS

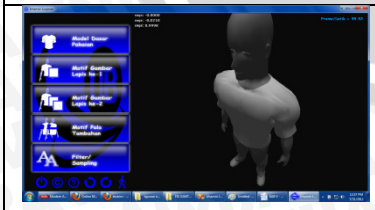
Untuk mengetahui apakah sistem bekerja dengan baik dan sesuai dengan perancangan, maka diperlukan serangkain pengujian. Pengujian yang dilakukan dalam bab ini adalah sebagai berikut:

a. Pengujian Subjektif Hasil Proyeksi
b. Pengujian Objektif Hasil Proyeksi
c. Pengujian Subjektif Metode Pengalamatan
d. Pengujian Objektif Metode Pengalamatan
e. Pengujian Manipulasi Koordinat Tekstur

6.1 Pengujian Subjektif Hasil Proyeksi

Pengujian subjektif hasil proyeksi dilakukan dengan melakukan perbandingan antara hasil proyeksi yang dilakukan terhadap objek 3D dengan perkalian matriks pandang dan matriks proyeksi dibandingkan dengan hasil proyeksi yang dihasilkan oleh fungsi yang disediakan oleh DirectX Graphics yaitu D3DXMatrixPerspectiveFovLH() atau D3DXMatrixPerspectiveFovRH().

Tabel 6.1 Pengujian Subjektif Hasil Proyeksi

Hasil Proyeksi dari Perkalian Matriks	Hasil Proyeksi dari Fungsi DirectX Graphics	Keterangan
		Sama
		Berbeda

Dari pengujian terlihat bahwa hasil proyeksi sama apabila koordinat lokal kamera berada tidak dekat dengan objek 3D yang diproyeksikan. Apabila koordinat lokal kamera didekatkan pada objek yang diproyeksikan, hasil proyeksi yang didapat dari perkalian matriks terdapat efek regangan atau *stretch*.

6.2 Pengujian Objektif Hasil Proyeksi

Pengujian objektif hasil proyeksi dilakukan dengan mencari nilai ekstrim *(extreme value)* hasil proyeksi terhadap titik-titik yang akan diproyeksikan. Nilai ekstrim didapatkan dengan melakukan rentetan perkalian matriks pada verteks terhadap matriks pandang dan matriks proyeksi.

$$Matriks\ Pandang = \begin{bmatrix} KameraRight.x & KameraUp.x & KameraLook.x & 0 \\ KameraRight.y & KameraUp.y & KameraLook.y & 0 \\ KameraRight.z & KameraUp.z & KameraLook.z & 0 \\ -(\overrightarrow{PosisiKamera}.\overrightarrow{KameraRight}) & -(\overrightarrow{PosisiKamera}.\overrightarrow{KameraUp}) & -(\overrightarrow{PosisiKamera}.\overrightarrow{KameraLook}) & 1 \end{bmatrix}$$

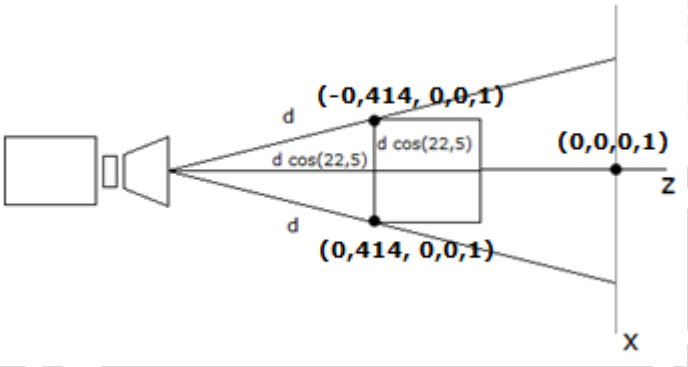
$$M_{proyeksi} = \begin{bmatrix} \frac{1}{Rasio\ Aspek} cot\frac{a}{2} & 0 & 0 & 0 \\ 0 & cot\frac{a}{2} & 0 & 0 \\ 0 & 0 & \frac{f}{f-n} & 1 \\ 0 & 0 & -\frac{fn}{f-n} & 0 \end{bmatrix}$$

Pada pengujian objektif didefinisikan koordinat lokal kamera sebesar $(0,0,-100)$ dengan sudut pandang *field of view* sebesar 45° dan rasio aspek 1:1. *Near plane* dan *far plane* diberikan nilai masing-masing sebesar 1 dan 100. Sehingga diperoleh nilai matriks pandang dan matriks proyeksi masing masing sebesar

Matriks Pandang = $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 100 & 1 \end{bmatrix}$

Matriks Proyeksi = $\begin{bmatrix} 2{,}414 & 0 & 0 & 0 \\ 0 & 2{,}414 & 0 & 0 \\ 0 & 0 & 1{,}01 & 1 \\ 0 & 0 & -1{,}01 & 0 \end{bmatrix}$

Nilai titik-titik yang akan diuji adalah (0,0,0,1), (0,414 ,0,0,1), dan (−0,414 ,0,0,1). Nilai titik (0,0,0,1) merupakan titik tengah objek yang akan diproyeksi. Sedangkan nilai 0,414 didapatkan dari persamaan pitagoras antara jarak *near plane* dengan posisi koordinat lokal kamera.



Gambar 6.1 Titik-titik yang diproyeksikan

Hasil proyeksi yang diharapkan adalah x,y bernilai (0,0) untuk titik (0,0,0,1), x, y bernilai (1,0) untuk titik (0,414 ,0,0,1), dan x,y bernilai (-1,0) untuk titik (−0,414 ,0,0,1). Nilai (0,0) karena objek diharapkan berada di tengah-tengah bidang proyeksi atau koordinat ternormalisasi sedangkan nilai (1,0) dan (-1,0) karena objek diharapkan berada pada tepi bidang proyeksi atau koordinat ternormalisasi.

Untuk titik (0,0,0,1) = (0,0,0,1) $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 100 & 1 \end{bmatrix} \begin{bmatrix} 2{,}414 & 0 & 0 & 0 \\ 0 & 2{,}414 & 0 & 0 \\ 0 & 0 & 1{,}01 & 1 \\ 0 & 0 & -1{,}01 & 0 \end{bmatrix}$

Didapatkan hasil = (0,0, 99,9 ,100)

Karena matriks proyeksi merupakan matriks homogen maka hasil nilai koordinat ternormalisasi dibagi dengan elemen z senilai 100 sehingga diperoleh nilai (0,0, 0,99 ,1)

Untuk titik (0,414 ,0,0,1) = (0,0,0,1) $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 100 & 1 \end{bmatrix} \begin{bmatrix} 2{,}414 & 0 & 0 & 0 \\ 0 & 2{,}414 & 0 & 0 \\ 0 & 0 & 1{,}01 & 1 \\ 0 & 0 & -1{,}01 & 0 \end{bmatrix}$

Didapatkan hasil = (0,99 ,0, 0,1)

Untuk titik $(-0{,}414\ ,0{,}0{,}1) = (0{,}0{,}0{,}1)$ $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 100 & 1 \end{bmatrix} \begin{bmatrix} 2{,}414 & 0 & 0 & 0 \\ 0 & 2{,}414 & 0 & 0 \\ 0 & 0 & 1{,}01 & 1 \\ 0 & 0 & -1{,}01 & 0 \end{bmatrix}$

Didapatkan hasil = $(-0{,}99\ ,0,\ 0{,}1)$

Dari pengujian secara objektif dapat ditarik kesimpulan bahwa hasil proyeksi yang dilakukan dengan menggunakan perkalian dengan matriks pandang dan matriks proyeksi sudah sesuai dengan hasil yang diharapkan.

6.3 Pengujian Subjektif Metode Pengalamatan

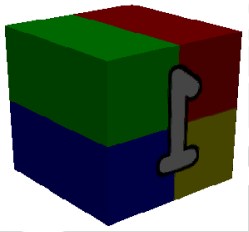
Pengujian subjektif metode pengalamatan dilakukan dengan melakukan pengamatan terhadap hasil tampilan gambar tekstur 2D pada objek 3D yang telah diberikan metode-metode pengalamatan planar, spheris, dan silindris. Gambar tekstur yang digunakan dalam pengujian subjektif merupakan gambar dengan tulisan angka satu beserta empat bagian warna yang berbeda untuk memudahkan dalam hal pengamatan.

Gambar 6.2 Gambar tekstur yang digunakan pada pengujian

6.3.1 Pengujian Subjektif Metode Pengalamatan Planar

Pengujian subjektif metode pengalamatan planar dilakukan dengan memberikan gambar tekstur kepada objek primitif yang mempunyai sifat planar atau bidang datar berbentuk kubus dengan metode pengalamatan planar, kemudian selanjutnya diterapkan kepada objek kompleks berbentuk pakaian.

Tabel 6.2 Pengujian Subjektif Metode Pengalamatan Planar

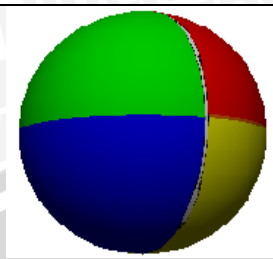
Tampak Depan	Tampak Belakang	Keterangan
		Berhasil diterapkan
		Berhasil diterapkan

Dari hasil pengujian subjektif metode pengalamatan planar, dapat diamati bahwa metode pengalamatan planar berhasil diterapkan pada objek bersifat planar yaitu kubus dan gambar tekstur masih tampak sesuai dengan gambar asli. Setelah metode pengalamatan planar diterapkan pada objek kompleks berbentuk pakaian, hasil yang didapatkan perihal gambar tekstur juga masih bersesuaian dengan gambar aslinya, hal ini dikarenakan objek pakaian juga memiliki elemen kumpulan verteks yang berbentuk mendekati bidang datar yaitu bagian depan dan belakang pakaian.

6.3.2 Pengujian Subjektif Metode Pengalamatan Spheris Berdasarkan Persamaan Koordinat Bola dan Sudut Dot Product

Pengujian subjektif metode pengalamatan spheris dilakukan dengan memberikan gambar tekstur kepada objek primitif yang mempunyai sifat spheris berbentuk bola dengan metode pengalamatan spheris yang berasal dari persamaan koordinat bola dan hasil sudut dot product. Setelah itu kemudian diterapkan kepada objek kompleks berbentuk pakaian.

Tabel 6.3 Pengujian Subjektif Metode Pengalamatan Spheris Berdasarkan Persamaan Koordinat Bola dan Sudut Dot Product

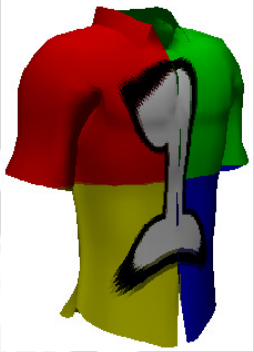
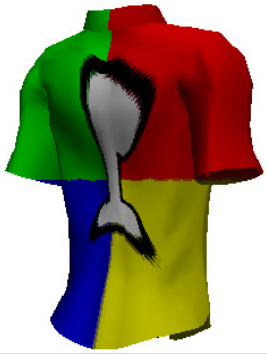
Tampak Depan	Tampak Belakang	Keterangan
		Berhasil diterapkan
		Berhasil diterapkan

Dari hasil pengujian subjektif metode pengalamatan spheris berdasarkan persamaan koordinat bola dan sudut dot product, dapat diamati bahwa metode pengalamatan spheris berhasil diterapkan pada objek bersifat spheris yaitu objek bola, dan gambar tekstur masih tampak sesuai dengan gambar asli. Setelah metode pengalamatan spheris diterapkan pada objek kompleks berbentuk pakaian, hasil yang didapatkan perihal gambar tekstur tampak mengalami sedikit distorsi pada bagian depan dan belakang objek pakaian. Hal ini disebabkan karena metode pengalamatan spheris berorientasi pada sudut verteks secara vertikal dan horisontal terhadap titik tengah objek; sehingga apabila sebuah objek pakaian mempunyai elemen bidang datar seperti pada sisi depan dan belakang pakaian, maka akan mengalami efek distorsi karena bidang datar tidak dipengaruhi oleh sudut verteks melainkan posisi verteks terhadap titik tengah.

6.3.3 Pengujian Subjektif Metode Pengalamatan Spheris Berdasarkan Normal Vektor

Pengujian subjektif metode pengalamatan spheris berdasarkan normal vektor dilakukan dengan memberikan gambar tekstur kepada objek primitif yang mempunyai sifat spheris berbentuk bola dengan metode pengalamatan spheris berdasarkan normal vektor pada masing-masing verteks milik objek primitif, kemudian selanjutnya diterapkan kepada objek kompleks berbentuk pakaian. Metode pengalamatan spheris berdasarkan normal vektor bersifat simetris parsial yang berarti pengalamatan gambar tekstur mirip dengan pengalamatan planar hanya saja pada pengalamatan spheris memperhitungkan faktor lengkungan.

Tabel 6.4 Pengujian Subjektif Metode Pengalamatan Spheris Berdasarkan Normal Vektor

Tampak Depan	Tampak Belakang	Keterangan
		Berhasil diterapkan
		Berhasil diterapkan

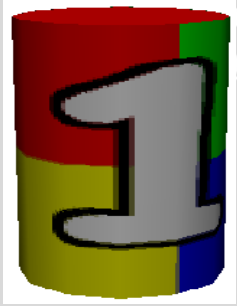
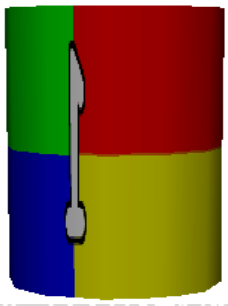
Dari hasil pengujian subjektif metode pengalamatan spheris berdasarkan normal vektor, dapat diamati bahwa metode pengalamatan spheris berhasil diterapkan pada objek bersifat spheris yaitu objek bola, dan gambar tekstur masih tampak sesuai dengan gambar asli. Setelah metode pengalamatan spheris diterapkan pada objek kompleks berbentuk pakaian, efek distorsi menjadi lebih

terlihat pada bagian depan dan belakang objek pakaian dibandingkan dengan pengalamatan spheris berdasarkan persamaan koordinat bola dan sudut dot product. Hal ini semakin menunjukkan bahwa metode pengalamatan secara spheris lebih sesuai dengan objek yang mempunyai elemen bersifat spheris atau mempunyai lengkungan dengan orientasi vertikal dan horisontal.

6.3.4 Pengujian Subjektif Metode Pengalamatan Silindris Berdasarkan Persamaan Koordinat Silinder

Pengujian subjektif metode pengalamatan silindris dilakukan dengan memberikan gambar tekstur kepada objek primitif yang mempunyai sifat silindris yaitu objek berbentuk silinder dengan metode pengalamatan silindris yang berasal dari persamaan koordinat silinder, setelah itu diterapkan kepada objek kompleks berbentuk pakaian.

Tabel 6.5 Pengujian Subjektif Metode Pengalamatan Silindris Berdasarkan Persamaan Koordinat Silinder

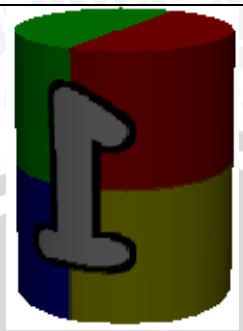
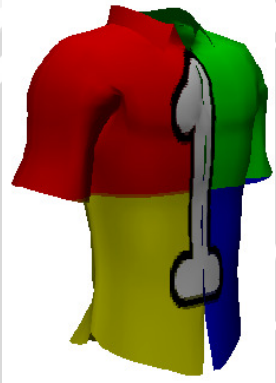
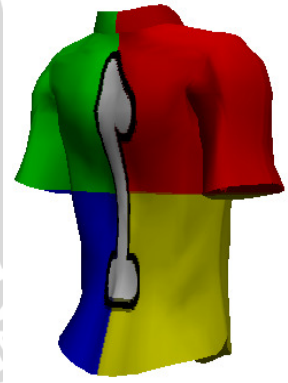
Tampak Depan	Tampak Belakang	Keterangan
		Berhasil diterapkan
		Berhasil diterapkan

Dari hasil pengujian subjektif metode pengalamatan silindris berdasarkan persamaan koordinat silinder, dapat diamati bahwa metode pengalamatan silindris berhasil diterapkan pada objek bersifat silindris yaitu objek silinder, dan gambar tekstur masih tampak sesuai dengan gambar asli. Setelah metode pengalamatan silindris diterapkan pada objek kompleks berbentuk pakaian, hasil yang didapatkan perihal gambar tekstur juga tampak sesuai dengan gambar asli dengan sedikit atau tidak ada distorsi sama sekali. Hal ini disebabkan karena metode pengalamatan silindris merupakan gabungan antara metode pengalamatan planar dan spheris. Sehingga apabila sebuah objek 3D mempunyai elemen bidang datar sekaligus memiliki elemen lengkungan maka metode pengalamatan yang lebih sesuai untuk dipilih adalah metode pengalamatan silindris.

6.3.5 Pengujian Subjektif Metode Pengalamatan Silindris Berdsasrkan Normal Vektor

Pengujian subjektif metode pengalamatan silindris berdasarkan normal vektor dilakukan dengan memberikan gambar tekstur kepada objek primitif yang mempunyai sifat silindris yaitu objek berbentuk silinder dengan metode pengalamatan silindris berdasarkan normal vektor, kemudian selanjutnya diterapkan kepada objek kompleks berbentuk pakaian.

Tabel 6.6 Pengujian Subjektif Metode Pengalamatan Silindris Berdasarkan Normal Vektor

Tampak Depan	Tampak Belakang	Keterangan
		Berhasil diterapkan
		Berhasil diterapkan

Dari hasil pengujian subjektif metode pengalamatan silindris berdasarkan persamaan normal vektor, dapat diamati bahwa metode pengalamatan silindris berhasil diterapkan pada objek bersifat silindris yaitu objek silinder, dan gambar tekstur masih tampak sesuai dengan gambar asli. Setelah metode pengalamatan silindris berdasarkan normal vektor diterapkan pada objek kompleks berbentuk pakaian, kesesuaian gambar tekstur lebih terlihat dibandingkan dengan pengalamatan silindris dari penurunan persamaan koordinat silinder, yaitu gambar tekstur berhasil dipetakan tanpa menghilangkan sifat asli pada objek yang memiliki elemen bidang datar sekaligus elemen bidang lengkung.

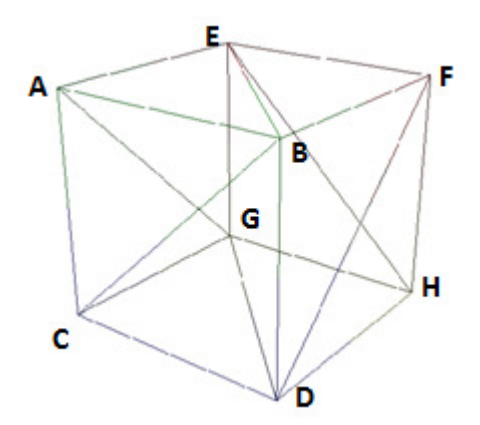
6.4 Pengujian Objektif Metode Pengalamatan

Pengujian objektif metode pengalamatan dilakukan dengan mendapatkan nilai-nilai koordinat tekstur u dan v terhadap objek-objek primitif berupa objek kubus, bola, dan silinder. Pencarian nilai-nilai koordinat tekstur u dan v

didapatkan dari sudut-sudut istimewa kelipatan 90 dan 0 pada bidang spheris dan silindris.

6.4.1 Pengujian Objektif Metode Pengalamatan Planar

Pengujian objektif metode pengalamatan planar dilakukan dengan mendapatkan nilai-nilai koordinat tekstur u dan v pada objek primitif berupa objek kubus. Nilai koordinat tekstur u dan v didapatkan dari verteks-verteks penyusun objek kubus.



Gambar 6.3 Titik-titik pada Kubus yang Digunakan sebagai Pengujian

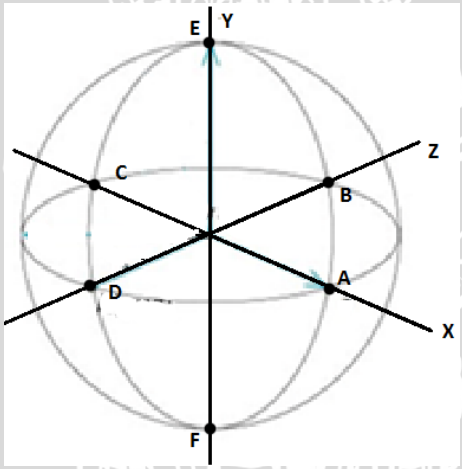
Tabel 6.7 Pengujian Objektif Metode Pengalamatan Planar

Titik	Posisi	$u = \frac{x}{x_{max} - x_{min}} + 0{,}5$	$v = -\frac{y}{y_{max} - y_{min}} + 0{,}5$
A	-10, 10, -10	0	0
B	10,10,-10	1	0
C	-10,-10,-10	0	1
D	10,-10,-10	1	1
E	-10, 10, 10	0	0
F	10, 10, 10	1	0
G	-10, -10, 10	0	1
H	10, -10, 10	1	1

Dari tabel pengujian dapat ditarik kesimpulan bahwa hasil yang didapatkan dari pengujian objektif sudah sesuai dengan nilai yang diharapkan untuk metode pengalamatan secara planar yaitu nilai koordinat tekstur sesuai atau berbanding lurus dengan bidang datar atau sumbu x dan y.

6.4.2 Pengujian Objektif Metode Pengalamatan Spheris Berdasarkan Persamaan Koordinat Bola dan Sudut Dot Product

Pengujian objektif metode pengalamatan spheris berdasarkan persamaan koordinat bola dilakukan dengan mendapatkan nilai-nilai koordinat tekstur u dan v terhadap objek primitif berupa objek bola yang dialamati dengan metode pemetaan spheris berdasarkan persamaan koordinat bola serta sudut hasil dot product. Pencarian nilai-nilai koordinat tekstur u dan v didapatkan dari sudut-sudut istimewa kelipatan 90 dan 0.



Gambar 6.4 Titik-titik pada Bola yang Digunakan sebagai Pengujian

Jika $N_Z \geq 0$

$$koordinat\ tekstur\ u = \frac{\arccos\left(A \cdot \vec{i}\right)}{2\pi}$$

Jika $N_Z < 0$

$$koordinat\ tekstur\ u = 1 - \frac{\arccos\left(A \cdot \vec{i}\right)}{2\pi}$$

$$koordinat\ tekstur\ v = \frac{arcos\left({}^{y}/_{r}\right)}{\pi}$$

Tabel 6.8 Pengujian Objektif Metode Pengalamatan Spheris Berdasarkan Persamaan Koordinat Bola dan Sudut Dot Product

Titik	Posisi	N_z	u	v
A	10, 0, 0	0	0	0,5
B	0, 0, 10	1	0,25	0,5
C	-10, 0, 0	0	0,5	0,5
D	0, 0, -10	-1	0,75	0,5
E	0, 10, 0	0	0,25	0
F	0, -10, 0	0	0,25	1

Dari tabel pengujian dapat ditarik kesimpulan bahwa hasil yang didapatkan dari pengujian objektif sudah sesuai dengan nilai yang diinginkan untuk metode pengalamatan secara spheris yaitu nilai koordinat tekstur u dan v berbanding lurus dengan besar sudut φ dan θ sesuai bidang lengkung horisontal dan vertikal.

6.4.3 Pengujian Objektif Metode Pengalamatan Spheris Berdasarkan Normal Vektor

Pengujian objektif metode pengalamatan spheris berdasarkan normal vektor dilakukan dengan mendapatkan nilai-nilai koordinat tekstur u dan v terhadap objek primitif berupa objek bola yang dialamati dengan metode pemetaan spheris berdasarkan normal vektor. Pencarian nilai-nilai koordinat tekstur u dan v didapatkan dari sudut-sudut istimewa kelipatan 90 dan 0.

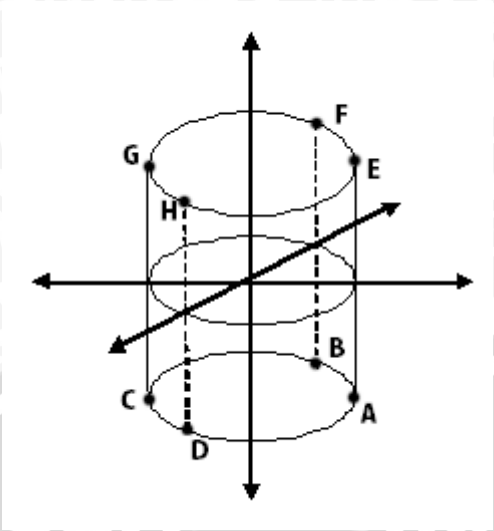
Tabel 6.9 Pengujian Objektif Metode Pengalamatan Spheris Berdasarkan Normal Vektor

Titik	Posisi	$u = \frac{\text{asin}(N_x)}{\pi} + 0{,}5$	$v = -\frac{\text{asin}(N_y)}{\pi} + 0{,}5$
A	10, 0, 0	1	0,5
B	0, 0, 10	0,5	0,5
C	-10, 0, 0	0	0,5
D	0, 0, -10	0,5	0,5
E	0, 10, 0	0,5	0
F	0, -10, 0	0,5	1

Dari tabel pengujian dapat ditarik kesimpulan bahwa hasil yang didapatkan dari pengujian objektif sudah sesuai dengan nilai yang diinginkan untuk metode pengalamatan secara spheris yaitu nilai koordinat tekstur u dan v berbanding lurus dengan besar sudut φ dan θ sesuai bidang lengkung horisontal dan vertikal.

6.4.4 Pengujian Objektif Metode Pengalamatan Silindris Berdasarkan Persamaan Koordinat Silinder

Pengujian objektif metode pengalamatan silindris berdasarkan persamaan koordinat silinder dilakukan dengan mendapatkan nilai-nilai koordinat tekstur u dan v terhadap objek primitif berupa objek silinder yang dialamati dengan metode pemetaan silindris yang berasal dari persamaan koordinat silinder. Pencarian nilai-nilai koordinat tekstur u didapatkan dari sudut-sudut istimewa kelipatan 90 dan 0.



Gambar 6.5 Titik-titik pada Silinder yang Digunakan sebagai Pengujian

Tabel 6.10 Pengujian Objektif Metode Pengalamatan Silindris Berdasarkan Persamaan Koordinat Silinder

Titik	Posisi	N_Z	$u = \frac{\arccos(x/r)}{2\pi}$	$v = -\frac{y}{y_{max} - y_{min}} + 0{,}5$
A	10, -20,0	0	0	1
B	0, -20,10	1	0,25	1
C	-10, -20, 0	0	0,5	1
D	0, -20,-10	-1	0,75	1
E	10, 20,0	0	0	0
F	0, 20,10	1	0,25	0
G	-10, 20, 0	0	0,5	0
H	0, 20,-10	-1	0,75	0

Dari tabel pengujian dapat ditarik kesimpulan bahwa hasil yang didapatkan dari pengujian objektif sudah sesuai dengan nilai yang diinginkan untuk metode pengalamatan secara silindris yaitu nilai koordinat tekstur u berbanding lurus dengan besar sudut φ sesuai dengan bidang lengkung horisontal

dan koordinat tekstur v berbanding lurus dengan sumbu y sesuai dengan tinggi silinder.

6.4.5 Pengujian Objektif Metode Pengalamatan Silindris Berdasarkan Normal Vektor

Pengujian objektif metode pengalamatan silinder berdasarkan normal vektor dilakukan dengan mendapatkan nilai-nilai koordinat tekstur u dan v terhadap objek primitif berupa objek silinder yang dialamati dengan metode Pengalamatan silindris berdasarkan normal vektor. Pencarian nilai-nilai koordinat tekstur u dan v didapatkan dari sudut-sudut istimewa kelipatan 90 dan 0.

Tabel 6.11 Pengujian Objektif Metode Pengalamatan Silindris Berdasarkan Normal Vektor

Titik	Posisi	$u = \frac{asin(N_x)}{\pi} + 0{,}5$	$v = -\frac{y}{y\,max\ -\ y\,min} + 0{,}5$
A	10, -20,0	1	1
B	0, -20,10	0,5	1
C	-10, -20, 0	0	1
D	0, -20,-10	0,5	1
E	10, 20,0	1	0
F	0, 20,10	0,5	0
G	-10, 20, 0	0	0
H	0, 20,-10	0,5	0

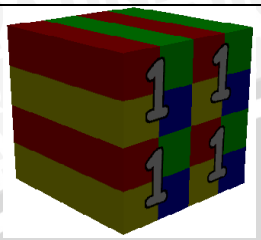
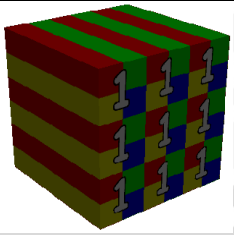
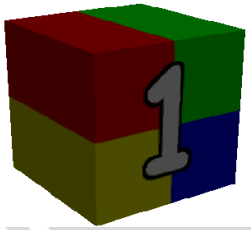
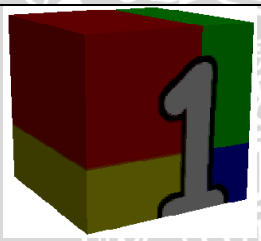
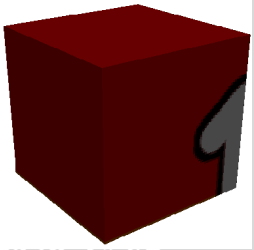
Dari tabel pengujian dapat ditarik kesimpulan bahwa hasil yang didapatkan dari pengujian objektif sudah sesuai dengan nilai yang diinginkan untuk metode pengalamatan secara silindris yaitu nilai koordinat tekstur u berbanding lurus dengan besar sudut φ sesuai dengan bidang lengkung horisontal

dan koordinat tekstur v berbanding lurus dengan sumbu y sesuai dengan tinggi silinder.

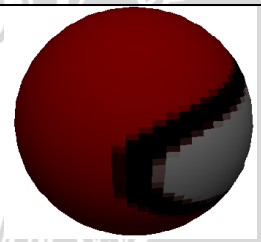
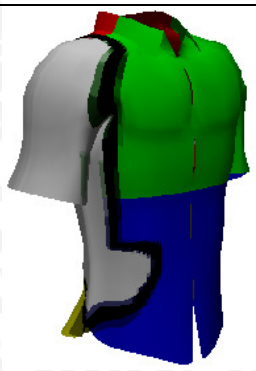
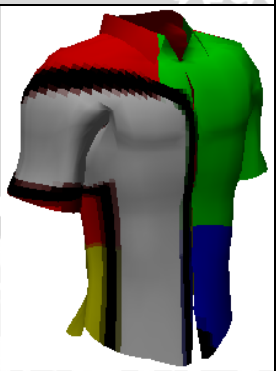
6.5 Pengujian Hasil Manipulasi Tekstur

Pengujian Hasil Manipulasi Tekstur dilakukan dengan mendapatkan tampilan proyeksi berupa gambar tekstur terpetakan dengan koordinat tekstur yang telah diubah nilainya. Pengujian manipulasi tektur dilakukan pada tiap-tiap metode pengalamatan dengan perubahan nilai koordinat tekstur u dan v dengan menggunakan faktor skalasi minifikasi dan faktor skalasi magnifikasi. Pengujian hanya dilakukan pada pengaruh skalasi karena faktor translasi dan faktor kemiringan *(shearing)* mempunyai sistematika yang hampir sama atau merupakan alterasi atau modifikasi dari konsep skalasi.

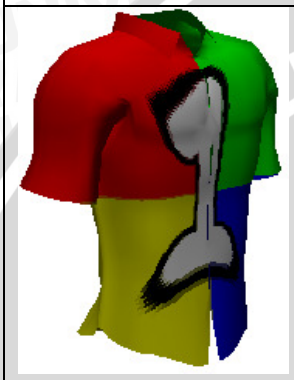
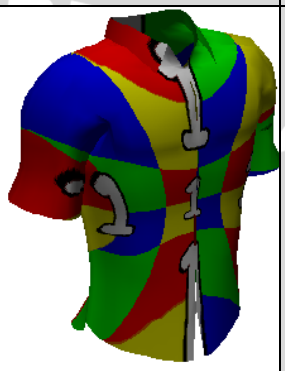
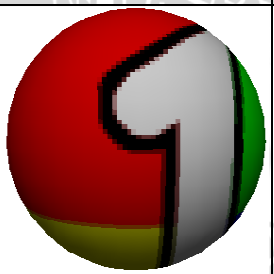
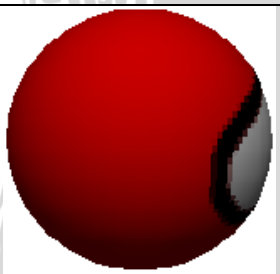
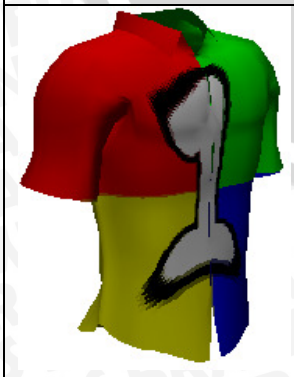
Tabel 6.12 Pengujian Manipulasi Tekstur Metode Pengalamatan Planar

$u = 1$ $v = 1$	$u = 2$ $v = 2$	$u = 3$ $v = 3$	Keterangan
			Berhasil diterapkan
			Berhasil diterapkan
$u = 1$ $v = 1$	$u = 0{,}75$ $v = 0{,}75$	$u = 0.5$ $v = 0.5$	Keterangan
			Berhasil diterapkan
			Berhasil diterapkan

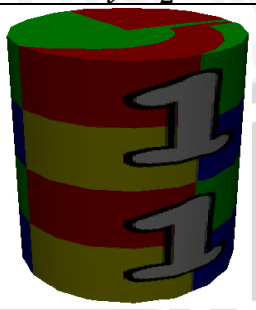
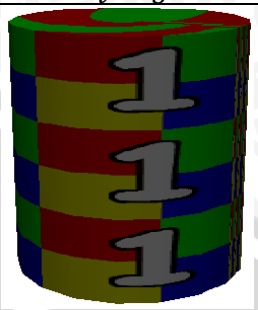
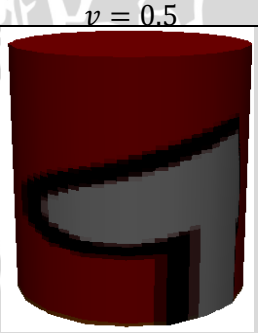
Tabel 6.13 Pengujian Manipulasi Tekstur Metode Pengalamatan Spheris Berdasarkan Persamaan Koordinat Bola dan Sudut Dot Product

$u = 1$ $v = 1$	$u = 2$ $v = 2$	$u = 3$ $v = 3$	Keterangan
			Berhasil diterapkan
			Berhasil diterapkan
$u = 1$ $v = 1$	$u = 0{,}75$ $v = 0{,}75$	$u = 0.5$ $v = 0.5$	Keterangan
			Berhasil diterapkan
			Berhasil diterapkan

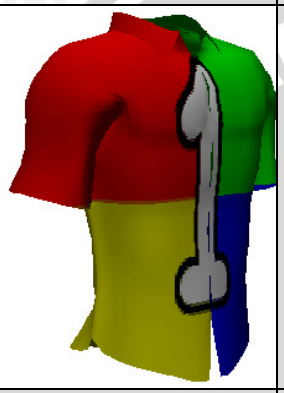
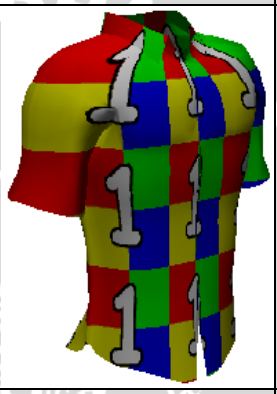
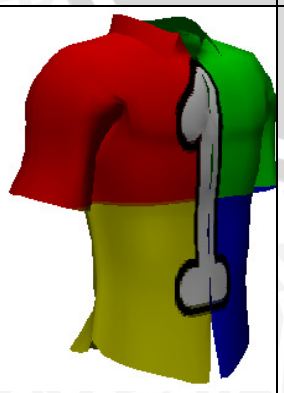
Tabel 6.14 Pengujian Manipulasi Tekstur Metode Pengalamatan Spheris Berdasarkan Normal Vektor

$u = 1$ $v = 1$	$u = 2$ $v = 2$	$u = 3$ $v = 3$	Keterangan
			Berhasil diterapkan
			Berhasil diterapkan
$u = 1$ $v = 1$	$u = 0{,}75$ $v = 0{,}75$	$u = 0.5$ $v = 0.5$	Keterangan
			Berhasil diterapkan
			Berhasil diterapkan

Tabel 6.15 Pengujian Manipulasi Tekstur Metode Pengalamatan Silindris Berdasarkan Persamaan Koordinat Silinder

$u = 1$ $v = 1$	$u = 2$ $v = 2$	$u = 3$ $v = 3$	Keterangan
			Berhasil diterapkan
			Berhasil diterapkan
$u = 1$ $v = 1$	$u = 0{,}75$ $v = 0.75$	$u = 0.5$ $v = 0.5$	Keterangan
			Berhasil diterapkan
			Berhasil diterapkan

Tabel 6.16 Pengujian Manipulasi Tekstur Metode Pengalamatan Silindris Berdasarkan Normal Vektor

$u = 1$ $v = 1$	$u = 2$ $v = 2$	$u = 3$ $v = 3$	Keterangan
			Berhasil diterapkan
			Berhasil diterapkan
$u = 1$ $v = 1$	$u = 0{,}75$ $v = 0{,}75$	$u = 3$ $v = 3$	Keterangan
			Berhasil diterapkan
			Berhasil diterapkan

6.6 Analisis Hasil Pengujian

Dengan hasil yang diperoleh dari pengujian maka dapat dilakukan analisis sebagai berikut

a. Hasil proyeksi yang didapat dari hasil perkalian matriks terjadi efek regangan *(stretch)* karena matris proyeksi yang digunakan tidak memperhitungkan nilai z milik verteks. Sedangkan DirectX Graphics masih menggunakan nilai z milik verteks sebagai faktor pengali pada fungsi proyeksi.
b. Pada metode pengalamatan secara spheris dan silindris terdapat kelemahan yaitu adanya dua titik berdekatan dengan nilai koordinat tekstur yang kontras.
c. Pada bidang datar, metode pengalamatan yang lebih tepat digunakan adalah metode pengalamatan planar karena interpolasi gambar diambil secara linier menurut sumbu x dan y tanpa memperhitungkan faktor lengkungan sehingga gambar tekstur hasil pemetaan tampak bersesuaian dengan gambar asli.
d. Pada bidang lengkung, metode pengalamatan yang lebih tepat digunakan adalah metode pengalamatan spheris karena interpolasi gambar diambil secara linier sebanding dengan sudut φ atau θ maupun normal vektor milik kumpulan verteks yang membentuk lengkungan sehingga gambar tekstur hasil pemetaan tampak bersesuaian dengan gambar asli.
e. Pada objek yang memiliki bidang datar dan bidang lengkung, metode pengalamatan yang lebih tepat digunakan adalah metode pengalamatan silindris karena interpolasi gambar diambil secara linier sebanding dengan sudut φ dan secara planar sebanding dengan tinggi silinder sehingga gambar tekstur hasil pemetaan tampak bersesuaian dengan gambar asli.
f. Manipulasi porsi dan posisi gambar berhasil dilakukan dengan melakukan kali matriks pada nilai koordinat tekstur dengan menggunakan elemen matriks m_{11} dan m_{22} sebagai faktor skalasi, elemen matriks m_{31} dan m_{32} sebagai faktor translasi, dan elemen matriks m_{12} dan m_{21} sebagai faktor kemiringan *(shearing)*.

BAB VII
KESIMPULAN DAN SARAN

Berdasarkan hasil-hasil yang dicapai selama perancangan, pembuatan dan pengujian dari penelitian ini, maka dapat diambil beberapa kesimpulan dan saran.

7.1 Kesimpulan

Dari hasil perancangan dan pengujian yang telah dilakukan dapat disimpulkan beberapa hal berikut:

1. Tahapan-tahapan transformasi untuk memproyeksikan objek pakaian 3D berhasil dilakukan dengan melakukan konkatenasi atau rentetan perkalian matriks *(matrix concatenation)* menggunakan matriks ruang, matriks pandang, dan matriks proyeksi yang diperoleh dari penurunan ataupun modifikasi matriks-matriks transformasi yang didapat dari materi referensi.
2. Keberhasilan proyeksi perspektif ditentukan dengan membandingkan hasil proyeksi yang didapat dengan menggunakan matriks pandang dan matriks proyeksi pada tahap perancangan dengan fungsi yang disediakan oleh DirectX Graphics. Hasil proyeksi yang diperoleh dengan menggunakan matriks proyeksi terdapat anomali berupa regangan *(stretch)* karena tidak menyertakan faktor pengali z dalam matriks proyeksi. Keberhasilan proyeksi perspektif juga ditentukan dengan melakukan perhitungan nilai-nilai ekstrim titik-titik dalam koordinat 3D pada bidang proyeksi ternormalisasi.
3. Pemberian alamat gambar pada objek pakaian 3D dapat dilakukan dengan mendistribusikan nilai-nilai koordinat tekstur dengan metode pengalamatan yang dilakukan secara planar yaitu sebanding dengan bidang datar sesuai sumbu x dan y, secara spheris yaitu sebanding dengan sudut φ dan θ, dan secara silindris yaitu sebanding dengan sudut φ dan sumbu y.
4. Agar tampilan gambar sesuai dengan gambar asli pada verteks-verteks penyusun objek pakaian 3D yang membentuk bidang datar, metode

pengalamatan yang lebih tepat digunakan adalah metode pengalamatan planar karena interpolasi gambar diambil secara linier menurut sumbu x dan y tanpa memperhitungkan faktor lengkungan. Sedangkan pada verteks-verteks penyusun objek 3D yang membentuk bidang lengkung, metode pengalamatan yang lebih tepat digunakan adalah metode pengalamatan spheris dan silindris karena interpolasi gambar diambil secara linier sebanding dengan sudut φ atau θ maupun normal vektor milik kumpulan verteks yang membentuk lengkungan.

5. Manipulasi porsi dan posisi gambar pada objek pakaian 3D berhasil dilakukan dengan melakukan kali matriks pada nilai koordinat tekstur dengan menggunakan elemen matriks m_{11} dan m_{22} sebagai faktor skalasi, elemen matriks m_{31} dan m_{32} sebagai faktor translasi, dan elemen matriks m_{12} dan m_{21} sebagai faktor kemiringan *(shearing)*.

7.2 Saran

Hasil penelitian ini masih terdapat kekurangan yang masih bisa diperbaiki pada penelitian selanjutnya di tahun yang akan datang. Adapun beberapa kekurangan yang perlu diperbaiki pada penelitian yang akan datang adalah:

1. Keterbatasan dari DirectX adalah minimnya kode sumber *(source code)*. Seorang programmer tentunya ingin melakukan kontrol hingga ke akar sehingga konsep COM*(Component Object Model)* masih dirasa kurang transparan.
2. Manipulasi tekstur mungkin dirasa kurang cukup dari sudut pandang seorang desainer. Para desainer tentu juga ingin melakukan manipulasi bentuk pada objek 3D sehingga tidak terkesan monoton.

DAFTAR PUSTAKA

Hanson, Chris. 2009. ***The Ultimate DirectX Tutorial***. http://www.directxtutorial.com. Diakses tanggal 15 Desember 2009

Farrell, Joe, 2005, ***Deriving Projection Matrices,*** http://www.codeguru.com/cpp/misc/misc/graphics/article.php/c10123/Deriving-Projection-Matrices.htm. Diakses tanggal 16 Januari 2010

Samyn, Koen, 2009, ***View transformation Local Coordinate System.*** http://www.knol.com/k/matrices-for-3d-applications-view-transformation. Diakses tanggal 26 Maret 2010

Anonymous, 2009, ***Graphics Programming with DirectX 9 Module I***, Team LRN

Anonymous, 2009, ***Graphics Programming with DirectX 9 Module II***, Team LRN

Horton's, Ivor. 2006. ***Beginning Visual C++ 2005.*** Indianapolis:Wiley Publishing, Inc.

Bjornander, Stefan. 2008. ***Microsoft Visual C++ Windows Application by Example***. Birmingham: Packt Publishing, Ltd.

Engel, Wolfgang F. 2001. ***Beginning Direct3D game programming***. California:Prima Publishing.

Jones, Wendi. 2004. ***BeginningDirectX9***. Boston:Premiere Press.

Gray, Kris. 2003. ***DirectX 9 Programmable Graphics Pipeline.*** Microsoft Press: Washington.

Thorn, Alan.2005. ***The Definitive Guide to Direct3D*** Texas:Word warez publishing, Inc.

Thorn, Alan.2004. ***DirectX9 User Intrfaces*** Texas:Word ware publishing, Inc.

Luna, Frank,D.2003. ***Introduction to 3D game programming with DirectX 9.0***Texas:Word warez publishing, Inc.

Zake, Dianne.2011. ***An Introduction to Programming with C++***. Boston:Course Technology.

Mayo, Joe.2010. ***Visual Studio 2010.***McGraw-Hill Companies.

Schildt, Herbert.2010. ***C++:The Complete Refference.***McGraw-Hill Companies.