Pembuatan Simulasi Pergerakan Objek 3D (Tiga Dimensi) Menggunakan *OpenGL*

Deddy Suhardiman, S.T.G. Kaunang, Rizal Sengkey, Arthur M. Rumagit
Jurusan Teknik Elektro-FT, UNSRAT, Manado-95115, Email: deddy050213024@gmail.com

Abstrak **- Grafika Komputer adalah bagian dari ilmu komputer yang berkaitan dengan pembuatan dan manipulasi gambar (visual) secara digital. Bentuk sederhana dari grafika komputer adalah grafika komputer 2D yang kemudian berkembang menjadi grafika komputer 3D, pemrosesan citra *(image processing),* dan pengenalan pola *(pattern recognition).* Grafika komputer sering dikenal juga dengan istilah visualisasi data. Grafik meliputi gambar dan pencitraan lain yang dihasilkan oleh komputer berbentuk garis, lengkungan, kurva dan sebagainya. Komputer dapat menghasilkan pencitraan dalam sejumlah *pixel*, dan *printer dot matrix* akan mencetak citra/gambar tersebut dalam sejumlah titik.**

***OpenGL* (*Open Graphics Library*) adalah standar API yang dapat digunakan untuk membuat aplikasi berbasis grafik, baik dua dimensi (2D) maupun tiga dimensi (3D). *OpenGL* ini bersifat *cross-platform*, artinya dapat dijalankan pada berbagai *platform* sistem operasi yang ada saat ini.**

Untuk membuat aplikasi menggunakan *OpenGL*, terlebih dahulu kita membutuhkan suatu konsepsi *interfacing* dalam implementasinya pada proteksi objek. Salah satu cara yang sudah umum digunakan adalah dengan membuat *window-based OpenGL*. Untuk dapat membuat konsep *windowing* pada *OpenGL*, kita memerlukan *tool* tertentu. Yang kita gunakan kali ini adalah GLUT (*OpenGL Utility Toolkit*). GLUT dipilih karena di dalamnya telah terdapat banyak fungsi yang dapat dipakai untuk pembuatan *application window*. Disamping itu, *windowing* pada GLUT juga bersifat independen terhadap sistem operasi, sehingga kita tidak perlu repot-repot untuk mengubah kode program jika diterapkan pada sistem operasi yang berbeda.

kata kunci : Grafika Komputer, *OpenGL*, Proyeksi Objek, GLUT

I. PENDAHULUAN

A. Latar Belakang

Digital Image Processing, Komputer Grafik, Analisis dan *Computer* telah dikembangkan dan diaplikasikan dengan mengesankan selama beberapa dekade. Dimana perkembangan aplikasi-aplikasi yang menggunakan disiplin ilmu ini telah memimpin teknologi di beberapa bidang seperti komunikasi digital dan internet, penyiaran (*broadcasting*), alat kedokteran, sistem multimedia, biologi, ilmu pengetahuan material, robot, dan manufaktur, sistem *intelligent sensing*, *remote sensing*, seni grafik dan proses *print*. Pertumbuhan yang pesat ini direfleksikan dengan diterbitkannya *paper* di jurnal ilmiah internasional setiap tahunnya dan diluncurkannya buku-buku tentang Pemrosesan *Image Digital* dan Komputer Grafik.

Riset-riset mengenai cara mempermudah memvisualisasikan ide atau data secara lebih cepat dan akurat telah banyak dilakukan, khususnya teknologi visualisasi tiga dimensi (3D) telah mengalami perkembangan yang sangat pesat, misalnya dengan diciptakan teknologi *motion capture*, menggerakan obyek 3D menjadi lebih mudah, *facial capture* membuat animasi ekspresi wajah menjadi lebih mudah dan akurat, *scanner* 3D membuat proses pemodelan objek tiga dimensi (3D) menjadi lebih cepat dan akurat, *software-software authoring* tiga dimensi yang lengkap dengan objek-objek 3D siap pakai (*pre-made*) serta sistem *template* 3D mempercepat desain suatu model objek tiga dimensi.

Namun pada umumnya *software-software authoring* tiga dimensi tersebut tidak mendukung representasi visual tiga dimensi (3D) dari data eksternal secara *real time*, melainkan harus melalui proses *rendering* untuk menjadi sebuah file video yang baru bisa diputar ketika proses *rendering* selesai. Ketika perubahan parameter objek dilakukan terhadap sebuah visualisasi 3D, maka perlu proses *render* yang cukup memakan waktu sebelum hasil akhirnya bisa dilihat.

B. Identifikasi Masalah

1. Bagaimana membuat grafik tiga dimensi dari suatu objek yang dapat dilihat dari berbagai sudut pandang.
2. Bagaimana cara implementasi *OpenGL* dengan aplikasi bahasa pemrograman C++.

C. Pembatasan Masalah

1. Membuat objek 3D (tiga dimensi) dengan cara memvisualisasikan grafik / gambar ke dalam bentuk tiga dimensi.
2. Membuat simulasi gerakan objek 3D (tiga dimensi).
3. Menggunakan *OpenGL* sebagai *library* berbasis grafis tiga dimensi yang di *compile* menggunakan C++.

D. Tujuan Penulisan

1. Merancang pemodelan objek 3D (tiga dimensi) menggunakan *OpenGL*.
2. Membuat simulasi objek 3D (tiga dimensi) menggunakan *OpenGL*.

II. LANDASAN TEORI

A. Computer Graphic / Grafika Komputer

Grafika komputer (Inggris: *Computer graphics*) adalah bagian dari ilmu komputer yang berkaitan dengan pembuatan dan manipulasi gambar (visual) secara digital. Bentuk sederhana dari grafika komputer adalah grafika komputer 2D yang kemudian berkembang menjadi grafika komputer 3D, pemrosesan citra *(image processing),* dan pengenalan pola *(pattern recognition).* Grafika komputer sering dikenal juga dengan istilah visualisasi data. Grafik meliputi gambar dan pencitraan lain yang dihasilkan oleh komputer berbentuk garis, lengkungan, kurva dan sebagainya. Komputer dapat menghasilkan pencitraan dalam sejumlah piksel, dan *printer dot matrix* akan mencetak citra/gambar tersebut dalam sejumlah titik.

Cabang ilmu Grafik yaitu adalah ;

1. Pemrosesan citra (*image processing*) adalah proses ini mempunyai ciri data masukan dan informasi keluaran yang berbentuk citra. Istilah pengolahan citra digital secara umum didefinisikan sebagai pemrosesan citra dua dimensi dengan komputer. Umumnya citra digital berbentuk persegi panjang atau bujur sangkar (pada beberapa sistem pencitraan ada pula yang berbentuk segienam) yang memiliki lebar dan tinggi tertentu. Ukuran ini biasanya dinyatakan dalam banyaknya titik atau piksel sehingga ukuran citra selalu bernilai bulat. Setiap titik memiliki koordinat sesuai posisinya dalam citra. Koordinat ini biasanya dinyatakan dalam bilangan bulat positif, yang dapat dimulai dari 0 atau 1 tergantung pada sistem yang digunakan. Setiap titik juga memiliki nilai berupa angka digital yang merepresentasikan informasi yang diwakili oleh titik tersebut. Format data citra digital berhubungan erat dengan warna.
2. Pengenalan pola (*pattern recognition*) dapat dikatakan sebagai kemampuan manusia mengenali objek-objek berdasarkan ciri-ciri dan pengetahuan yang pernah diamatinya dari objek-objek tersebut. Tujuan dari pengenalan pola ini adalah mengklasifikasi dan mendeskripsikan pola atau objek kompleks melalui pengetahuan sifat-sifat atau ciri-ciri objek tersebut. Apakah pola itu, pola dapat dikatakan sebagai identitas yang terdefinis dan dapat diberi suatu identifikasi atau nama. Pendekatan pengenalan pola ada 3 yaitu secara sintaks, statistik serta melalui jaringan saraf tiruan. Pendekatan secara sintaks adalah pendekatan dengan menggunakan aturan-aturan tertentu. Pendekatan metoda statistik adalah pendektan dengan menggunakan data-data yang berasal dari statistik seperti pasar saham. Pendekatan dengan pola jaringan saraf tiruan adalah pendekatan melalui pola-pola ini meniru cara kerja otak manusia, Pada pola ini sistem membuat *rule-rule* tertentu disertai dengan menggunakan data statistik sebagai dasar untuk pengambilan keputusan.

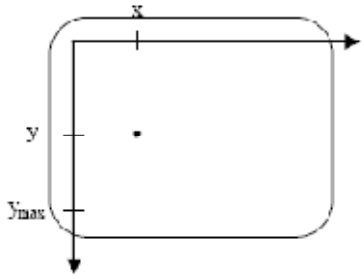
Bagian dari grafika komputer meliputi:

- Geometri: mempelajari cara menggambarkan permukaan bidang
- Animasi: mempelajari cara menggambarkan dan memanipulasi gerakan
- *Rendering*: mempelajari algoritma untuk menampilkan efek cahaya
- Citra (*Imaging*): mempelajari cara pengambilan dan penyuntingan gambar

B. Sistem Koordinat

Untuk menghasilkan suatu gambar pada paket pemrograman, yang pertama dibutuhkan untuk dilakukan adalah mendeskripsikan geometrik dari objek yang akan ditampilkan. Deskripsi ini menentukan lokasi dan bentuk dari objek. Untuk mendeskripsikannya dibutuhkan suatu standard yang dipakai kemudian ditetapkan standard yang dipakai menggunakan koordinat *Cartesian.*

Koordinat *Cartesian* mereferensikan posisi koordinat pada kuadrant pertama dari dimensi dua. Yang menjadi titik acuan dari koordinat ini adalah pada sudut kiri bawah seperti pada gambar 1. Sedangkan koordinat yang dipakai pada layar adalah juga koordinat *Cartesian* hanya saja titik acuan dari koordinat pada layar berada pada sudut kiri atas.



Gambar 1. Koordinat *Cartesian* yang dimulai dari sudut kiri bawah

Pada gambar 2, menunjukkan *Cartesian* yang dimulai dari sudut kanan atas. Jadi posisi pada layar merupakan kebalikan koordinat *Cartesian* yang biasanya digunakan.

Nilai untuk koordinat horizontal dari kedua sistem ini adalah sama dan untuk membalik nilai y dikonversikan ke nilai y yang sebenarnya yang dimulai dari bawah dengan menggunakan rumus sebagai berikut :

$$y = ymax - yinvert$$

Jika koordinat yang digunakan pada gambar menggunakan suatu koordinat yang lain maka harus dikonversikan kedalam koordinat *Cartesian* sebelum dimasukkan kedalam suatu paket grafik.

Suatu paket mungkin didesain untuk aplikasi-aplikasi khusus yang memberikan penggunaan sistem koordinat lain yang sesuai dengan aplikasi-aplikasi ini.

C. Transformasi Proyeksi

Pada tahap selanjutnya pada penglihatan tiga dimensi setelah ditransformasikan kedalam sistem koordinat, objek yang dideskripsikan diproyeksikan kedalam penglihatan pada bidang datar.

Pada umumnya grafik memiliki dua macam proyeksi yakni:

- Proyeksi Parallel
- Proyeksi Perspektif

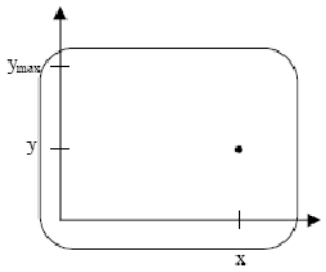
1. Proyeksi Parallel

Pada proyeksi paralel, posisi koordinat dirubah kedalam penglihatan pada bidang datar dengan menggunakan bantuan garis-garis paralel. Gambar 3 mengilustrasikan proyeksi paralel untuk suatu garis yang didefinisikan dengan dua titik koordinat yakni P1 and P2. Semua garis paralel pada layar akan ditampilkan secara paralel ketika sudut pandang yang dipakai adalah proyeksi paralel.

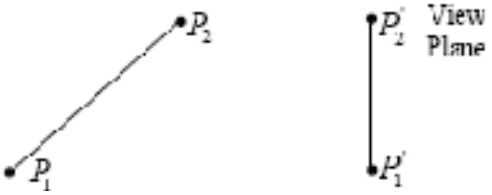
Ada dua metode yang umumnya digunakan untuk memperoleh proyeksi paralel pada penglihatan suatu objek. Dapat menarik suatu garis-garis yang tegak lurus pada suatu bidang datar atau dengan menggunakan suatu sudut miring (*oblique angle*) terhadap suatu bidang datar.

Jika transformasi yang dilakukan terhadap suatu objek terhadap bidang datar berupa garis-garis lurus yang semuanya parallel terhadap vektor normal pada bidang datar N maka disebut dengan proyeksi ortogonal (atau proyeksi orthografik).

Proyeksi ini menghasilkan proyeksi paralel dimana garis garis-garis proyeksi berupa garis yang tegak lurus terhadap bidang datar. Proyeksi orthogonal kebanyakan dipakai untuk menghasilkan sudut pandang penglihatan objek dari sisi depan, samping dan atas.



Gambar 2. Koordinat *Cartesian* yang dimulai dari sudut kanan atas



Gambar 3. Proyeksi parallel berupa garis terhadap bidang datar

2. Proyeksi Perspektif

Walaupun proyeksi paralel tampak lebih mudah membuatnya dan menjaga agar proporsi dari objek tetap, akan tetapi hal tersebut tidak merepresentasikan pada keadaan yang sebenarnya. Untuk mensimulasikan proyeksi ini, dibutuhkan pertimbangan arah sinar cahaya yang dicerminkan terhadap objek pada layar. Dimana dilakukan pendekatan pada efek optik geometrik dengan memproyeksikan objek terhadap bidang datar sepanjang satu jalur yang memusat pada suatu posisi yang disebut dengan titik acuan proyeksi (atau pusat proyeksi). Gambar 4 menunjukkan objek yang ditampilkan dengan efek-efek perspektif dan objek yang jauh akan tampak lebih kecil dibandingkan dengan proyeksi objek dengan ukuran yang sama yang terletak lebih dekat dengan bidang datar.

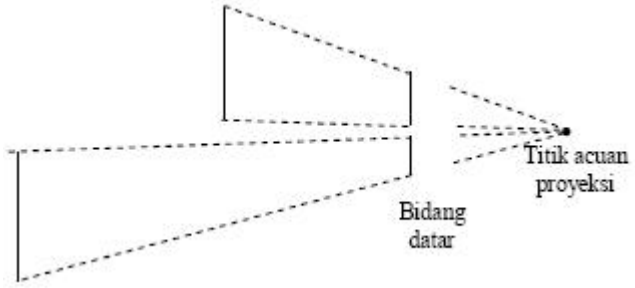
D. Konsep Tekstur

1. Pemetaan Tekstur

Tekstur adalah suatu metode umum yang digunakan untuk menambahkan suatu detail pada objek dengan memetakan suatu pola kedalam geometris dari objek yang didefinisikan. Pola tekstur didefinisikan dalam suatu array yang berisi nilai-nilai warna atau suatu prosedur yang mengubah warna objek. Metode ini untuk menambah detail objek pada layar yang biasa dikenal dengan *texture mapping* atau *pattern mapping*. Spesifikasi suatu tekstur yang menunjukkan pada suatu permukaan tekstur menggunakan koordinat tekstur yang mempunyai range nilai dari 0 sampai 1.0.

Tekstur pada suatu permukaan benda/objek umumnya didefinisikan dengan pola warna berbentuk segiempat dan posisi tekstur ini pada suatu permukaan berbentuk koordinat dua dimensi (s,t). Spesifikasi dari masing – masing warna pada pola tekstur disimpan pada suatu *array* yang terdiri dari 3 komponen *array*.

Misal jika pola tekstur didefinisikan dengan pola 16 x 16 warna RGB, jadi *array* untuk pola ini berisi 16 x 16 x 3 = 768 elemen.



Gambar 4. Proyeksi Perspektif Dari Dua Garis yang Memiliki Panjang yang Sama Dengan Jarak yang Berbeda Dari Bidang Datar

2. Redukasi Pola Tekstur

Pada animasi dan aplikasi-aplikasi yang lain, ukuran dari objek selalu berubah. Untuk objek yang ditampilkan dengan menggunakan tekstur, dibutuhkan prosedur pemetaan tekstur untuk mengubah dimensi dari objek. Ketika ukuran objek tekstur dikurangi, pola tekstur diaplikasikan pada wilayah yang kecil dan dapat menjadi petunjuk pada penyimpangan-penyimpangan pola. Untuk menghindari hal ini dapat dilakukan dengan cara mereduksi pola tekstur yang digunakan ketika ukuran objek yang ditampilkan ukurannya berkurang.

Secara khusus, masing-masing reduksi pola adalah setengah dari ukuran sebelumnya. Sebagai contoh, jika memiliki pola dua dimensi 16x16, dapat dirubah menjadi empat pola dengan ukuran pengurangan 8x8, 4x4, 2x2, dan 1x1. Untuk suatu sudut pandang tertentu, suatu pola yang direduksi dapat diterapkan untuk memperkecil penyimpangan. Pola pengurangan ini direferensikan kedalam MIP *maps* atau *mip maps.*

E. OpenGL

OpenGL (*Open Graphics Library*) adalah standar API yang dapat digunakan untuk membuat aplikasi berbasis grafik, baik dua dimensi (2D) maupun tiga dimensi (3D). *OpenGL* ini bersifat *cross-platform*, artinya dapat dijalankan pada berbagai *platform* sistem operasi yang ada saat ini.

Keuntungan dari pendekatan ini adalah bahwa hal itu memungkinkan fleksibilitas yang besar dalam proses menghasilkan gambar. Aplikasi ini gratis untuk *trade-off rendering* kecepatan dan kualitas gambar dengan mengubah langkah-langkah di mana foto tersebut diambil. Cara termudah untuk menunjukkan kekuatan dari antarmuka prosedural adalah untuk dicatat bahwa antarmuka deskriptif dapat dibangun di atas antarmuka prosedural, tetapi tidak sebaliknya. Pikirkan *OpenGL* sebagai "bahasa *assembly* grafis": potongan-potongan fungsi *OpenGL* dapat dikombinasikan sebagai building blocks untuk menciptakan teknik inovatif dan menghasilkan kemampuan baru grafis.

Aspek kedua adalah bahwa spesifikasi *OpenGL* tidak tepat piksel. Ini berarti bahwa dua implementasi yang berbeda *OpenGL* sangat tidak mungkin untuk membuat gambar yang sama persis. Hal ini memungkinkan *OpenGL*

untuk diimplementasikan di berbagai platform perangkat keras.

Jika spesifikasi terlalu tepat, hal itu akan membatasi jenis akselerasi perangkat keras yang dapat digunakan; membatasi kegunaannya sebagai standar. Dalam prakteknya, kurangnya ketepatan tidak perlu menjadi beban - kecuali jika Anda berencana untuk membangun sebuah peternakan rendering dari berbagai jenis mesin.

Kurangnya ketepatan piksel muncul bahkan dalam pelaksanaan tunggal, dalam *path* yang berbeda melalui implementasi mungkin tidak menghasilkan set yang sama fragmen, meskipun spesifikasi tidak mandat seperangkat aturan invarian untuk menjamin perilaku berulang di berbagai keadaan.

III. METODOLOGI PENELITIAN

A. Bahan dan Peralatan

Dalam mengerjakan tugas akhir ini mulai dari mendesain sampai tahap pemrograman, *software* visualisasi ini membutuhkan spesifikasi *hardware* dan *software* tertentu untuk bisa berjalan optimal. Meskipun *OpenGL* didukung oleh berbagai *platform hardware* dan *software*, dalam implementasi kali ini penulis menggunakan perlengkapan *hardware* dan *software* sebagai berikut:

Spesifikasi *Hardware*

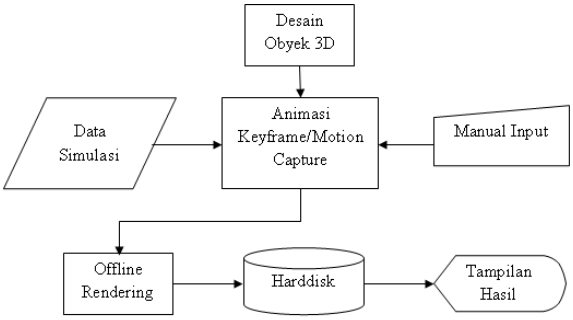
1. Mainboard ASUS P5KPL SE Intel Platform
2. Processor Intel® Core™ 2 Duo E7400 @ 2.80 GHz
3. VGA NVIDIA GeForce 9500GT 512MB
4. RAM DDR2 2GB
5. Harddisk 1TB HDD

Spesifikasi *Software*

1. Sistem operasi Windows 7 Ultimate SP1 32bit
2. Driver OpenGL versi 3.7.6
3. Microsoft Visual Studio 2008 sebagai media compile C++

B. Perancangan Sistem

Secara umum proses untuk menghasilkan visualisasi dan animasi tiga dimensi (3D) dari suatu kasus simulasi adalah dengan cara membangun model-model objek 3D di sebuah *software 3D authoring*, menganimasikan menggunakan metode *Keyframe* atau *Motion Capture* sesuai data atau parameter simulasi yang ada, me-*render* menjadi gambar atau video.



Gambar 5. Sistem visualisasi tidak *real time*

Gambar 5 menunjukkan proses menganimasikan dan *render* dalam metode ini memakan waktu cukup panjang bisa beberapa menit, jam atau bahkan hari. Hal ini tidak bisa diterima jika diperlukan pengambilan keputusan secara cepat saat data atau parameter sudah tersedia. Gambar 6 menunjukkan sistem yang dibuat dengan menghilangkan animasi manual serta proses *rendering* yang memakan waktu dengan memanfaatkan teknologi *real time rendering*.

C. Instalasi Driver OpenGL

Terlebih dahulu *download file-file library OpenGL* yang terdapat pada situs resminya di :
http://www.opengl.org/resources/libraries/glut/glut_downloads.php

Untuk sistem operasi *Windows*, *library* ini terdiri dari 3 *files* yaitu:

1. glut.h
2. glut32.lib
3. glut32.dll

Dan berikut ini cara instalasinya:

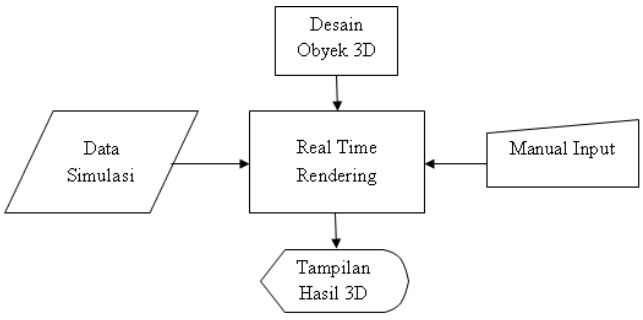
1. Salin file glut.h di dalam folder C:\Program Files\Microsoft Visual Studio 9.0\VC\include\GL
2. Salin file glut32.lib di dalam folder C:\Program Files\Microsoft Visual Studio 9.0\VC\lib
3. Salin file glut32.dll di dalam folder C:\Windows\System32

IV. HASIL DAN PEMBAHASAN

A. Cara Kerja OpenGL

OpenGL lebih mengarah pada prosedural daripada sebuah deskriptif API grafis. Untuk mendeskripsikan *scene* dan bagaimana penampilannya, sebenarnya *programer* lebih tahu untuk menentukan hal-hal yang dibutuhkan untuk menghasilkan efek yang diinginkan. Langkah tersebut termasuk memanggil banyak perintah *OpenGL*, perintah tersebut digunakan untuk menggambarkan grafis primitif seperti titik, garis dan poligon dalam tiga dimensi. Sebagai tambahan, *OpenGL* mendukung *lighting*, *shading*, *texture*, *mapping*, *blending*, *transparancy*, dan banyak kemampuan efek khusus lainnya.

OpenGL mempunyai banyak fungsi dan penggunaan perintah yang sangat luas. Penggunaan *OpenGL* membutuhkan *library* tambahan yang harus di letakkan pada direktori sistem dari *windows* (OS).



Gambar 6. Sistem visualisasi *real time*

B. Inisialisasi Awal

Inti dari tahapan ini adalah mengatur *viewport* dan perspektif untuk penampilan objek ke dalam layar monitor, *viewport* adalah besarnya layar monitor (*image*) yang dipakai untuk menampilkan objek, sedangkan perspektif yang dimaksud adalah pengaturan sumbu z dalam penampilan objek 3 dimensi, sehingga *user* dapat melihat objek seolah-olah dalam bidang 3 dimensi (x-y-z). Selain itu penggambaran objek yang dilakukan oleh *programmer* juga dapat menggunakan koordinat 3 dimensi. Selain ke dua tujuan di atas pada tahap ini juga dilakukan koneksi awal dengan *library OpenGL*. Koneksi ini dilakukan supaya fungsi-fungsi yang disediakan *OpenGL* dapat digunakan. Tabel I menunjukkan fungsi-fungsi dalam *OpenGL*.

TABEL I. FUNGSI *OPENGL*

Fungsi / perintah dalam *OpenGL*	**Kegunaan**
LoadGlut('glut32.dll)	Pemanggilan *library OpenGL*
InitGL	Inisialisasi *OpenGL* awal yang harus dilakukan
glViewport	Untuk pengaturan besarnya layar monitor yang dipakai
glMatrixMode	Pengaturan *viewport*
gluPerspective	Pengaturan sumbu z dalam penampilan objek 3 dimensi
glPointSize(Glfloat Size);	Menentukan besar titik
glBegin(GL_POINTS) … glEnd();	Menggambar titik
glColor3f(…);	Menentukan warna
glVertex3f(…);	Menentukan posisi sebuah titik
glTranslatef(…);	Mengubah posisi titik pusat sumbu koordinat
glPushMatrix(); … glPopMatrix();	Membuat baris kode diantaranya menjadi tidak berlaku untuk bagian luar
glLineWidth(GLfloat Width);	Menentukan lebar garis
glBegin(GL_POINTS); … glEnd();	Menggambar titik
glBegin(GL_LINES); … glEnd();	Menggambar garis
glBegin(GL_LINE_STRIP); … glEnd();	Menggambar garis terhubung
glBegin(GL_LINE_LOOP); … glEnd();	Menggambar garis terhubung
glBegin(GL_TRIANGLES); … glEnd();	Menggambar segitiga
glBegin(GL_TRIANGLE_STRIP); … glEnd();	Menggambar segitiga terhubung
glBegin(GL_TRIANGLE_FAN); … glEnd();	Menggambar segitiga terhubung
glBegin(GL_QUADS); … glEnd();	Menggambar empat garis yang menginterprestasikan sebuah polygon

C. Pembuatan Objek

Didalam *OpenGL* pembuatan objek dilakukan dengan titik-titik 3 dimensi, dengan mode GL_QUARDS, maka otomatis setiap 4 titik digambar menjadi sebuah bidang segi empat, sedangkan mode GL_LINES, pada setiap 2 titik digambar menjadi sebuah garis. Didalam tahap ini setiap garis atau bidang juga dapat diatur warnanya. Fungsi yang digunakan glColor3f.

glColor3f(1.0, 0.0, 0.0, 0.0);

Contoh di atas menghasilkan warna merah. Karena dalam pewarnan *OpenGL* menggunakan format RGBα.

Pada *OpenGL* utuk memiliki suatu pemrograman grafik sangat sederhana, hanya memerlukan empat fungsi yang telah disediakan oleh *OpenGL*, fungsi-fungsi tersebut adalah:

1. glutInitWindowSize
   Fungsi ini untuk membuat *window* yang akan digunakan untuk menampilkan objek-objek 3D, format fungsi ini adalah:
   void glutInitWindowSize(init width, init height);
   void glutInitWindowSize(500, 350);
   width adalah lebar dari *window* yang akan dibuat (dalam *pixel*), sedangkan *height* adalah tinggi dari *window* (dalam *pixel*).
2. glutInit
   fungsi untuk memberi tahu MS VC++ bahwa *OpenGL library* dipakai pada program, format fungsi ini adalah:
   *void glutInit(init *argcp, char **argv);*
   argcp dan argv adalah parameter identik yang dipakai oleh fungsi main() pada C++, jadi pastikan bahwa parameter ini sama dengan parameter yang ada fungsi main() pada setiap program grafik 3D.
3. glutInitDisplayMode
   Fungsi yang dipakai untuk menginisialisasi model dari tampilan (*display mode*), yang mempunyai format:
   void glutInitDisplayMode(unsigned int mode);
   variable mode diisi dengan:
   GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH
   Pilihan untuk *variable mode* amat sangat bervariasi, tetapi program hanya akan menggunakan tiga kombinasi diatas. GLUT_DOUBLE berarti menggunakan *buffer* pada *window* sebesar dua kali. GLUT_RGB menggunakan *Red Green Blue Alpha* untuk pewarnaan. GLUT_DEPTH berarti menggunakan *depth buffer* agar objek 3D yang ditampilkan akan terlihat lebih nyata. Untuk mengkombinasikan *variable mode* yang dipakai gunakan garis *vertical*(|).
4. glutCreateWindow
   Fungsi ini dipakai untuk membuat/meng-*create window*, mempunyai format sebagai berikut:
   Int glutCreateWindow (char name);*
   Name adalah nama dari *window* yang akan dibuat.

Dengan memakai empat fungsi tersebut maka telah berhasil melakukan inisialisasi pada *OpenGL*.

D. Hasil Simulasi

Aplikasi pembuatan berupa simulasi pesawat terbang. Dengan melakukan beberapa pembentukkan objek-objek seperti badan pesawat, baling-baling depan, sayap, roda pesawat, dan sayap belakang. kemudian di gabung menjadi satu bagian pesawat. Adapun pembentukan tanah rumput dan jalan.

Dalam pembuatan objek, tiap-tiap bagian pesawat dibuat terpisah lalu digabungkan menjadi satu. Misalnya bagian badan pesawat *scriptnya* sebagai berikut:

```
…
{
GLfloat bodyleft[][3] = {{ 0 , 60,-260},
                         {-5 , 60,-230},
                         {-5 , 20,-200},
                         {-20, 10,-70 },
                         {-30, 30, 50 },
                         {-30, 10, 70 },
                         {-10, 0 , 140},
                         {-10,-20, 140},
                         {-30,-40, 110},
                         {-30,-50, 90 },
}
```

Fungsi yang digunakan adalah *GLfloat bodyleft[][3]* dan *GLfloat bodyright[][3]* yang berfungsi untuk penempatan titik pada sumbu koordinat x, y dan z.

Setelah menempatkan titik-titik koordinat selanjutnya membuat garis dengan cara menghubungkan koordinat tiap titik menjadi bidang poligon.

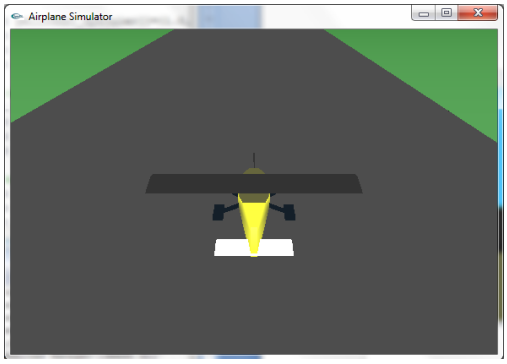
```
void drawbody()
    {
        lbodypolygon5(5,6,7,8,9);
        lbodypolygon5(9, 10, 13, 4, 5);
        lbodypolygon3(10,3,13);
        lbodypolygon5(10, 11, 12, 2, 3);
        lbodypolygon4(12,2,1,0);
        rbodypolygon5(5,6,7,8,9);
        rbodypolygon5(9, 10, 13, 4, 5);
        rbodypolygon3(10,3,13);
        rbodypolygon5(10, 11, 12, 2, 3);
    }
```

Untuk pemberian warna pada badan pesawat yang menggunakan warna kuning *scriptnya* sebagai berikut

```
…
//glColor3f(1.0,1.0,0.0); warna kuning untuk fitur pesawat
GLfloat yellowmat_specular[]={1.0, 1.0, 0.5, 1.0};
GLfloat yellowmat_diffuse[]={1.0, 1.0, 0.0, 1.0};
GLfloat yellowmat_ambient[]={1.0, 1.0, 0.6, 1.0};
GLfloat yellowmat_shininess={32.0};
...
```

glColor3f(1.0,1.0,0.0); adalah warna kuning. Dan fungsi lainnya berupa pencahayaan dengan mengatur arah cahaya.

Hasil *compile* dari program ini ditunjukkan pada gambar 7, dimana ada beberapa tampilan halaman untuk antarmuka kemudian dilakukan pengujian gerakan terhadap objek.



Gambar 7. Pesawat dilihat dari arah atas bagian belakang

Penempatan arah kamera dari atas bagian belakang pesawat.

```
// kamera dari atas
case '5':
        floatcamera = 0;
        insidecamera = 0;
        eyex = 0.0;
        eyey = 750.0;
        eyez = -2000.0;
        atx = 0.0;
        aty = 0.0;
        atz = 0.0;
        upx = 1.0;
        upy = 0.0;
        upz = 0.0;
break;
```

Pada sumbu y *eyey = 750.0* dan sumbu z *eyez = -2000.0* dari titik 0.0.0.

V. KESIMPULAN

1. Pemanggilan dan penggunaan titik-titik vektor akan berpengaruh terhadap bentuk objek yang dibuat.
2. Gerakkan dari simulasi akan sesuai dengan variabel yang dikirimkan/diterima karena simulasi yang dibuat terdapat sistem yang melakukan perbandingan antara variabel yang dikirimkan/diterima dengan variabel pada objek.
3. Gerakkan perpindahan objek yakni pesawat tidak terganggu karena adanya latar berupa rumput dan jalan, karena pada pembuatan simulasi perintah yang digunakan adalah *glPushMatrix();* dan *glPopMatrix();* sehingga tiap bagian tersimpan koordinat masing-masing.
4. Pemakaian jenis *Graphics Card* yang tidak menggunakan 3D akan mempengaruhi kinerja dari simulasi yang akan mengakibatkan pergerakan objek menjadi lambat.
5. *OpenGL* merupakan status mesin dalam proses *rendering* dan atribut-atribut diubah melalui pemanggilan prosedur. Didesain untuk mengakomodasikan teknik *rendering* grafis tingkat lanjut, seperti *texture mapping*, anti-aliasing, transparasi, pencahayaan dan transformasi objek 3D.

DAFTAR PUSTAKA

[1] Andi, "*Pemrograman Microsoft Visual C++*", Wahana Komputer, Yogyakarta dan Semarang, 2009.

[2] D. Astle & K. Hawkins, "*Beginning OpenGL Game Programming*", Thomson Course Technology, USA, 2004.

[3] Addison, "*OpenGL Programming Guide*", Wesley Publishing Company.

[4] B. Achmad & R. Nana, "*Pengantar Grafika Komputer*", Politeknik Elektronika Negeri Surabaya.

[5] OpenGL Programming, Resources and Headline News, tersedia di : http//www.opengl.org

[6] OpenGL Programming Guide, The Official Guide to Learning OpenGL, tersedia di : http//www.glprogramming.com/red/

[7] Tutorial OpenGL, tersedia di : http//nehe.gamedev.net/