ISSN : 2356-0010

JURNAL KomTekInfo FAKULTAS ILMU KOMPUTER UNIVERSITAS PUTRA INDONESIA YPTK PADANG

VOLUME 1 NOMOR 1 JUNI 2014 ISSN : 2356-0010

FAKULTAS ILMU KOMPUTER
UNIVERSITAS PUTRA INDONESIA YPTK PADANG
Jln. Raya Lubuk Begalung Padang, Telp. (0751) 7055555,
Fax. (0751) 91913, Email: fppti_upi_yptk@yahoo.com

ISSN : 2356-0010

JURNAL KOMTEKINFO FAKULTAS ILMU KOMPUTER UNIVERSITAS PUTRA INDONESIA YPTK PADANG

VOLUME 1 NOMOR 1 JUNI 2014

FAKULTAS ILMU KOMPUTER
UNIVERSITAS PUTRA INDONESIA YPTK PADANG
Jln. Raya Lubuk Begalung Padang, Telp : 0751-776666,
Fax. 0751-71913. Email : lppm_upi_yptk@yahoo.com

JURNAL KOMTEKINFO FAKULTAS ILMU KOMPUTER
UNIVERSITAS PUTRA INDONESIA YPTK PADANG
JUNI 2014 VOL. 1 NO. 1 ISSN : 2356-0010

DAFTAR ISI

IMPLEMENTASI OPEN GL32 UNTUK MEMANIPULASI GAMBAR SEGITIGA DAN SEGIEMPAT

Agung Slamet Riyadi, Universitas Gunadarma,
Jl. Margonda Raya No. 100 Pondok Cina Depok Jawa Barat
e-mail : agungsr@staff.gunadarma.ac.id

Abstrak - Pada dunia komputer grafik, pembuatan model tiga dimensi untuk simulasi permainan 3D saat ini menjadi trend sejalan dengan perkembangan perangkat lunak dan perangkat keras saat ini. Besrrbagai industri terutama yang bergerak di bidang periklanan dan permainan (*games*) sangat membutuhkan aplikasi grafik ini. Aplikasi grafik berbasis 3D dapat dibuat melalui perantara API (*Application Programming Interface*) yang dapat menjembatani antara aplikasi dan kartu grafis. OpenGL adalah salah satu contoh dari *Application Programming Interface* tersebut. Pada makalah ini dijelaskan beberapa fungsi yang terdapat pada OpenGL, diantaranya adalah untuk membuat, mewarnai dan mengatur posisi dari suatu polygon segitiga (*Triangle*) dan polygon segiempat (*Quadrilateral*). Penggunaan fungsi-fungi akan dilengkapi dengan contoh-contoh program sederhananya.

Kata Kunci : *Application Programming Interface*, OpenGL, *Triangle* dan *Quadrilateral*

1 Pendahuluan

Bahasa pemrograman OpenGL merupakan sebuah library untuk pemrograman grafik (*Graphics Programming*). Dasar untuk mempelajari *Graphics Programming* adalah terletak pada fungsi matematikanya, terutama operasi matriks. Pada *Graphics Programming* yang perlu dipelajari adalah mengenai *shading*, shape, *transform* (*rotate*, *translation*, *scala*). OpenGL (*Open Graphics Library*) adalah Sebuah antarmuka pemrograman aplikasi (API) render yang menghasilkan gambar berwarna berkualitas tinggi dan mendefinisikan sebuah *cross platform* API untuk menulis aplikasi yang menghasilkan gambar dalam bentuk 2D dan 3D. *Source code* untuk pemrograman OpenGL dibuat dengan menggunakan Bahasa Pemograman C++. Dalam pembuatan Program OpenGl nantinya akan dirancang 2 macam objek gambar yaitu polygon segitiga (*Triangle*) dan polygon segiempat (*Quadrilateral*). Dari pembuatan polygon tersebut maka akan dilakukan manipulasi gambar yaitu akan diberi warna dan perubahan bentuk objek polygon Proses pewarnaan (*coloring*) objek polygon akan dilakukan pada objek gambar dan diluar objek tersebut (*background*). Sedangkan proses bentuk akan dilakukan pergesaran tempat objek polygon dan perubahan ukurannya.

2 Tinjauan Pustaka

2.1 OpenGL

OpenGL (*Open Graphics Library*) adalah spesifikasi standar yang mendefinisikan sebuah *cross* bahasa, *cross* platform API untuk menulis aplikasi yang menghasilkan komputer 2D dan 3D grafis. Terdiri dari lebih dari 250 panggilan fungsi yang berbeda yang dapat digunakan untuk menggambar tiga dimensi yang komplek. OpenGL dikembangkan oleh Silicon Graphics Inc (SGI) pada tahun 1992 dan secara luas digunakan dalam CAD, *Virtual Reality*, Visualisasi Ilmiah, Visualisasi Informasi, dan Simulasi Penerbangan [1].

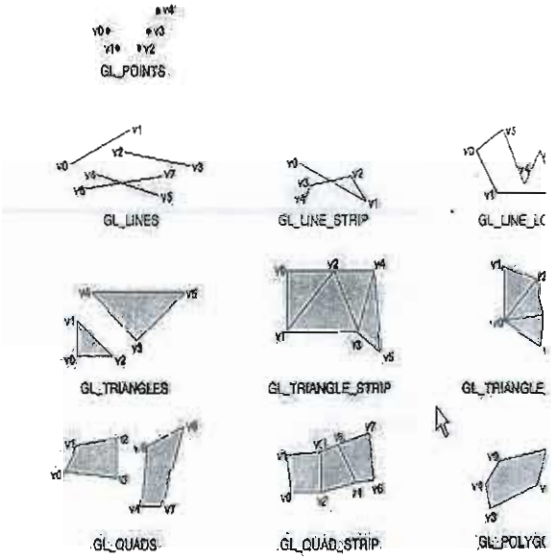
OpenGL diproduksi oleh Silicon Graphics, Inc (SGI) dan pada awalnya ditujukan hanya untuk sistem komputer mereka, tetapi dalam perkembangannya, OpenGL diterima menjadi salah satu bakuan (standard) dalam grafika komputer dan saat ini telah diimplementasikan dalam berbagai sistem komputer. OpenGL merupakan pustaka program (program library) yang menyediakan sejumlah perintah yang berhubungan dengan grafika. Perintah glBegin (mode) mengawali perintah menggambar. Mode merupakan konstanta yang menyatakan bagaimana OpenGL harus terhubung titik (vertex) yang akan digambar.

Perintah glBegin (mode) mengawali perintah menggambar. Mode merupakan konstanta yang menyatakan bagaimana OpenGL harus terhubung titik (vertex) yang akan digambar. Mode yang dapat digunakan diperlihatkan pada table 1, dibawah ini :

Tabel 1: Mode pada GLBegin

Nilai	Arti
GL_POINTS	Setiap titik diperlakukan sebagai titik terpisah
GL_LINES	Dua pasang diperlakukan sebagai garis
GL_LINE_S	Sama seperti GL_LINES tetapi
GL_LINE_L OOP	Sama sepeti GL_LINE_STRIP tetapi titik pertama dan terakhir
GL_TRIAN GLES	Tiga pasang titik dianggap sebagai bidar segitiga
GL_TRIAN	Bidang segitiga yang saling
GL_TRIAN GLES_FAN	Mirip GL_TRIANGLE_STRIP tetapi semua bidang menggunakan satu
GL_QUADS	Empat titik dianggap sebagai empat polygon-empat-sisi (quadrilaterals)
GL_QUAD_ STRIP	Pasangan quadrilaterals
GL_POLYG ON	titik dianggap sebagai titik sudut polygon

Pada gambar 1 di bawah ini menjelaskan bahwa mode menggambar pada OpenGL dan bagaimana tata urutan titik-titik dihubungkan sehingga bentuk tertentu.



Gambar 1: Mode gambar pada OpenGL

2.2 GLUT

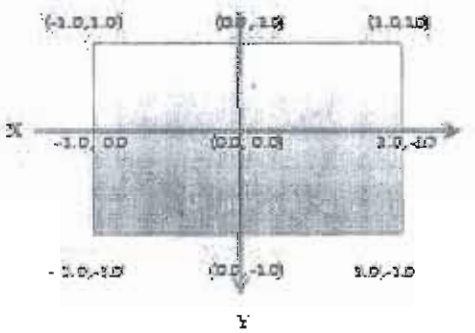
OpenGL tidak menyediakan fungsi fungsi untuk manajemen antar muka dan interaksi dengan pengguna. Hal ini dikarenakan setiap sistem operasi memiliki fungsi tersendiri untuk menangani OpenGL, misalnya GLX untuk Linux, Wiggle untuk Microsoft Windows, AGL, NSOpenGl, CGL untuk MacOSX. Karena fungsi ini spesifik untuk system operasi tertentu, maka membuat program yang ditulis menjadi tidak multi platform. Untuk menghindari hal ini maka dapat digunakan GLUT (*OpenGL Utility Toolkit*) untuk membuat antarmuka yang independen [2] .

2.3 Konsep Vertex Pada OpenGL

Obyek dalam komputer grafis mencirikan bentuk dan dimensi. Angkanya mengacu ke beberapa sistem koordinat, paling sering Cartesian koordinat x, y, dan z. Kita sering membutuhkan lebih dari satu sistem koordinat, misalnya suatu sistem koordinat lokal untuk menggambarkan setiap bagian dari mesin, yang kemudian dapat dihubungkan dengan menetapkan hubungan dari masing-masing koordinasi sistem lokal satu sama lain. Terkadang benda menunjukkan simetri tertentu, sehingga hanya bagian dari itu yang perlu dijelaskan, dan sisanya dibangun dengan merefleksikan, berputar dan menerjemahkan bagian asli.

Seorang desainer mungkin ingin melihat dari titik pandang yang berbeda dengan memutar objek. Dalam animasi, satu atau lebih obyek harus bergerak relatif terhadap satu sama lain, sehingga koordinasi sistem harus digeser dan diputar sebagai hasil animasi.

Pada pemrograman OpenGL untuk mengatur koordinat digunakan perintah glVertex, dengan cara : glVertex3f (x,y,z). X menyatakan untuk koordinat sumbu x, Y untuk koordinat sumbu Y dan Z untuk koordinat sumbu Z. Seperti dapat dilihat pada contoh gambar 2 berikut ini :



Gambar 2: Koordinat Untuk Pengaturan Polygon

Berdasarkan gambar 2 maka dapat dibuat coding program untuk membuat sebuah polygon segiempat sebagai berikut :

```
Sc
ri
pt
:
glVertex3f(-1.0f,
1.0f, 0.0f); // Top
Left
glVertex3f( 1.0f, 1.
0f, 0.0f); // Top Ri
ght
glVertex3f( 1.0f,-1.0f
, 0.0f); // Bottom Righ
t glVertex3f(-1.0f,-1
.0f, 0.0f); // Bottom
Left
```

Untuk membuat sebuah polygon segitiga maka coding programnya adalah sebagai berikut :

```
Sc
ri
pt
:
glVertex3f(0.0f, 1.0
f, 0.0f); // Top Cen
tre
glVertex3f(-1.0f,-1.
0f, 0.0f); // Bottom
Left
glVertex3f( 1.0f,-1.0f
, 0.0f); // Bottom Righ
t
```

atau
Scrips :

```
Sc
ri
pt
:
glVertex3f( 1.0f,-1.0f
, 0.0f); // Bottom Cent
reglVertex3f(-1.0f, 1.
0f, 0.0f); // Top Left
glVertex3f( 1.0f, 1.
0f, 0.0f); // Top Ri
ght
```

2.4 Konsep Warna

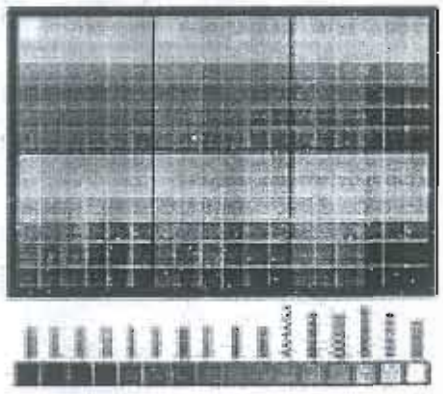
Warna adalah sensasi yang terjadi ketika energi cahaya, insiden pada retina ditafsirkan oleh otak. Untuk Komputer Grafis warda terdiri dari : Warna Visi, Warna Reproduksi dan Warna Sintesis. Manusia memiliki tiga jenis kerucut menanggapi wavelenghts cahaya yang berbeda yaitu ; pendek, menengah atau panjang dari ukuran gelombang. Atau ditinjau dari sudut warna adalah Merah, Hijau dan Biru. Kerucut menyerap cahaya dan mengirimkan sinyal ke otak.

Banyak sistem terbatas pada 256 warna walaupun 24-bit citra RGB tersedia. Kemudian dibentuklah kumpulan warna RGB (dapat digunakan pada semua sistem: *all systems safe*) dari 256 warna tersebut, 40 warna diproses dengan cara yang berbeda oleh bermacam OS, sisanya tinggal 216 warna yang berlaku umum bagi semua sistem. 216 warna ini telah menjadi standar de facto untuk safe colors, terutama untuk aplikasi internet. Setiap 216 warna ini terdiri dari 3 komponen RGB, tapi masing-masing hanya boleh bernilai 0,51,102, 153, 204, 255, dapat dilihat pada tabel 2 berikut :

Tabel 2: Nilai setiap komponen RGB dalam warna yang aman.

Number System	Color Equivalents					
Hex	00	33	66	99	CC	FF
Decimal	0	51	102	153	204	255

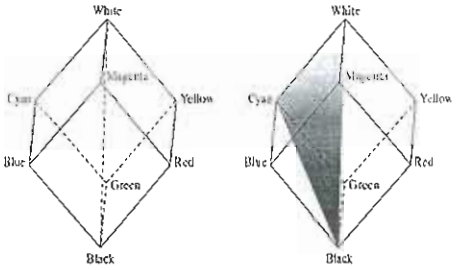
Warna merah murni: FF0000, biru murni: 0000FF, hitam: 000000, putih: FFFFFF. Seperti pada gambar 2 berikut ini :



Gambar 2. Kumpulan warna

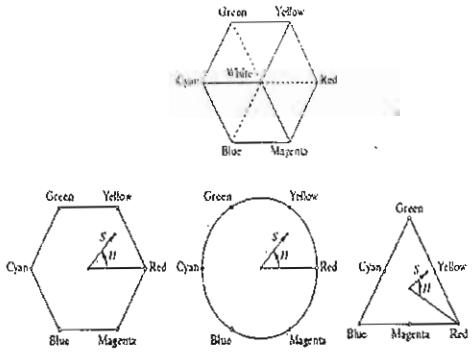
Pada gambar di atas Semua abu-abu di 256 warna RGB system (abu-abu yang merupakan bagian dari kelompok warna yang aman ditunjukkan garis bawah
Model CMY Asumsikan semua nilai warna dinormalisasi menjadi [0,1] Model CMY digunakan untuk membuat output hardcopy CMYK. K adalah warna keempat: hitam; karena CMY yang dicampur tidak dapat menghasilkan warna hitam pekat, sedangkan seringkali kita harus mencetak dengan warna hitam pekat. Rumusan: C = 1 R M = 1 G Y = 1 B
Model HSI RGB dan CMY tidak cocok untuk mendeskripsikan colors berdasarkan interpretasi manusia Hue (H), Saturation

(S), Intensitas (I) Hue: mendeskripsikan warna murni Saturasi: derajat banyaknya warna murni dilunakkan dengan warna putih Intensitas: menggabungkan informasi warna dari H dan S. I (intensity) garis yang menghubungkan titik *black* dan *white* Semua titik pada garis ini adalah abu-abu.

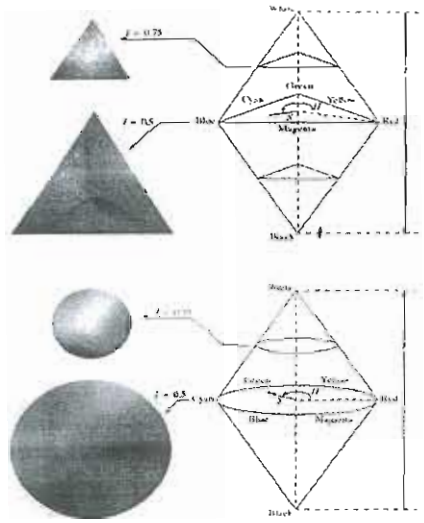


Gambar 3: Konseptual hubungan antara RGB dan Model warna HSI

H (hue) semua titik pada bidang yang dibatasi oleh titik *black*, *white* dan warna-x, memiliki hue yang sama, yaitu warna-x. Contoh pada gambar sebelumnya: warna-x: cyan S (saturasi) untuk menentukan saturasi (kemurnian) dari warna-x: buat bidang dari titik warna-x tegak lurus dengan sumbu intensitas dan memiliki hue yang sama. Saturasi adalah jarak terdekat antara titik warna-x dengan sumbu intensitas.



Gambar 4: Bagan warna dan saturasi dalam model warna HSI.



Gambar 5: Proses warna dan saturasi dalam model warna HSI

Model YIQ YIQ (Y untuk pencahayaan, saya dalam fase dan Q untuk quadrature), seperti YUV, adalah ruang warna yang digunakan dalam sinyal televisi. YIQ digunakan terutama oleh standar televisi NTSC untuk informasi encoding warna. Komponen Y, seperti di YUV, digunakan untuk mengkodekan informasi pencahayaan, dan merupakan satu-satunya komponen yang digunakan oleh hitam-putih penerima televisi.

3 Metode Penelitian

Metodologi yang digunakan dalam pembuatan objek gambar segitiga dan segiempat ini adalah sebagai berikut :

1. Studi literatur mempelajari tentang bagaimana proses penetuan warna dan titik atau vertex. Studi untuk mempelajari pengontrolan koordinat dalam ruang dalam domain tiga dimensi juga diperlukan untuk proses visualisasi.
2. Pembuatan dan analisa program Proses pembuatan program dibagi ke dalam beberapa tahapan : membuat blok program untuk pengontrolan window OpenGL, membuat blok program untuk melakukan proses penentuan titik dan warna objek. Membuat blok program untuk penerapan kontrol visualisasi tiga dimensi dan juga menampilkan hasil ke layar.
3. Pengujian dan simulasi program Langkah selanjutnya adalah menguji program atau aplikasi yang telah dibuat, dan mengamati hasil yang didapatkan.

4 Hasil Dan Pembahasan

4.1 Analisis dan Simulasi Program Pembuatan Objek dan Warna Dasar

Merupakan penjelasan dalam tahapan buatan objek gambar (polygon) dan warna tersebut, sebagai berikut :

1. Pendefinisian dan pemanggilan *library header* file untuk : OpenGL32 Library, GLU32 Library, dan Glaux Library.

```
Sc
ri
pt
:
#include <gl.h>// Header
File For The OpenGL32 Lib
rary
#include <glu.h>// Hea
der File For The GLu32 L
ibrary
#include <glaux.h>// He
ader File For The GlauxL
ibrary
```

2. Pemberian warna dasar hitam untuk latar tampilan (background)

```
Sc
ri
pt
:
int InitGL(GLvoid)// All
Setup For OpenGL Goes Here
{
glShadeModel(GL_SMOOTH);
// Enable Smooth Shading
glClearColor(0.0f, 0.0f,
0.0f, 0.5f);// Black Backg
round
glClearDepth(1.0f);//
Depth Buffer SetupglE
nable(GL_DEPTH_TEST);
// Enables Depth Testi
ng
glDepthFunc(GL_LEQUAL);/
/ The Type Of Depth Testing
To Do glHint(GL_PERS
PECTIVE_CORRECTION_HINT,
GL_NICEST);// Really Nice
PerspectiveCalculations
     return TRUE; // Init
     ialization Went OK
}
```

3. Penentuan objek tampilan pada layar monitor (view) baik lebar dan tinggi objek dari gambar objek tersebut secara perspektif.

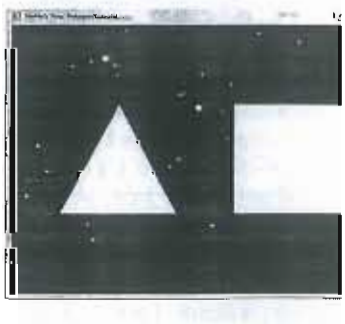
```
Script :
Lvoid ReSizeGLScene(GLsiz
ei width, GLsizeiheight)/
/ Resize And Initialize
The GL Window
{
if(height==0)//Prevent A D
ivide By Zero By
{
height=1;// Making H
eight Equal One
     }
glViewport(0,0,width,heigh
t);// Reset The Current View
port
glMatrixMode
(GL_PROJECTION);// Select
The ProjectionMatrix
glLoadIdentity();// Reset
The Projection Matrix// Ca
lculate The Aspect Ratio Of
The Window
gluPerspective(45.0f,(GLfl
oat)width/(GLfloat)height,
0.1f,100.0f);
glMatrixMode
(GL_MODELVIEW); // Select
The ModelviewMatrix
glLoadIdentity();// Reset
The Modelview Matrix }
glViewport(0,0,width,heigh
t);// Reset The Current View
port
glMatrixMode
(GL_PROJECTION);// Select
The ProjectionMatrix
glLoadIdentity();// Reset
The Projection Matrix
// Calculate The Aspect Rat
io Of The Window
gluPerspective(45.0f,(GLfl
oat)width/(GLfloat)height,
0.1f,100.0f);
glMatrixMode
(GL_MODELVIEW);// Select
The ModelviewMatrix
glLoadIdentity();//Reset
The Modelview Matrix
```

4. Penentuan titik-titik (vertex) polygon, yaitu triangle dan quadrilateral

```
Script :
int DrawGLScene(GLvoid)//He
re's Where We Do All The Dra
wing
{
gl Clear
   (GL_COLOR_BUFFER_BIT |
   GL_DEPTH_BUFFER_BIT) ;
   // Clear Screen And
   Depth Buffer
glLoadIdentity();//Reset
```

```
The Current Modelview Matri
x
glTranslatef(-1.5f,0.0f,-
6.0f);//Move Left 1.5 Units
And Into The Screen 6.0
glBegin(GL_TRIANGLES);
// Drawing Using Trian
glesglVertex3f( 0.0f, 1.
0f, 0.0f);//Top
glVertex3f(-1.0f,-1.0f, 0.
0f);// Bottom Left
glVertex3f( 1.0f,-1.0
f, 0.0f);// Bottom Ri
ght glEnd();// Finishe
d Drawing The Triangl
e
glTranslatef(3.0f,0.0f,0.0f
);
// Move Right 3 Units
glBegin(GL_QUADS);// Draw
A Quad
glVertex3f(-1.0f, 1.0f, 0.
0f);// Top Left
glVertex3f( 1.0f, 1.0f, 0.0
f);// Top Right
glVertex3f( 1.0f,-1.0
f, 0.0f);// Bottom Ri
ght glVertex3f(-1.0f,
-1.0f, 0.0f);//
Bottom Left
glEnd();//
Done Drawing
The Quad return
TRUE;
// Keep Going
```

Hasil dari simulasi pada pembuatan program terhadap objek polygon tersebut, adalah seperti pada gambar 6 berikut ini :



Gambar 6: Obyek segitiga dan segiempat

Tampilan hasil akhir program bagian-1, membuat 2 buah objek polygon, Triangle dan Quadrilateral dengan warna putih (*white*) dan warna latar hitam (*black*).

4.2 Analisis dan Simulasi Program Perubahan Warna pada Bidang datar Objek Polygon

Untuk menambah objek polygon segitiga dan polygon segiempat maka kembali dgunakan perintah atau statemen glBegin (GL_TRIANGLES) dan glBegin (GL_QUADS). Untuk lebih lengkapnya dapat dilihat pada script program berikut ini :

S

c

r

i

p

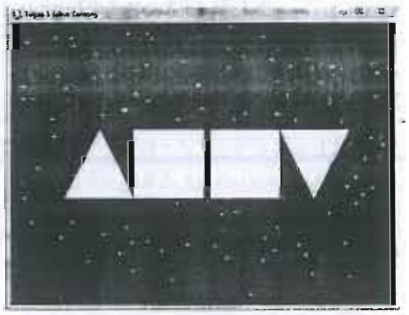
t

:

```
glTranslatef(-3.0f,0.0f
,-10.0f);// Move Left 3
.0
    Units And Into The Scree
    n 10.0
glBegin(GL_TRIANGLES);
      // Drawing Using
      TrianglesglVertex3
      f( 0.0f, 1.0f, 0.0f
      );// Top
      glVertex3f(-1.0f,-1
      .0f, 0.0f);// Bottom
      Left glVertex3f( 1.0f
      ,-1.0f, 0.0f);//
      Bottom Right
glEnd();// Finishe
d Drawing The Tria
ngle
glTranslatef(2.0f,0.0f,0.
0f);// Move Right 2 Unit
sglBegin(GL_QUADS);//
Draw A Quad
      glVertex3f(-1.0f, 1.
      0f, 0.0f);// Top Left
      glVertex3f( 1.0f, 1.
      0f, 0.0f);// Top Ri
      ght glVertex3f( 1.0f
      ,-1.0f, 0.0f);//
      Bottom
          Right
      glVertex3f(-1.0f,-1
          .0f, 0.0f);//
          Bottom Left glEnd(
          );
      glTranslatef(2.2f,0.0
      f,0.0f);// Move Right
          2.2 Units
glBegin
```

```
(GL_QUADS) ; /
/ Draw A Quad
      glVertex3f(-1.0f, 1.
      0f, 0.0f);// Top Left
      glVertex3f( 1.0f, 1.
      0f, 0.0f);// Top Ri
      ght glVertex3f( 1.0f
      ,-1.0f, 0.0f);//
      Bottom
         Right
      glVertex3f(-1.0f,-1
         .0f, 0.0f);//
         Bottom Left glEnd(
         );
      glTranslatef(2.0f,0.0
      f,0.0f);// Move Right
      2 Units
glBegin(GL_TRIANGLES);
      // Drawing Using
      Triangles glVertex3
      f( 0.0f, -1.0f, 0.0
      f);// Top
      glVertex3f(-1.0f,1.0
      f, 0.0f);// Bottom Le
      ft
      glVertex3f( 1.0f,1.0f
, 0.0f);// Bottom Right glE
nd();// Finished Drawing
The Triangle
```

Berdasarkan script program di atas maka dihasilkan output sebagai berikut :



Gambar 7. Dua buah polygon segitiga dan dua buah polygon segiempat

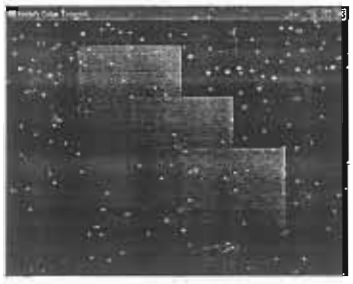
4.3 Analisis dan Simulasi Program Pemberian Warna Pada Polygon

Untuk memeberi warna sebuah polygon maka digunakan statemen : glColor3f (xf,yf,zf); x untuk warna merah (red), y untuk warna hijau (green), dan z untuk warna biru (blue). Contoh : glColor3f (1.0,0.0,0,0). Statemen ini adalah untuk memberi warna merah pada sebuah objek. Berikut adalah Implementasi Program OpenGL32 Pada Kombinasi Warna Dan Perputaran Polygon Obyek Gambar Segitiga dan Segiempat :

```
Sc
ri
pt
:
int InitGL(GLvoid)// All
Setup For OpenGL Goes Here
{
glShadeModel
(GL_SMOOTH);// Enable
Smooth Shading glClearCo
lor(0.0f, 0.0f, .0f, 0.9
f);// Black Background
glClearDepth(1.0f)
;// Depth Buffe Setu
p
g l E n a b l e
(GL_DEPTH_TEST) ; /
/ Enables Depth Tes
ting glDepthFunc
(GL_LEQUAL) ; / /
The Type Of Depth
Testing To Do
glHint
    (GL_PERSPECTIVE_CORRE
    CTION_HINT,
    GL_NICEST);// Really N
    ice Perspective Calcula
    tions
return TRUE;// I
nitialization
Went OK
}
int DrawGLScene( GLvoid )//
Here's Where We Do All The
Drawing
{
glClear
(GL_COLOR_BUFFER_BIT
| GL_DEPTH_BUFFER_BIT)
;// Clear Screen And
Depth Buffer
glLoadIdentity();// Re
set The Current Modelvi
ew Matrix
glTranslatef(-1.9f,0.0
f,-6.0f);// Move Left
1.5 Units And Into The S
creen 6.0
glTranslatef(1.0f,1.0f,0.
0f);// Move Left 1.5 Unit
s And Into The Screen 6.0
glBegin(GL_QUADS);//
Drawing Using Triang
les
      glColor3f(1.0f,0.0f,0.
      0f);// Set The Color To
      Red
      glVertex3f( -1.0f, 1.0
      f, 0.0f);// Top
```

```
        glColor3f(0.0f,1.0f,0.
        0f);// Set The Color To
        Green
        glVertex3f(1.0f,1.0
        f, 0.0f);// Bottom
        Left
        glColor3f(0.0f,0.0f
        ,1.0f);// Set The Co
        lor To Blue
        glVertex3f( 1.0f,-1.0
        f, 0.0f);// Bottom Righ
        t
        glColor3f(0.0f,0.0f,1.
        0f);// Set The Color
        To Blue
        glVertex3f( -1.0f,-1.
        0f, 0.0f);// Bottom Ri
        ght
glEnd();// Finishe
d Drawing The Trian
gle
glTranslatef(1.0f,-1.0f,0
.0f);// Move Left 1.5 Unit
s And Into The Screen 6.0
glBegin(GL_QUADS);//
Drawing Using Triang
les
        glColor3f(1.0f,0.0f,0.
        0f);// Set The Color
        To Red
        glVertex3f( -1.0f, 1.0
        f, 0.0f);// Top
        glColor3f(0.0f,1.0f,0.
        0f);// Set The Color
        To Green
        glVertex3f(1.0f,1.0f,
        0.0f);// Bottom Left
        glColor3f(0.0f,0.0f,1.
        0f);// Set The Color
        To Blue
         glVertex3f( 1.0f,-1.
         0f, 0.0f);// Bottom Ri
         ght
         glColor3f(0.0f,0.0f,1
         .0f);// Set The Color
         To Blue
         glVertex3f( -1.0f,-1
         .0f, 0.0f);// Bottom R
         ight
 glEnd();//Finished Drawin
 g The Triangle
 glTranslatef(1.0f,-1.0f,0.
 0f);// Move Left 1.5 Units
 And Into The Screen 6.0
 glBegin(GL_QUADS);// Draw
 ing Using Triangles
        glColor3f(1.0f,0.0f,0
        .0f);// Set The Color
        To Red
        glVertex3f( -1.0f, 1.
        0f, 0.0f);// Top
        glColor3f(0.0f,1.0f,0
        .0f);// Set The Color
        To Green
        glVertex3f(1.0f,1.0
        f, 0.0f);// Bottom
        Left
        glColor3f(0.0f,0.0f
        ,1.0f);// Set The Co
        lor To Blue
        glVertex3f( 1.0f,-1.
        0f, 0.0f);// Bottom Ri
        ght
        glColor3f(0.0f,0.0f,1
        .0f);
        // Set The Color To Bl
        ue
 glVertex3f( -1.0f,-1.
 0f, 0.0f);
 // Bottom Right
  glEnd();// Finished D
 rawing The Triangle
  // Done Drawing The
 Quad
 //rtri+=0.2f; // rquad
 --=0.15f;
  return TRUE;
  // Keep Going
 }
```

Berdasarkan script program di atas maka dihasilkan output sebagai berikut :



Gambar 8. Tiga buah polygon segiempat

Berikut contoh implementasi buah polygon segitiga dan tiga buah polygon segiempat :

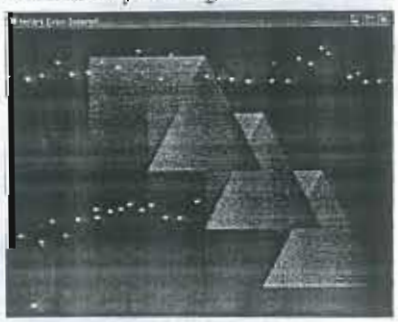
Script :

```
int InitGL(GLvoid)// All
Setup For OpenGL Goes Here
{
glShadeModel
(GL_SMOOTH);// Enable
Smooth ShadingglClearCo
lor(0.0f, 0.0f, .0f, 0.9
f);// Black Background
```

```
        glClearDepth(1.0f)
        ;// Depth Buffer Se
        tupglEnable
        (GL_DEPTH_TEST);/
        / Enables Depth Tes
        ting
        glDepthFunc
        (GL_LEQUAL);// The
        Type Of Depth Testing
        To Do
        glHint
           (GL_PERSPECTIVE_
           CORRECTION_HINT
           , GL_NICEST);// R
           eally Nice Perspe
           ctiveCalculation
           s
return
TRUE;// Initialization
Went OK }
int DrawGLScene(GLvoid)//
Here's Where We Do All The
Drawing
{
glClear
(GL_COLOR_BUFFER_BIT
| GL_DEPTH_BUFFER_BIT)
;// Clear Screen And
Depth Buffer
glLoadIdentity();// Reset
The Current ModelviewMatr
ix
glTranslatef(-1.9f,0.0f,-
6.0f);// Move Left 1.5Unit
s And Into The Screen 6.0
glTranslatef(1.0f,1.0f,0.0
f);// Move Left 1.5 Units
And Into The Screen 6.0
glBegin(GL_QUADS);
// Drawing Using Triangles
      glColor3f(1.0f,0.0f,0
      .0f);// Set The Color
      To Red
      glVertex3f( -1.0f, 1.
      0f, 0.0f);// Top
      glColor3f(0.0f,1.0f,0
      .0f);// Set The Color
      To Green
    glVertex3f(1.0f,1.0
    f, 0.0f);// Bottom
    Left glColor3f(0.0f,
    0.0f,1.0f);// Set
    The Color To Blue
glVertex3f( 1.0f,-1.0f, 0.0
f);
// Bottom Right
glColor3f(0.0f,0.0f,1.0f
);// Set The Color To Blu
eglVertex3f( -1.0f,-1.0f
, 0.0f);// Bottom Right gl
End();// Finished Drawin
g The Triangle
glTranslatef(1.0f,-1.0f,0.0
f);// Move Left 1.5 Units
And Into The Screen 6.0
glBegin(GL_QUADS);// Drawi
ng Using Triangles
      glColor3f(1.0f,0.0f,0.
      0f);
      // Set The Color To Red
      glVertex3f( -1.0f, 1.0
      f, 0.0f);// Top
      glColor3f(0.0f,1.0f,0.
      0f);// Set The Color To
      Green
      glVertex3f(1.0f,1.0
      f, 0.0f);// Bottom
      Left glColor3f(0.0f,
      0.0f,1.0f);// Set
      The Color To Blue
      glVertex3f( 1.0f,-1.0
      f, 0.0f);// Bottom Right
      t
      glColor3f(0.0f,0.0f,1.
      0f);// Set The Color To
      Blue
glVertex3f( -1.0f,-1.
0f, 0.0f);// Bottom Ri
ght glEnd();// Finished
Drawing The Triangle
glTranslatef(1.0f,-1.0f,0.0
f);// Move Left 1.5 Units
And Into The Screen 6.0
glBegin(GL_QUADS);// Drawi
ng Using Triangles
      glColor3f(1.0f,0.0f,0.
      0f);// Set The Color To
      Red
      glVertex3f( -1.0f, 1.0
      f, 0.0f);// Top
      glColor3f(0.0f,1.0f,0.
      0f);// Set The Color To
      Green
      glVertex3f(1.0f,1.0
      f, 0.0f);// Bottom
      Left glColor3f(0.0f,
      0.0f,1.0f);// Set
      The Color To Blue
      glVertex3f( 1.0f,-1.0
      f, 0.0f);// Bottom Right
      t
      glColor3f(0.0f,0.0f,1.
      0f);// Set The Color To
      Blue
glVertex3f( -1.0f,-1.
0f, 0.0f);// Bottom Ri
ght glEnd();// Finished
Drawing The Triangle
glTranslatef(1.0f,0.0f,0.0f
);// Move Left 1.5 Units
```

```
And Into The Screen 6.0
glBegin(GL_TRIANGLES);//
Drawing Using Triangles
      glColor3f(1.0f,0.0f,0.
      0f);// Set The Color To
      Red
      glVertex3f( 0.0f, 1.0f
      , 0.0f);// Top
      glColor3f(0.0f,1.0f,0.
      0f);// Set The Color To
      Green
      glVertex3f(-1.0f,-1
      .0f, 0.0f);// Bottom
      Left glColor3f(0.0f,0
      .0f,1.0f);// Set The
      Color To Blue
glVertex3f( 1.0f,-1.0
f, 0.0f);// Bottom Rig
ht glEnd();
glTranslatef(-1.0f,1.0f,0.
0f);
// Move Left 1.5 Units And
Into The Screen 6.0
glBegin(GL_TRIANGLES);//
Drawing Using Triangles
      glColor3f(1.0f,0.0f,0.
      0f);// Set The Color To
      Red
      glVertex3f( 0.0f, 1.0f
      , 0.0f);// Top
      glColor3f(0.0f,1.0f,0.
      0f);// Set The Color To
      Green
      glVertex3f(-1.0f,-1
      .0f, 0.0f);// Bottom
      Left glColor3f(0.0f,0
      .0f,1.0f);// Set The
      Color To Blue
glVertex3f( 1.0f,-1.0
f, 0.0f);// Bottom Rig
ht glEnd();
glTranslatef(-1.0f,1.0f,0.
0f);
// Move Left 1.5 Units And
Into The Screen 6.0
glBegin(GL_TRIANGLES);
// Drawing Using Triangles
      glColor3f(1.0f,0.0f,0.
      0f);
      // Set The Color To Red
      glVertex3f( 0.0f, 1.0f
      , 0.0f);
      // Top
      glColor3f(0.0f,1.0f,0.
      0f);// Set The Color To
      Green
      glVertex3f(-1.0f,-1
      .0f, 0.0f);// Bottom
      Left glColor3f(0.0f,0
      .0f,1.0f);
      // Set The Color To B
      lue
glVertex3f( 1.0f,-1.0
   f, 0.0f);// Bottom
   Right glEnd();
      // Done Drawing The
      Quad
      //rtri+=0.2f;
      //rquad-=0.15f;
      return TRUE;
      // Keep Going
}
glEnd();
// Finished Drawing The Tr
iangle
glTranslatef(-1.0f,0.0f,0.0
f);
// Move Left 1.5 Units And
Into The Screen 6.0
glBegin(GL_TRIANGLES);
// Drawing Using Triangles
      glColor3f(1.0f,0.0f,0.
      0f);
      // Set The Color To Red
      glVertex3f( 0.0f, -1.0
      f, 0.0f);// Top
      glColor3f(0.0f,1.0f,0.
      0f);
      //
      Set The Color To Green
      glVertex3f(-1.0f,1.0
      f, 0.0f);// Bottom L
      eft glColor3f(0.0f,0.
      0f,1.0f);
      // Set The Color To
      Blue
glVertex3f( 1.0f,1.0f
   , 0.0f);
// Bottom Right glEnd
   ();
      // Done Drawing The
      Quad
      //rtri+=0.2f; //
         rquad-=0.15f;
      return TRUE;// Keep Go
      ing
}
```

Berdasarkan script program di atas maka dihasilkan output sebagai berikut :



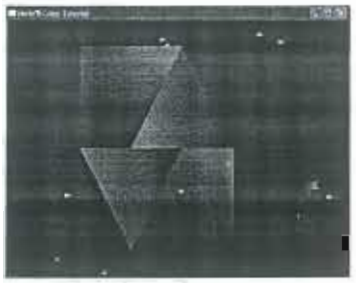
Gambar 9: Tiga buah polygon segitiga dan tiga buah polygon segiempat

Berikut ini dua buah polygon segitiga dan Dua buah polygon segiempat :

```
Script :
int InitGL(GLvoid)// All
Setup For OpenGL Goes Here
{
glShadeModel
(GL_SMOOTH);// Enable
Smooth ShadingglClearCo
lor(0.0f, 0.0f, .0f, 0.9
f);// BlackBackground
glClearDepth(1.0f);//
Depth Buffer Setup
glEnable(GL_DEPTH_TEST);/
/ Enables Depth Testinggl
DepthFunc(GL_LEQUAL);//
The Type Of Depth Testing
To Do
glHint
    (GL_PERSPECTIVE_CORRE
    CTION_HINT,
    GL_NICEST);// Really N
    ice Perspective Calcula
    tions
return TRUE;// Initializa
tion Went OK
}
int DrawGLScene(GLvoid)//
Here's Where We Do All The
Drawing
{
glClear
(GL_COLOR_BUFFER_BIT |
GL_DEPTH_BUFFER_BIT)
   ;// Clear Screen And
   Depth Buffer
glLoadIdentity();// Reset
The Current ModelviewMatr
ix
glTranslatef(-1.9f,0.0f,-
6.0f);// Move Left 1.5Unit
s And Into The Screen 6.0
glTranslatef(1.0f,1.0f,0.0
f);// Move Left 1.5 Units
And Into The Screen 6.0
glBegin(GL_QUADS);
// Drawing Using Triangles
      glColor3f(1.0f,0.0f,0
      .0f);// Set The Color
      To Red
      glVertex3f( -1.0f, 1.
      0f, 0.0f);// Top
      glColor3f(0.0f,1.0f,0
      .0f);// Set The Color
      To Green
      glVertex3f(1.0f,1.0
      f, 0.0f);// Bottom
      Left glColor3f(0.0f,
      0.0f,1.0f);// Set
      The ColorTo Blue
      glVertex3f( 1.0f,-1.
      0f, 0.0f);// Bottom'Ri
      ght
      glColor3f(0.0f,0.0f,1
      .0f);
      // Set The ColorTo Bl
      ue
glVertex3f( -1.0f,-1.
0f, 0.0f);// Bottom Ri
ght glEnd();// Finished
Drawing The Triangle
glTranslatef(1.0f,0.0f,0.0
f);// Move Left 1.5 Units
And Into The Screen 6.0
glBegin(GL_TRIANGLES);//
Drawing Using Triangles
      glColor3f(1.0f,0.0f,0
      .0f);// Set The Color
      To Red
      glVertex3f( 0.0f, 1.0
      f, 0.0f);// Top
      glColor3f(0.0f,1.0f,0
      .0f);// Set The Color
      To Green
      glVertex3f(-1.0f,-1
      .0f, 0.0f);// Bottom
      Left glColor3f(0.0f,0
      .0f,1.0f);// Set The
      ColorTo Blue
glVertex3f( 1.0f,-1.0
   f, 0.0f);// Bottom
   Right glEnd();
glTranslatef(0.0f,-2.0f,0.
0f);// Move Left 1.5 Units
And Into The Screen 6.0
glBegin(GL_QUADS);// Draw
ing Using Triangles
      glColor3f(1.0f,0.0f,0
      .0f);// Set The Color
      To Red
      glVertex3f( -1.0f, 1.
      0f, 0.0f);// Top
      glColor3f(0.0f,1.0f,0
      .0f);// Set The Color
      To Green
      glVertex3f(1.0f,1.0
      f, 0.0f);// Bottom
      Left glColor3f(0.0f,
      0.0f,1.0f);// Set
      The ColorTo Blue
      glVertex3f( 1.0f,-1.
      0f, 0.0f);// Bottom Ri
      ght
      glColor3f(0.0f,0.0f,1
```

```
      .0f);// Set The Color
      To Blue
glVertex3f( -1.0f,-1.0f,
0.0f);// Bottom Right
```

Berdasarkan program di atas dihasilkan output sebagai berikut :



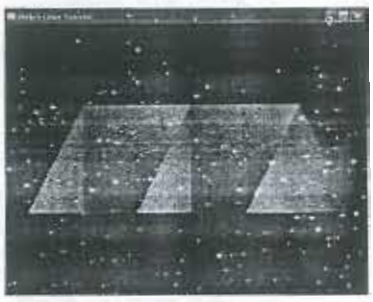
Gambar 10: Hasil pergeseran polygon Dua buah polygon segitiga dan Dua buah polygon segiempat

Berikut ini adalah script program untuk menggabungkan Tiga buah polygon segitiga dan Dua buah polygon segiempat :

```
Sc
ri
pt
:
int InitGL(GLvoid)// All
Setup For OpenGL Goes Here
{
glShadeModel
(GL_SMOOTH);// Enable
Smooth ShadingglClearCo
lor(0.0f, 0.0f, .0f, 0.9
f);// BlackBackground
glClearDepth(1.0f)
;// Depth Buffer Se
tup
glEnable(GL_DEPTH_TEST);/
/ Enables Depth TestinggI
DepthFunc(GL_LEQUAL);//
The Type Of Depth Testing
To Do
glHint
    (GL_PERSPECTIVE_CORRE
    CTION_HINT,
    GL_NICEST);// Really N
    ice Perspective Calcula
    tions
return TRUE;// I
nitialization
Went OK
}
int DrawGLScene(GLvoid)//
Here's Where We Do All The
Drawing
{
glClear
(GL_COLOR_BUFFER_BIT |
GL_DEPTH_BUFFER_BIT)
   ;// Clear Screen And
   Depth Buffer
glLoadIdentity();// Re
set The Current Modelvi
ewMatrix
glTranslatef(-1.9f,0.0
f,-6.0f);// Move Left
1.5Units And Into The S
creen 6.0
//glTranslatef(1.0f,0.
0f,0.0f);// Move Left
1.5Units And Into The S
creen 6.0
glBegin(GL_TRIANGLES);
// Drawing Using Trian
gles
      glColor3f(1.0f,0.0f,0.
      0f);
      // Set The Color To Red
      glVertex3f( 0.0f
      , 1.0f, 0.0f);//
            Top
      glColor3f(0.0f,1.0f,0.
      0f);
      // Set The ColorTo Gr
      een
      glVertex3f(-1.0f,-1
      .0f, 0.0f);// Bottom
      Left glColor3f(0.0f,0
      .0f,1.0f);// Set The
      ColorTo Blue
glVertex3f( 1.0f,-1.0
   f, 0.0f);// Bottom
   Right glEnd();
glTranslatef(1.0f,0.0f,0.
0f);// Move Left 1.5 Unit
sAnd Into The Screen 6.0
glBegin(GL_QUADS);//
Drawing Using Triang
les
      glColor3f(1.0f,0.0f,0
      .0f);// Set The Color
      To Red
      glVertex3f( -1.0f, 1.
      0f, 0.0f);// Top
      glColor3f(0.0f,1.0f,0
      .0f);// Set The Color
      To Green
      glVertex3f(1.0f,1.0
      f, 0.0f);// Bottom
      Left glColor3f(0.0f,
      0.0f,1.0f);// Set
      The ColorTo Blue
      glVertex3f( 1.0f,-1.0
      f, 0.0f);// BottomRig
      ht
      glColor3f(0.0f,0.0f,1
```

```
        .0f);// Set The Color
        To Blue
glVertex3f( -1.0f,-1.
0f, 0.0f);// Bottom Ri
ght glEnd();// Finished
Drawing The Triangle
glTranslatef(1.0f,0.0f,0.
0f);// Move Left 1.5 Unit
sAnd Into The Screen 6.0
glBegin(GL_TRIANGLES);
// Drawing Using Trian
gles
        glColor3f(1.0f,0.0f,0
        .0f);// Set The Color
        To Red
        glVertex3f( 0.0f, 1.0
        f, 0.0f);// Top
        glColor3f(0.0f,1.0f,0
        .0f);// Set The Color
        To Green
        glVertex3f(-1.0f,-1
        .0f, 0.0f);// Bottom
        Left glColor3f(0.0f,0
        .0f,1.0f);// Set The
        Color To Blue
glVertex3f( 1.0f,-1.0
    f, 0.0f);// Bottom
    Right glEnd();
glTranslatef(1.0f,0.0f,0.
0f);// Move Left 1.5 Unit
sAnd Into The Screen 6.0
glBegin(GL_QUADS);//
Drawing Using Triang
les
        glColor3f(1.0f,0.0f,0
        .0f);// Set The Color
        To Red
        glVertex3f( -1.0f, 1.
        0f, 0.0f);// Top
        glColor3f(0.0f,1.0f,0
        .0f);// Set The Color
        To Green
        glVertex3f(1.0f,1.0
        f, 0.0f);// Bottom
        Left glColor3f(0.0f,
        0.0f,1.0f);// Set
        The Color To Blue
        glVertex3f( 1.0f,-1.0
        f, 0.0f);// Bottom Rig
        ht
        glColor3f(0.0f,0.0f,1
        .0f);// Set The Color
        To Blue
glVertex3f( -1.0f,-1.
0f, 0.0f);// Bottom Ri
ght glEnd();// Finished
Drawing The Triangle
glTranslatef(1.0f,0.0f,0.
0f);// Move Left 1.5 Unit
sAnd Into The Screen 6.0
glBegin(GL_TRIANGLES);
// Drawing Using Trian
gles
        glColor3f(1.0f,0.0f,0
        .0f);// Set The Color
        To Red
        glVertex3f( 0.0f, 1.0
        f, 0.0f);// Top
        glColor3f(0.0f,1.0f,0
        .0f);// Set The Color
        To Green
        glVertex3f(-1.0f,-1
        .0f, 0.0f);// Bottom
        Left glColor3f(0.0f,0
        .0f,1.0f);// Set The
        Color To Blue
glVertex3f( 1.0f,-1.0
    f, 0.0f);// Bottom
    Right glEnd();
        // Done Drawing The
        Quad //rtri+=0.2f; //
            rquad-=0.15f;
        return TRUE;// Keep G
        oing
}
```

Berdasarkan script program di atas maka dihasilkan output sebagai berikut :



Gambar 11: Penggabungan polygon Tiga buah polygon segitiga dan Dua buah polygon segiempat

5 Kesimpulan

Berdasarkan hasil analisis dan dan pengujian program yang dilakukan, maka dapat disimpulkan beberapa hal yaitu bahwa pembuatan pemrograman grafis dapat digunakan OpenGL yang mempunyai suatu library grafis standard. Library grafis selain OpenGL yang bisa digunakan adalah DirectX. Dengan Pemrograman OpenGL sangat membantu dalam membuat dan merancang suatu model objek baik untuk 2D dan 3D, serta melakukan modifikasi obyek yang telah dibuat dengan cara mengubah parameter dari fungsi-fungsi dari OpenGL yang ada. Baik modifikasi warna dengan statemen glColor3f

ataupun modifikasi letak dari objek dengan statemen glVertex3f.

Daftar Pustaka

[1] Jeff Molofee, http://nehe.gamedev.net, Jakarta, 2012.

[2] Shreiner, Dave, OpenGL Programming Guide; The Khronos OpenGL ARB Working Group, Seventh Edition : The Official Guide to Learning OpenGL, Version 3.0 and 3.1Addison Wesley, Boston, 2009.

[3]URL:http://download.microsoft.com/download/9/b/0/9b06f663-23d0-4709-a2ff-90df8dc558bb/MSDNlib-vs2008sp1Readme.htm/, MSDN Library for Visual Studio 2008, Jakarta, 2012.

[4]URL:http://www-users.itlabs.umn.edu/classes/Spring-2009/c-sci4107/GlutSetupWin.html, Jakarta, 2012.

[5]URL:http://user.xmission.com/~nate/glut.html, Jakarta, 2012.

[6] K.Akeley.OpenGL philosophy and the philosopher's drinking song. Personal Communication, 1996.