2. TEORI PENUNJANG

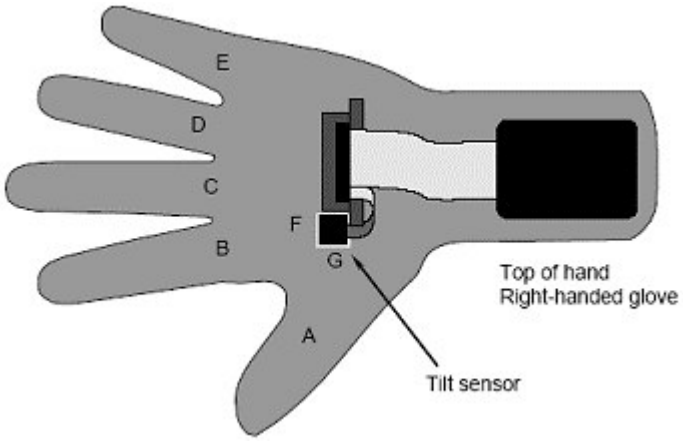
2.1. *Data Glove* (*Wired Glove*)

2.1.1. Tipe *Data Glove*

Data glove (untuk definisinya lihat bab pertama, subbab pertama, paragraf kedua dan baris ketujuh belas) yang digunakan pada aplikasi ini diproduksi oleh perusahaan Fifth Dimension Technologies (5DT). Tipe-tipe *data glove* yang tersedia sampai sekarang ini adalah :

a. 5DT Data Glove 5 (kanan dan kiri)
b. 5DT Data Glove 5-W (*wireless* (tanpa kabel), kanan dan kiri)
c. 5DT Data Glove 5 Ultra
d. 5DT Data Glove 16 (kanan dan kiri)
e. 5DT Data Glove 16-W (*wireless* (tanpa kabel), kanan dan kiri)
f. 5DT Data Glove 14 Ultra
g. 5DT Data Glove 14 Ultra Wireless Kit (*wireless* (tanpa kabel))

Dari tipe-tipe yang telah disebutkan di atas terdapat angka yang menunjukkan jumlah banyak sensor yang terdapat dalam satu *data glove* tersebut. Jumlah sensor terkecil dalam sebuah *data glove* adalah 5 (lima), 14 (empat belas) dan 16 (enam belas). Letak sensor pada *data glove* yang memiliki lima sensor ada pada tiap jari yang terdapat satu sensor. Letak sensor untuk *data glove* yang memiliki empat belas sensor adalah setiap jari memiliki dua sensor dan setiap ruas antar jari memiliki satu sensor. Letak sensor untuk *data glove* yang memiliki enam belas sensor hampir sama dengan *data glove* dengan 14 sensor tetapi dengan tambahan dua sensor yang letaknya di dekat pergelangan tangan (untuk mendeteksi gerakan pergelangan tangan dan pangkal ibu jari (jari jempol)). Kemudian dari sebagian tipe ada yang mempunyai nama belakang “Ultra” yang berarti kabel penyambungnya merupakan USB (*Universal Serial Bus*) yang lebih sering dipakai untuk berbagai alat dan untuk tipe-tipe selain “Ultra” hanya memiliki kabel penyambung berupa kabel *Communications Port* (COM) yang sekarang sudah jarang dipakai.

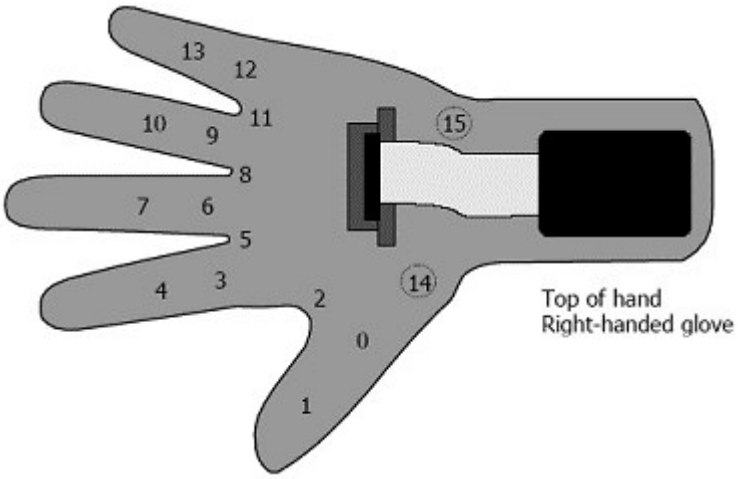


Gambar 2.1 Letak sensor untuk 5DT Data Glove 5
(Sumber : "*5DT Data Glove 16 Manual*" – 5DT Company)

Tabel 2.1 Pemetaan sensor untuk 5DT Data Glove 5

Sensor	Indeks sensor *driver*	Keterangan
A	0 atau 1	Sudut ibu jari
B	3 atau 4	Sudut jari telunjuk
C	6 atau 7	Sudut jari tengah
D	9 atau 10	Sudut jari manis
E	12 atau 13	Sudut jari kelingking
F	16 (tidak ada di tipe "Ultra")	Sudut kemiringan (atas-bawah) tangan
G	17 (tidak ada di tipe "Ultra")	Sudut perputaran (kiri-kanan) tangan

(Sumber : "*5DT Data Glove 16 Manual*" – 5DT Company)



Gambar 2.2 Letak sensor 5DT Data Glove 16
(Sumber : "*5DT Data Glove 16 Manual*" – 5DT Company)

Tabel 2.2 Pemetaan sensor untuk 5DT Data Glove 16 atau 14

Sensor	Indeks sensor *driver*	Keterangan
0	0	Sudut ibu jari 1
1	1	Sudut ibu jari 2
2	2	Sudut ruas ibu jari – jari telnujuk
3	3	Sudut jari telunjuk 1
4	4	Sudut jari telunjuk 2
5	5	Sudut ruas jari telunjuk – jari tengah
6	6	Sudut jari tengah 1
7	7	Sudut jari tengah 2
8	8	Sudut ruas jari tengah – jari manis
9	9	Sudut jari manis 1
10	10	Sudut jari manis 2
11	11	Sudut ruas jari manis – jari kelingking
12	12	Sudut jari kelingking 1
13	13	Sudut jari kelingking 2
14	14 (Belum terimplementasi)	Sudut pangkal ibu jari
15	15 (Belum terimplementasi)	Sudut pergelangan tangan

(Sumber : "*5DT Data Glove 16 Manual*" – 5DT Company)

2.1.2. Fitur *Data Glove*

Data glove yang dibuat oleh 5DT ini mempunyai fitur-fitur yang membuat *data glove* ini bisa dipakai oleh *programmer* pemula maupun yang sudah ahli. Fitur-fitur tersebut adalah :

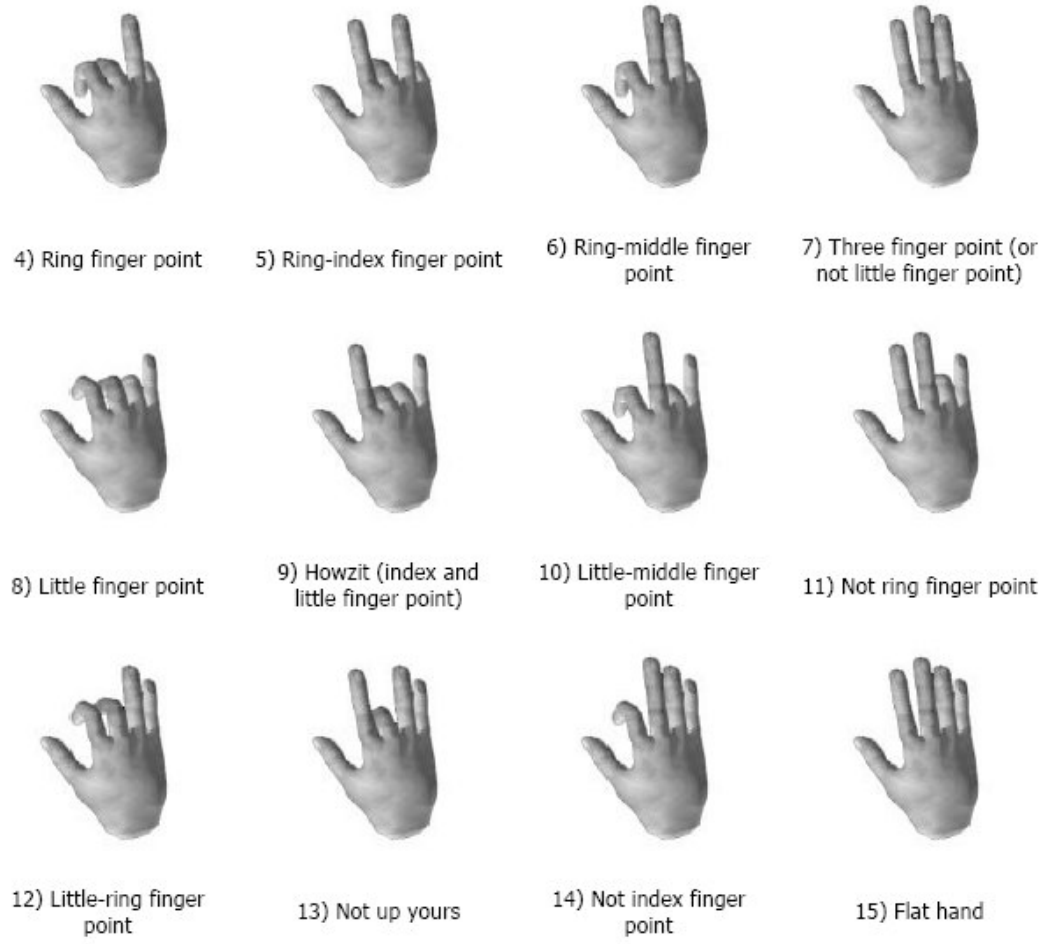
a. Desain yang sangat nyaman dan cocok dipakai untuk berbagai ukuran tangan baik wanita maupun pria.
b. Sensor yang akurat dan sensitif yang bisa meminimalkan penggunaan penyaringan sinyal.
c. Mempunyai fasilitas pengenalan bahasa isyarat yang bisa dikenali langsung oleh *data glove* tersebut (bisa dilihat pada halaman berikutnya).
d. Aplikasi diagnosa yang berfungi untuk melihat kondisi *data glove* yang ada sudah tersedia dalam paket tersebut.
e. Dalam paket tersebut terdapat plug-in untuk digunakan pada aplikasi Kaydara MOCAP™.
f. Fungsi dan data dari *data glove* dapat diakses melalui 5DT Data Glove SDK (*Software Development Kit*).

Cara pemakaian *data glove* relatif tidak sulit dan terutama bagi pengguna aplikasi Microsoft Visual C++ 6.0. Hal ini dikarenakan SDK yang sudah ada memakai bahasa pemrograman C++. Untuk pengambilan data dari *data glove* bisa dengan dua metode, yaitu :

a. Langsung (*Raw*), artinya data dari sensor tidak disaring atau diubah sedikitpun. Metode ini tidak memakai kalibrasi otomatis yang ada di metode lainnya. Untuk menggunakan metode ini, yang harus dilakukan adalah transformasi data dari sensor menjadi sudut gerakan jari.
b. Tersaring (*Scaled*), artinya data dari sensor telah tersaring dan mempunyai nilai data *float* antara 0 sampai 1. Metode ini menggunakan kalibrasi otomatis yang digunakan untuk mempermudah pencarian kalibrasi optimal bagi semua jenis dan ukuran tangan manusia. Untuk menggunakan metode ini, yang harus dilakukan cuma memakai nilai *float* dari sensor untuk dirubah menjadi sudut gerakan jari.

Perusahaan 5DT memberikan sebuah fasilitas tambahan selain sensor gerakan tangan ke dalam *data glove*, yaitu fasilitas pengenalan bahasa isyarat yang bisa dikenali langsung dari alat tersebut tanpa membutuhkan aplikasi khusus. Pengenalan bahasa isyarat ini hanya terbatas pada enam belas macam bentuk isyarat saja sesuai jumlah biner dari 2^4. Jadi bahasa isyarat yang bisa dikenali langsung oleh alatnya tidak bisa digunakan untuk bahasa isyarat untuk orang cacat seperti bisu dan tuli. Mengenai enam belas macam bentuk bahasa isyarat tersebut bisa dilihat pada gambar 2.3.

0) Fist　1) Index finger point　2) Up Yours (Middle finger point)　3) Two finger point



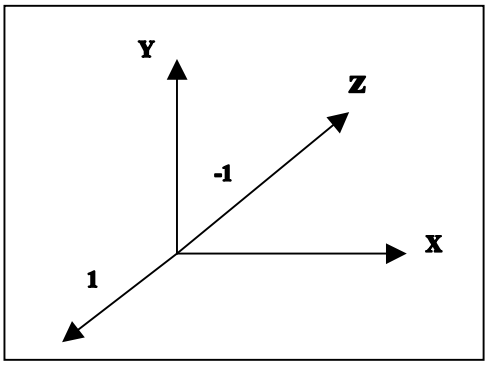
Gambar 2.3 Bentuk 16 isyarat sederhana
(Sumber : "5DT Data Glove Manual" – 5DT Company)

Penjelasan lebih lanjut mengenai penggunaan data glove akan dijelaskan pada bab selanjutnya.

2.2. OpenGL dan Komputer Grafis 3D

Komputer grafis 3D merupakan pengembangan dari komputer grafis 2D. Perbedaan antara penggambaran obyek dua dimensi dan obyek tiga dimensi yaitu ada pada sistem koordinat yang digunakan. Jika dalam penggambaran obyek secara dua dimensi menggunakan sistem koordinat x dan y yang mewakili lebar dan tinggi maka penggambaran obyek secara tiga dimensi menggunakan sistem koordinat x, y dan z dimana z mewakili kedalaman (atau ketebalan) dari sebuah obyek. Dengan adanya penggambaran kedalaman dari suatu obyek maka obyek yang digambar akan terlihat mendekati aslinya atau semakin mendekati dunia

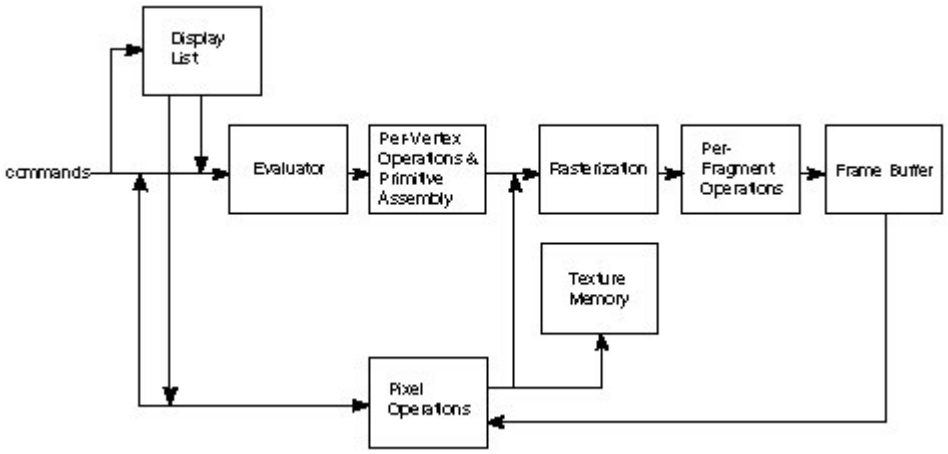
nyata yang mempunyai panjang, lebar dan kedalaman sehingga komputer grafis 3D ini bisa dipakai dalam VR untuk simulasi dan permainan. Gambar sistem koordinat 3D bisa dilihat pada gambar 2.4.



Gambar 2.4 Sistem koordinat 3D

Dalam aplikasi Tugas Akhir ini untuk penampilan animasi tangan dipakai bahasa grafik OpenGL (Open Graphic Library). OpenGL adalah sebuah perangkat lunak antar muka (*software interface*) untuk perangkat keras grafik (*graphics hardware*). Antar muka ini terdiri dari sekitar 150 perintah yang digunakan untuk menggambarkan sebuah obyek dan terdiri dari operasi yang diperlukan untuk menghasilkan aplikasi tiga dimensi yang interaktif.

OpenGL dibuat sebagai sebuah perangkat lunak antar muka yang bisa diimplementasikan pada banyak perangkat keras yang berlainan. OpenGL tidak menyediakan perintah tingkat tinggi untuk menggambar model tiga dimensi seperti sebuah pesawat, mobil dan manusia. Untuk menggambar model tiga dimensi di atas dengan OpenGL dapat dibuat dari sebuah *geometric primitives* (kelompok obyek dasar) seperti titik, garis dan poligon. OpenGL mempunyai sebuah *library* (GLU) yang mendukung untuk pembuatan model tiga dimensi.



Gambar 2.5 Proses penggambaran OpenGL
(Sumber : "Blue Book OpenGL" – Addison - Wesley Publishing Company)

Dari gambar 2.5, semua perintah yang dijalankan tidak selalu langsung masuk ke dalam proses penggambaran tetapi bisa disimpan dulu ke dalam *display list* (daftar tampilan) yang digunakan untuk proses penggambaran yang lebih cepat dan konstan nantinya. Maksud dari lebih cepat adalah untuk menggambar obyek tidak harus melewati proses lengkap penggambaran.

Evaluator adalah proses yang memberikan efisiensi untuk menghitung dan memperkirakan geometri kurva dan permukaan dengan mengevaluasi perintah-perintah *polynomial* dari nilai input yang masuk. Pada langkah berikutnya, OpenGL memproses berbagai *geometric primitives* yang telah terdeskripsikan oleh *vertices* (deskripsi obyek dari *evaluator*). Setelah proses *Rasterization* maka didapat tiap *frame fragments* (bagian-bagian frame atau tampilan) yang nantinya ditampung di *frame buffer* (tempat tampung tampilan).

Data input tidak hanya berupa *vertices* tetapi bisa berupa *pixels*. Untuk data input *pixels*, proses yang dilakukan sedikit berbeda, yaitu proses pertama tidak masuk ke *evaluator*, operasi *vertex* dan *primitive assembly* tetapi masuk ke proses operasi *pixels* yang nantinya digunakan untuk proses *rasterization* atau dimasukkan ke *texture memory* untuk *texture mapping*.

Hal-hal yang berkaitan dengan OpenGL pada *windows* adalah :

a. GDI *Device Context*

Untuk menggambar pada *windows* tanpa menggunakan O*penGL*, digunakan fungsi *Graphics Device Interface* (GDI). Setiap *window* mempunyai

sebuah *device context* yang menerima *output* grafis dan setiap fungsi GDI mengambil sebuah *device context* sebagi sebuah argumen untuk mengindikasikan pada *window* mana fungsi tersebut bekerja[1].

OpenGL tidak mempunyai *context identifier*, tetapi disebut *rendering context*. *Rendering context* mirip dengan GDI *device context*, karena fungsi *rendering context* adalah untuk mengingat warna sekarang, *state setting* dan lainnya.

b. *Pixel Formats*

Konsep *device context* pada *window* terbatas jika digunakan untuk grafis 3D, karena sesungguhnya *device context* didesain untuk digunakan pada aplikasi 2D. Format *device context* tergantung pada format *device*. Jadi jika *desktop* menggunakan warna 16-bit, maka *device context* juga menggunakan warna 16-bit.

Sementara pada OpenGL, dapat ditentukan format window yang akan digunakan untuk menggambar. Konfigurasi window yang dapat ditentukan di antaranya: *software rendering*, *hardware rendering*, *single buffer*, *double buffer*, *depth buffer* dan lainnya.

Karakteristik 3D window diset satu kali, biasanya pada awal pembuatan window. Nama kolektif untuk setting ini adalah pixel format[2]. Window mempunyai struktur yang disebut PIXELFORMATDESCRIPTOR yang digunakan untuk mendeskripsikan pixel format. Struktur ini didefinisikan sebagai berikut:

```
typedef struct tag PIXELFORMATDESCRIPTOR {
WORD  nSize;              // Ukuran struktur ini
WORD  nVersion;     // Versi struktur ini (seharusnya 1)
DWORD dwFlags;      // Properti pixel buffer
BYTE  iPixelType;   // Tipe dari dat pixel (RGBA atau Color Index)
BYTE  cColorBits;   // Jumlah warna bit plane pada color buffer
BYTE  cRedBits;     // Berapa banyak bit untuk merah
BYTE  cRedShift;    // Shift count untuk bit merah
BYTE  cGreenBits;   // Berapa banyak bit untuk hijau
```

[1] Wright, Richard S., & Lipchak, Benjamin. (2005). *OpenGL Superbible* (3rd ed.), hal 648
[2] Wright, Richard S., & Lipchak, Benjamin. (2005). *OpenGL Superbible* (3rd ed.), hal 650

```
BYTE    cGreenShift; // Shift count untuk bit hijau
BYTE    cBlueBits;   // Berapa banyak bit untuk biru
BYTE    cBlueShift;  // Shift count untuk bit biru
BYTE    cAlphaBits;  // Berapa banyak bit untuk alpha tujuan
BYTE    cAlphaShift; // Shift count untuk bit alpha tujuan
BYTE    cAccumBits; // Bbanyak bit untuk acummulation buffer
BYTE    cAccumRedBits;      // Banyak bit merah untuk acummulation buffer
BYTE    cAccumGreenBits;  // Banyak bit hijau untuk acummulation buffer
BYTE    cAccumBlueBits;   // Banyak bit biru untuk acummulation buffer
BYTE    cAccumAlphaBits; // Banyak bit alpha untuk acummulation buffer
BYTE    cDepthBits;  // Banyak bit untuk depth buffer
BYTE    cStencilBits; // Banyak bit untuk stencil buffer
BYTE    cAuxBuffer; // Banyak auxiliary buffer
BYTE    iLayerType;  // Tidak terpakai - diabaikan
BYTE    bReserved;   // Jumlah overlay dan underlay plane
DWORD  dwLayerMask;      // Tidak terpakai - diabaikan
DWORD  dwVisibleMask;   // Warna transparan dari underlay plane
DWORD  dwDamageMask; // Tidak terpakai - diabaikan
} PIXELFORMATDESCRIPTOR;
```

Keterangan :

- Struktur di atas ditulis dalam bahasa pemrograman C++.
- BYTE, WORD dan DWORD adalah jenis variabel yang ada di C++.

c. OpenGL *Rendering Context*

Aplikasi *windows* pada umumnya dapat terdiri dari banyak *windows*. Setiap *window* dapat mempunyai *pixel format* tersendiri dan penentuan *pixel format* hanya dapat dilakukan satu kali untuk tiap *window*.

Untuk menggunakan fungsi-fungsi dalam *OpenGL*, setiap *environment* harus menspesifikasikan *rendering window* yang sekarang digunakan sebelum mengeksekusi perintah-perintah OpenGL. Seperti fungsi *Windows* GDI yang menggunakan *windows device context*, OpenGL mengunakan apa yang disebut

rendering context[3]. Seperti *device context* yang mengingat *setting* tentang mode penggambaran dan perintah-perintah untuk GDI, *rendering context* juga mengingat *setting* dan perintah-perintah.

Untuk membuat OpenGL *rendering context* digunakan perintah fungsi WglCreateContext. Fungsi ini mempunyai satu parameter yaitu *device context* dari sebuah *window* dengan sebuah *pixel format* yang valid. Tipe data *OpenGL rendering context* adalah HGLRC.

2.2.1. Proyeksi pada Obyek 3D

Proyeksi pada obyek tiga dimensi digunakan untuk menampilkan obyek tiga dimensi pada bidang dua dimensi karena layar pada komputer menggunakan sistem dua dimensi. Ada dua macam proyeksi yaitu *parallel projection* dan *perspective projection*[4]. *Parallel projection* dilakukan dengan cara menarik garis lurus pada setiap koordinat obyek sampai memotong bidang xy pada layar sesuai dengan sudut pandang pengamat. *Perspective projection* menggunakan satu titik pusat sebagai sudut pandang, semua koordinat obyek yang tampak ditarik garis lurus menuju titik pusat sehingga akan menghasilkan gambar pada bidang xy. Dalam *OpenGL* ada beberapa *function* yang digunakan untuk melakukan proyeksi tiga dimensi salah satunya adalah menggunakan fungsi gluPerspective:

void gluPerspective(GLdouble fovy, GLdouble aspect, GLdouble zNear, GLdouble zFar)
fovy = sudut pandang pengamat.
aspect = perbandingan antara tinggi dan lebar dari layar (lebar / tinggi)
zNear = jarak terdekat ke layar
zFar = jarak terjauh ke layar

2.2.2. Transformasi Tiga Dimensi

Transformasi tiga dimensi merupakan perubahan bentuk maupun perubahan posisi dari obyek yang disebabkan oleh proses translasi, skala dan rotasi. Masing-masing akan dijelaskan sebagai berikut:

[3] Wright, Richard S., & Lipchak, Benjamin. (2005). *OpenGL Superbible* (3rd ed.), hal 657
[4] Harrington, Steven. "*Computer Graphics A Programming Approach"*.Mc-Graw-Hill,Inc

a. Translasi, yaitu memindahkan posisi obyek dari posisi awal menuju posisi tertentu sejauh x terhadap sumbu x, sejauh y terhadap sumbu y dan sejauh z terhadap sumbu z.

void glTranslatef(GLfloat x, GLfloat y, GLfloat z)

b. Skala, yaitu melakukan skala pada obyek menjadi lebih besar atau lebih kecil. *Scaling* dilakukan berdasarkan perkalian sebesar x terhadap ukuran x semula, perkalian sebesar y terhadap ukuran y semula dan perkalian sebesar z terhadap ukuran z semula.

void glScalef(GLfloat x,GLfloat y, GLfloat z)

c. Rotasi, yaitu memutar obyek searah maupun berlawanan arah jarum jam. Rotasi dapat dilakukan dengan memutar pada sumbu x, y atau z. Rotasi dilakukan berdasarkan besar sudut rotasi yang sudah ditentukan terhadap sumbu x, y atau z.

	glRotatef(GLfloat angle, GLfloat x, GLfloat y, GLfloat z)
angle	= sudut rotasi
x	= skala 0 – 1 untuk perubahan yg ingin terjadi sumbu x
y	= skala 0 – 1 untuk perubahan yg ingin terjadi sumbu y
z	= skala 0 – 1 untuk perubahan yg ingin terjadi sumbu z

2.2.3. Obyek *Mesh*

Obyek *mesh* merupakan suatu obyek yang terdiri dari kumpulan banyak poligon yang membentuk suatu bentuk geometri. Biasanya poligon yang digunakan untuk membentuk suatu *mesh* obyek berbentuk segitiga. Poligon merupakan suatu kurva tertutup yang terdiri dari tiga atau lebih garis yang membentuk suatu bidang. Poligon dibentuk oleh titik-titik koordinat yang disebut *vertex* yang dihubungkan dengan dua *vertex* lain sehingga akan membentuk segitiga. Poligon yang terbentuk dinamakan *face* dari obyek tersebut. Jika poligon-poligon yang terhubung membentuk suatu obyek dan poligon tersebut tidak diberi warna hanya terdiri dari garis-garis yang terhubung maka poligon-poligon tersebut membentuk *wireframe* dari obyek. Jadi *wireframe* merupakan gambaran obyek tiga dimensi yang memperlihatkan garis-garis atau *edges* yang membentuk obyek tersebut.

Fungsi yang digunakan untuk membuat obyek *mesh* :

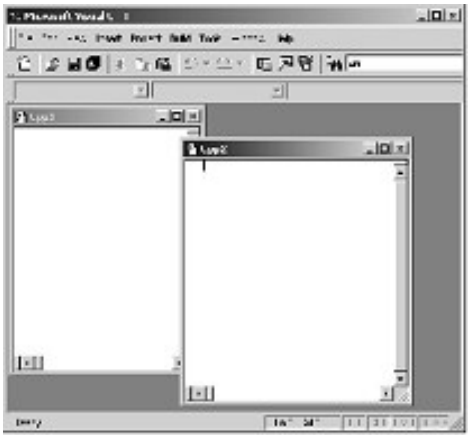
a. void glVertex3f(GLfloat x, GLfloat y, GLfloat z) : koordinat 3D dari *vertex.*
b. void glNormal3f(GLfloat nx, GLfloat ny, GLfloat nz) : normal dari *vertex.*
c. glBegin(GL_POLYGON) : menentukan jenis poligon.
d. void gl*Polygon*Mode(GLenum face, GLenum mode) : menentukan jenis *polygon* yang dibuat yaitu *lines* atau *fill.*

2.3. Bahasa Pemrograman yang Dipakai

Dalam pembuatan aplikasi untuk karya Tugas Akhir ini akan dibuat dalam bahasa pemrograman Microsoft Visual C++ 6.0. Mengenai cara dan tipe aplikasi atau proyek yang dibuat nantinya akan memakai jenis proyek MFC (*Microsoft Foundation Class*). Dalam MFC ini terdapat tiga buah tipe aplikasi, yaitu MDI (*Multiple Documents Interface*), SDI (*Single Document Interface*), dan *Dialog-Based Document*.

2.3.1. MDI

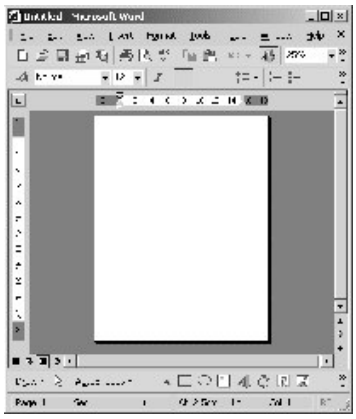
MDI adalah tipe aplikasi (*window*) yang bisa mempunyai *child form* (form turunan) yang artinya dalam sebuah *window* utama bisa membuka sebanyak mungkin *window* turunan yang hanya akan tampak di dalam batas antarmuka (*interface*) *window* utama. Jadi MDI ini cocok untuk tipe aplikasi yang membutuhkan pembukaan banyak dokumen langsung dan sekaligus. Contoh aplikasi yang memakai MDI ini adalah Microsoft Visual C++ 6.0.



Gambar 2.6 Bentuk aplikasi MDI

2.3.2. SDI

SDI adalah tipe aplikasi (*window*) yang tidak bisa mempunyai banyak *child form* (form turunan) yang artinya dalam sebuah *window* utama hanya bisa membuka sebuah dokumen yang hanya akan tampak di dalam batas antarmuka (*interface*) *window* utama. Jadi SDI ini cocok untuk tipe aplikasi yang hanya membutuhkan pembukaan sebuah dokumen saja dan biasanya tipe aplikasi ini digunakan untuk *text editor* (perubah teks) yang sederhana. Contoh aplikasi yang memakai SDI ini adalah Microsoft Word. Contoh bentuk dari SDI dapat dilihat pada gambar di bawah ini.



Gambar 2.7 Bentuk aplikasi SDI

2.3.3. *Dialog-Based Documents*

Dialog-Based Documents adalah bagian utama dari semua jenis aplikasi baik SDI maupun MDI yang tidak memiliki fasilitas menu dan operasi dokumen seperti SDI dan MDI. Denga kata lain, tipe aplikasi ini digunakan jika ingin membuat jenis aplikasi yang unik dan seefisien mungkin dimana *programmer* bisa mengatur semuanya dari awal dengan sendirinya. Contoh kasus adalah jika *programmer* ingin membuat aplikasi tanpa operasi dokumen dan menu, maka sebaiknya memilih *dialog-based* saja dan memilih sendiri fasilitas dan komponen yang diperlukan. Dengan begitu aplikasi yang dibuat tidak memboroskan *memory* dan tidak memerlukan ukuran tempat penyimpanan yang besar untuk aplikasi yang disimpan. Contoh aplikasi yang memakai dialog-based adalah aplikasi *game* (permainan) seperti 3D Pinball yang tidak memakai fasilitas utama operasi

dokumen seperti *load*, *save* dan *print*. Contoh bentuk aplikasinya dapat dilihat pada gambar 2.8.

Gambar 2.8 Bentuk aplikasi *Dialog-Based Documents*

2.4. OpenAL

OpenAL (Open Audio Library) adalah sebuah perangkat lunak antar muka (*software interface*) untuk perangkat keras *audio* (*audio hardware*) yang bisa dipakai di berbagai bahasa pemrograman. OpenAL ini didesain untuk efisiensi *rendering* untuk *multichannel three dimensional positional audio*. Cara pemakaian dan pemrogramannya mirip dengan OpenGL.

Kegunaan umum dari OpenAL adalah dikodekan dalam bentuk obyek sumber suara, *audio buffers* dan sebuah pendengar. Sebuah obyek sumber suara terdiri dari sebuah pointer terhadap sebuah *buffer*, kecepatan, posisi, arah dan intesitas dari suara. Sebuah obyek pendengar terdiri dari kecepatan, posisi dan arah dari pendengar dan volume untuk semua sumber suara. *Audio buffer* terdiri dari data *audio* dalam *format* PCM (Pulse Code Modulation) yang bisa 8-*bit*, 16-*bit*, *monaural* atau *stereo*.

Jika penggunaan OpenAL yang benar diimplementasikan pada aplikasi yang dibuat oleh end-user, maka suara yang dihasilkan bisa menyerupai suara yang dihasilkan seperti di dunia nyata ketika pendengar bergerak dalam ruang *virtual* tiga dimensi. Keindahan OpenAL dari perpektif *programmer* adalah sedikitnya kebutuhan untuk membuat hal di atas terjadi dalam aplikasi OpenGL tiga dimensi.