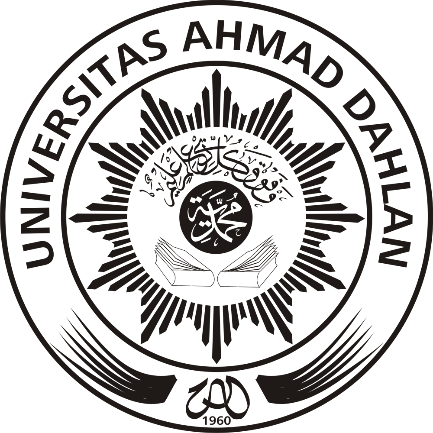
**LAPORAN AKHIR PRAKTIKUM**

**GRAFIKA KOMPUTER**



**DISUSUN OLEH**

SIRLY ZIADATUL MUSTAFIDAH (2200018345)

JUMAT 10.00 - H

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS TEKNOLOGI INDUSTRI**

**UNIVERSITAS AHMAD DAHLAN**

**JULI 2024**

**LEMBAR PENGESAHAN ASISTEN**

**RESPONSI**

**LAPORAN AKHIR PRAKTIKUM GRAFIKA KOMPUTER**

Disusun oleh:

**SIRLY ZIADATUL MUSTAFIDAH**

**2200018345**

**Laboratorium Informatika**

**Program Studi S1 Informatika**

**Fakultas Teknologi Industri**

**Universitas Ahmad Dahlan**

**Telah disetujui oleh:**

**PJ Asisten**

[ttd]

**Tata Alyssa**

**2100018315**

**DAFTAR ISI**

[KATA PENGANTAR iii](#_Toc171186286)

[DAFTAR GAMBAR iv](#_Toc171186287)

[BAB I. PENGANTAR OPEN GL 1](#_Toc171186288)

[1. Pretest 1 1](#_Toc171186289)

[2. Langkah Praktikum 1 3](#_Toc171186290)

[3. Posttest 1 11](#_Toc171186291)

[BAB II. ALGORITMA GARIS 20](#_Toc171186292)

[1. Pretest 2 20](#_Toc171186293)

[2. Langkah Praktikum 2 21](#_Toc171186294)

[3. Posttes 2 49](#_Toc171186295)

[BAB III. INTERPOLASI DAN KURVA 65](#_Toc171186296)

[1. Pretest 3 65](#_Toc171186297)

[2. Langkah Praktikum 3 66](#_Toc171186298)

[3. Posttest 3 78](#_Toc171186299)

[BAB IV. TRANSFORMASI 2D DAN 3D 89](#_Toc171186300)

[1. Pretest 4 89](#_Toc171186301)

[2. Langkah Praktikum 4 91](#_Toc171186302)

[3. Posttest 4 104](#_Toc171186303)

[BAB V. PROYEKSI 3D 117](#_Toc171186304)

[1. Pretest 5 117](#_Toc171186305)

[2. Langkah Praktikum 5 118](#_Toc171186306)

[3. Postest 5 126](#_Toc171186307)

[BAB VI. REPRESENTASI OBYEK 3D 134](#_Toc171186308)

[1. Pretest 6 134](#_Toc171186309)

[2. Langkah Praktikum 6 135](#_Toc171186310)

[3. Posttest 6 148](#_Toc171186311)

[BAB VII. KURVA SPLINE 161](#_Toc171186312)

[1. Pretest 7 161](#_Toc171186313)

[2. Langkah Praktikum 7 162](#_Toc171186314)

[3. Posttest 7 186](#_Toc171186315)

[BAB VIII. TEKNIK REPRESENTASI PERMUKAAN 210](#_Toc171186316)

[1. Pretest 8 210](#_Toc171186317)

[2. Langkah Praktikum 8 211](#_Toc171186318)

[3. Posttest 8 222](#_Toc171186319)

[BAB IX. TEKNIK PEMODELAN OBYEK 3D 233](#_Toc171186320)

[1. Pretest 9 233](#_Toc171186321)

[2. Langkah Praktikum 9 234](#_Toc171186322)

[3. Posttest 9 246](#_Toc171186323)

[BAB X. TEKNIK SUBDIVISI 258](#_Toc171186324)

[1. Pretest 10 258](#_Toc171186325)

[2. Langkah Praktikum 10 259](#_Toc171186326)

[3. Posttest 10 274](#_Toc171186327)

[DAFTAR PUSTAKA 288](#_Toc171186328)

# KATA PENGANTAR

*Assalamu’alaikum Warahmatullahi Wabarakatuh*

*Alhamdulillah* saya haturkan puja dan puji syukur kehadirat Allah SWT atas karunia dan segala rahmat, taufik, dan hidayah-Nya sehingga saya akhirnya dapat menyelesaikan laporan yang dibuat ini.

Laporan ini disusun untuk memenuhi tugas akhir Responsi Praktikum Grafika Komputer 2024. Laporan ini berisi tentang pembahasan pretest, program-program menggunakan openGL dari pertemuan 1 hingga pertemuan 10 untuk meguji pemahaman selama mengikuti praktikum 10 pertemuan yang lalu.

Sebagai penyusun, saya menyadari bahwa masih terdapat banyak kekurangan, baik dari penyusunan maupun tata bahasa penyampaian dalam laporan ini. Oleh sebab itu, saya berharap adanya kritik dan saran demi perbaikan yang akan datang. Saya mohon maaf apabila ada kesalahan kata yang kurang berkenan.

*Wassalamu’alaikum Warahmatullahi Wabarakatuh*

Surakarta, 6 Juli 2024

Sirly Ziadatul Mustafidah

# DAFTAR GAMBAR

[Gambar 1. 1 1](#_Toc171179137)

[Gambar 1. 2 10](#_Toc171179138)

[Gambar 1. 3 19](#_Toc171179139)

[Gambar 2. 1 33](#_Toc171179220)

[Gambar 2. 2 48](#_Toc171179221)

[Gambar 2. 3 64](#_Toc171179222)

[Gambar 3. 1 77](#_Toc171179254)

[Gambar 3. 2 88](#_Toc171179255)

[Gambar 4. 1 103](#_Toc171179265)

[Gambar 4. 2 116](#_Toc171179266)

[Gambar 5. 1 125](#_Toc171179273)

[Gambar 5. 2 133](#_Toc171179274)

[Gambar 6. 1 147](#_Toc171179280)

[Gambar 6. 2 160](#_Toc171179281)

[Gambar 7. 1 185](#_Toc171179287)

[Gambar 7. 2 209](#_Toc171179288)

[Gambar 8. 1 221](#_Toc171179298)

[Gambar 8. 2 221](#_Toc171179299)

[Gambar 8. 3 232](#_Toc171179300)

[Gambar 9. 1 245](#_Toc171179306)

[Gambar 9. 2 257](#_Toc171179307)

[Gambar 10. 1 272](#_Toc171179313)

[Gambar 10. 2 272](#_Toc171179314)

[Gambar 10. 3 273](#_Toc171179315)

[Gambar 10. 4 273](#_Toc171179316)

[Gambar 10. 5 286](#_Toc171179317)

[Gambar 10. 6 286](#_Toc171179318)

[Gambar 10. 7 287](#_Toc171179319)

[Gambar 10. 8 287](#_Toc171179320)

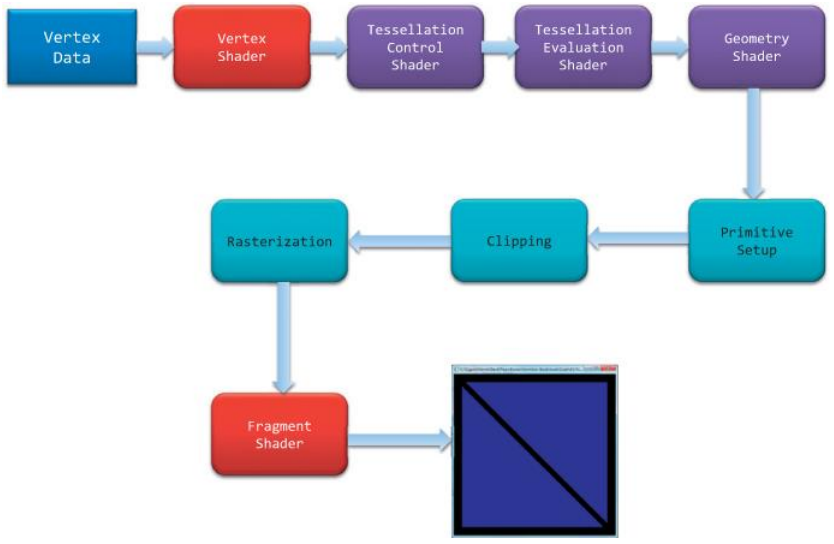
# BAB I. PENGANTAR OPEN GL

## Pretest 1

1. Apa saja yang bisa dilakukan dengan library OpenGL?

Open Graphics Library (OpenGL) digunakan untuk menggambar sebuah obyek dua dimensi (2D) dan tiga dimensi (3D)

1. Gambarkan kemudian jelaskan tahapan OpenGL pipeline!



Gambar 1. 1

Vertex data: titik awal informasi model 3D seperti posisi vertex, warna, dan koordinat tekstur disimpan dalam buffer di CPU

Vertex shader: program shader ini dieksekusi di GPU untuk setiap vertex

Tessellation control shader (optional): digunakan untuk pembuatan geometri tingkat lanjut. Ini menentukan seberapa banyak primitif harus dibagi

Tessellation evaluation shader (optional): mengambil informasi subdivisi dari control shader dan menghitung posisi akhir dari vertex baru yang dihasilkan selama tessellation

Geometry shader (optional): shader ini beroperasi pada primitif dan dapat memodifikasinya. Bisa digunakan untuk membuat primitif baru, memodifikasi primitif yang ada dan menghasilkan data per primitif untuk fragment shader

Primitive setup: mengambil primitif yang sudah diproses dari tahap sebelumnya dan mempersiapkannya untuk rasterisasi

Clipping: memoting primitif yang berada di luar area visible untuk meningkatkan efiseiensi rendering

Rasterizer: mengubah primitif geometris menjadi fragmen yang merupakan sampel kecil yang mewakli warna pixel pada layar

Fragment shader: program shader ini dieksekusi untuk setiap fragmen yang dihasilkan oleh rasterizer

Frame buffer: warna fragmen yang dihitung ditulis ke frame buffer yang mewakili gambar akhir yang ditampilkan di layar

## Langkah Praktikum 1

Source Code

// kode dasar untuk semua praktikum grafika komputer

// adhi prahara. 2018

// deklarasikan semua header disini

#include <windows.h>

#include <GL/glut.h>

// inisialisasi variabel untuk transformasi seperti translasi, rotasi atau scaling

float angle = 0.0f; // sudut transformasi kamera

float posX = 0.0f, rotX = 0.0f; // posisi kamera di sumbu X

float posY = 0.0f, rotY = 0.0f; // posisi kamera di sumbu Y

float posZ = 5.0f, rotZ = -1.0f; // posisi kamera di sumbu Z

float objectAngleX = 0.0f; // sudut tranformasi obyek di sumbu X

float objectAngleY = 0.0f; // sudut tranformasi obyek di sumbu Y

float objectAngleZ = 0.0f; // sudut tranformasi obyek di sumbu Z

// fungsi untuk menggambar obyek kubus

void drawObject()

{

// obyek bisa dimasukkan diantara glPushMatrix() dan glPopMatrix()

// fungsinya agar obyek tidak terpengaruh atau mempengaruhi obyek lain

// saat diwarnai, ditransformasi dan sebagainya

glPushMatrix();

// operasi transformasi rotasi obyek ke arah kanan-kiri

glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);

glPushMatrix();

// operasi transformasi rotasi obyek ke arah atas-bawah

glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

// set warna obyek ke warna hijau (0.0f, 1.0f, 0.0f)

glColor3f(1.0f, 0.0f, 1.0f);

// bila menggambar obyek harus diawali glBegin(tipe obyek) dan diakhiri dengan glEnd()

// kecuali menggunakan fungsi yang sudah ada di GLUT-OpenGL seperti dibawah ini

// glutSolidCube(1.0f); // menggambar obyek kubus

glutSolidDodecahedron(); //menggambar obyek Dodecahedron

glPopMatrix();

glPopMatrix();

}

// taruh semua obyek yang akan digambar di fungsi display()

void display()

{

// bersihkan dan reset layar dan buffer

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

// posisikan kamera pandang

// dalam hal ini sumbu Y ada diatas dan posisi kamera pandang di (posX, posY, posZ)

gluLookAt(posX, posY, posZ, posX + rotX, posY + rotY, posZ + rotZ, 0.0f, 1.0f, 0.0f);

// panggil fungsi untuk menggambar obyek

drawObject();

// tampilkan obyek ke layar

// gunakan glFlush() bila memakai single buffer

// gunakan glutSwapBuffers() bila memakai double buffer

glutSwapBuffers();

}

// inisialisasikan variabel, pencahayaan, tekstur dan pengaturan kamera pandang di fungsi init()

void init(void)

{

// inisialisasi warna latar belakang layar dalam hal ini warna putih (1.0, 1.0, 1.0, 0.0)

glClearColor(1.0, 1.0, 1.0, 0.0);

glEnable(GL\_DEPTH\_TEST); // mengaktifkan depth buffer

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45.0, 1.0, 1.0, 100.0); // set proyeksi ke perspektif

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// inisialisasi kamera pandang

gluLookAt(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

}

// fungsi ini digunakan bila layar akan diresize (default)

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45, (GLfloat)w / (GLfloat)h, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

}

// fungsi untuk mengatur masukan dari keyboard

// untuk arah kiri, kanan, atas, bawah, PgUp, dan PgDn

void keyboard(int key, int x, int y)

{

float fraction = 0.1f;

switch (key)

{

// masukkan perintah disini bila tombol kiri ditekan

case GLUT\_KEY\_LEFT:

// dalam hal ini perintah rotasi obyek ke kiri sebanyak 1 derajat

objectAngleY -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol kanan ditekan

case GLUT\_KEY\_RIGHT:

// dalam hal ini perintah rotasi obyek ke kanan sebanyak 1 derajat

objectAngleY += 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol atas ditekan

case GLUT\_KEY\_UP:

// dalam hal ini perintah rotasi obyek ke atas sebanyak 1 derajat

objectAngleX -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol bawah ditekan

case GLUT\_KEY\_DOWN:

// dalam hal ini perintah rotasi obyek ke bawah sebanyak 1 derajat

objectAngleX += 1.0f;

glutPostRedisplay(); // update obyek

break;

// zoom in

case GLUT\_KEY\_PAGE\_UP:

// masukkan perintah disini bila tombol PgUp ditekan

posX += rotX \* fraction;

posZ += rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

// zoom out

case GLUT\_KEY\_PAGE\_DOWN:

// masukkan perintah disini bila tombol PgDn ditekan

posX -= rotX \* fraction;

posZ -= rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

}

}

// timer untuk animasi (gunakan bila perlu)

void timer(int value)

{

glutPostRedisplay();

glutTimerFunc(55, timer, 0);

}

// program utama

int main(int argc, char\*\* argv)

{

// inisialisasi jendela OpenGL

// GLUT\_SINGLE berarti memakai single buffer

// GLUT\_DOUBLE berarti memakai double buffer

// GLUT\_RGB berarti mode tampilan yang dipakai RGB

// GLUT\_RGBA berarti mode tampilan yang dipakai RGBA

// GLUT\_DEPTH berarti memakai depth buffer

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA | GLUT\_DEPTH);

// set ukuran jendela tampilan

glutInitWindowSize(480, 480); // besarnya jendela dalam piksel dalam hal ini 300x300

glutInitWindowPosition(100, 100); // posisi jendela dilayar komputer dalam piksel

// judul jendela (wajib diubah dengan informasi NAMA / NIM - JUDUL PRAKTIKUM masing-masing)

glutCreateWindow("SIRLY ZIADATUL MUSTAFIDAH / 2200018345 - PENGANTAR OPENGL");

// panggil fungsi init untuk inisialisasi awal

init();

// event handler untuk display, reshape dan keyboard

glutDisplayFunc(display); // display

glutReshapeFunc(reshape); // reshape

glutSpecialFunc(keyboard); // keyboard

//glutTimerFunc(0, timer, 0); // aktifkan timer bila perlu

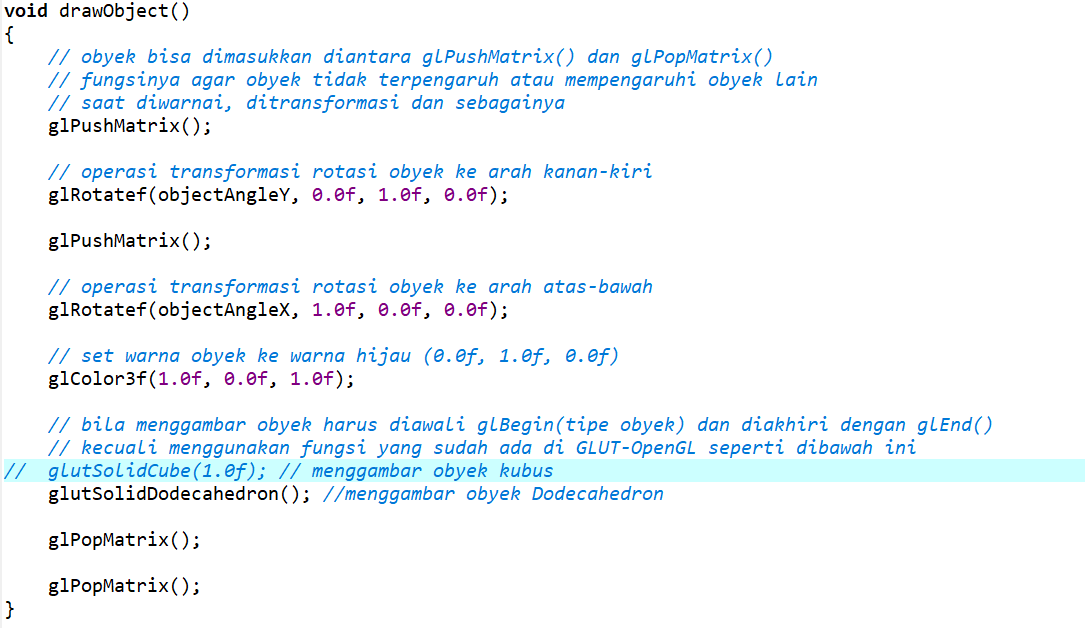
// looping

glutMainLoop();

return 0;

}

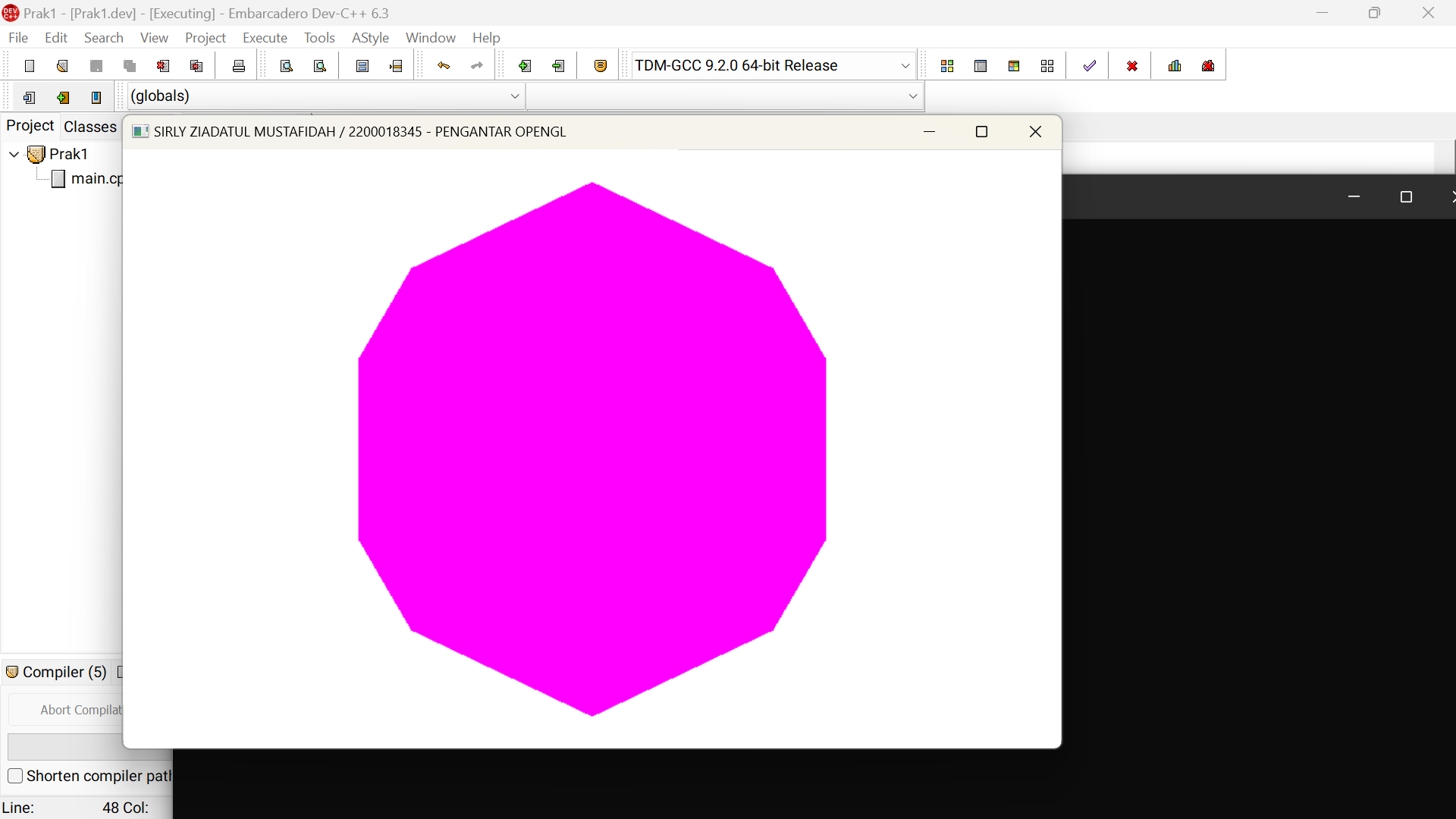
Penjelasan



Untuk mengubah warna obyek menggunakan fungsi glColor3f yang berupa warna RGB

Menambahkan fungsi glutSolidDodecahedron untuk membuat obyek Dodecahedron

Output



Gambar 1. 2

## Posttest 1

Source Code

// kode dasar untuk semua praktikum grafika komputer

// adhi prahara. 2018

// deklarasikan semua header disini

#include <windows.h>

#include <GL/glut.h>

// inisialisasi variabel untuk transformasi seperti translasi, rotasi atau scaling

float angle = 0.0f; // sudut transformasi kamera

float posX = 0.0f, rotX = 0.0f; // posisi kamera di sumbu X

float posY = 0.0f, rotY = 0.0f; // posisi kamera di sumbu Y

float posZ = 5.0f, rotZ = -1.0f; // posisi kamera di sumbu Z

float objectAngleX = 0.0f; // sudut tranformasi obyek di sumbu X

float objectAngleY = 0.0f; // sudut tranformasi obyek di sumbu Y

float objectAngleZ = 0.0f; // sudut tranformasi obyek di sumbu Z

// fungsi untuk menggambar obyek kubus

void drawObject()

{

// obyek bisa dimasukkan diantara glPushMatrix() dan glPopMatrix()

// fungsinya agar obyek tidak terpengaruh atau mempengaruhi obyek lain

// saat diwarnai, ditransformasi dan sebagainya

glPushMatrix();

// operasi transformasi rotasi obyek ke arah kanan-kiri

glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);

glPushMatrix();

// operasi transformasi rotasi obyek ke arah atas-bawah

glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

// set warna obyek ke warna hijau (0.0f, 1.0f, 0.0f)

glColor3f(0.5f, 0.5f, 1.0f);

// bila menggambar obyek harus diawali glBegin(tipe obyek) dan diakhiri dengan glEnd()

// kecuali menggunakan fungsi yang sudah ada di GLUT-OpenGL seperti dibawah ini

// glutSolidCube(1.0f); // menggambar obyek kubus

glutSolidOctahedron(); //menggambar obyek Octahedron

glPopMatrix();

glPopMatrix();

}

// taruh semua obyek yang akan digambar di fungsi display()

void display()

{

// bersihkan dan reset layar dan buffer

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

// posisikan kamera pandang

// dalam hal ini sumbu Y ada diatas dan posisi kamera pandang di (posX, posY, posZ)

gluLookAt(posX, posY, posZ, posX + rotX, posY + rotY, posZ + rotZ, 0.0f, 1.0f, 0.0f);

// panggil fungsi untuk menggambar obyek

drawObject();

// tampilkan obyek ke layar

// gunakan glFlush() bila memakai single buffer

// gunakan glutSwapBuffers() bila memakai double buffer

glutSwapBuffers();

}

// inisialisasikan variabel, pencahayaan, tekstur dan pengaturan kamera pandang di fungsi init()

void init(void)

{

// inisialisasi warna latar belakang layar dalam hal ini warna putih (1.0, 1.0, 1.0, 0.0)

glClearColor(1.0, 1.0, 1.0, 0.0);

glEnable(GL\_DEPTH\_TEST); // mengaktifkan depth buffer

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45.0, 1.0, 1.0, 100.0); // set proyeksi ke perspektif

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// inisialisasi kamera pandang

gluLookAt(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

}

// fungsi ini digunakan bila layar akan diresize (default)

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45, (GLfloat)w / (GLfloat)h, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

}

// fungsi untuk mengatur masukan dari keyboard

// untuk arah kiri, kanan, atas, bawah, PgUp, dan PgDn

void keyboard(int key, int x, int y)

{

float fraction = 0.1f;

switch (key)

{

// masukkan perintah disini bila tombol kiri ditekan

case GLUT\_KEY\_LEFT:

// dalam hal ini perintah rotasi obyek ke kiri sebanyak 1 derajat

objectAngleY -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol kanan ditekan

case GLUT\_KEY\_RIGHT:

// dalam hal ini perintah rotasi obyek ke kanan sebanyak 1 derajat

objectAngleY += 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol atas ditekan

case GLUT\_KEY\_UP:

// dalam hal ini perintah rotasi obyek ke atas sebanyak 1 derajat

objectAngleX -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol bawah ditekan

case GLUT\_KEY\_DOWN:

// dalam hal ini perintah rotasi obyek ke bawah sebanyak 1 derajat

objectAngleX += 1.0f;

glutPostRedisplay(); // update obyek

break;

// zoom in

case GLUT\_KEY\_PAGE\_UP:

// masukkan perintah disini bila tombol PgUp ditekan

posX += rotX \* fraction;

posZ += rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

// zoom out

case GLUT\_KEY\_PAGE\_DOWN:

// masukkan perintah disini bila tombol PgDn ditekan

posX -= rotX \* fraction;

posZ -= rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

}

}

// timer untuk animasi (gunakan bila perlu)

void timer(int value)

{

glutPostRedisplay();

glutTimerFunc(55, timer, 0);

}

// program utama

int main(int argc, char\*\* argv)

{

// inisialisasi jendela OpenGL

// GLUT\_SINGLE berarti memakai single buffer

// GLUT\_DOUBLE berarti memakai double buffer

// GLUT\_RGB berarti mode tampilan yang dipakai RGB

// GLUT\_RGBA berarti mode tampilan yang dipakai RGBA

// GLUT\_DEPTH berarti memakai depth buffer

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA | GLUT\_DEPTH);

// set ukuran jendela tampilan

glutInitWindowSize(480, 480); // besarnya jendela dalam piksel dalam hal ini 300x300

glutInitWindowPosition(100, 100); // posisi jendela dilayar komputer dalam piksel

// judul jendela (wajib diubah dengan informasi NAMA / NIM - JUDUL PRAKTIKUM masing-masing)

glutCreateWindow("SIRLY ZIADATUL MUSTAFIDAH / 2200018345 - PENGANTAR OPENGL");

// panggil fungsi init untuk inisialisasi awal

init();

// event handler untuk display, reshape dan keyboard

glutDisplayFunc(display); // display

glutReshapeFunc(reshape); // reshape

glutSpecialFunc(keyboard); // keyboard

//glutTimerFunc(0, timer, 0); // aktifkan timer bila perlu

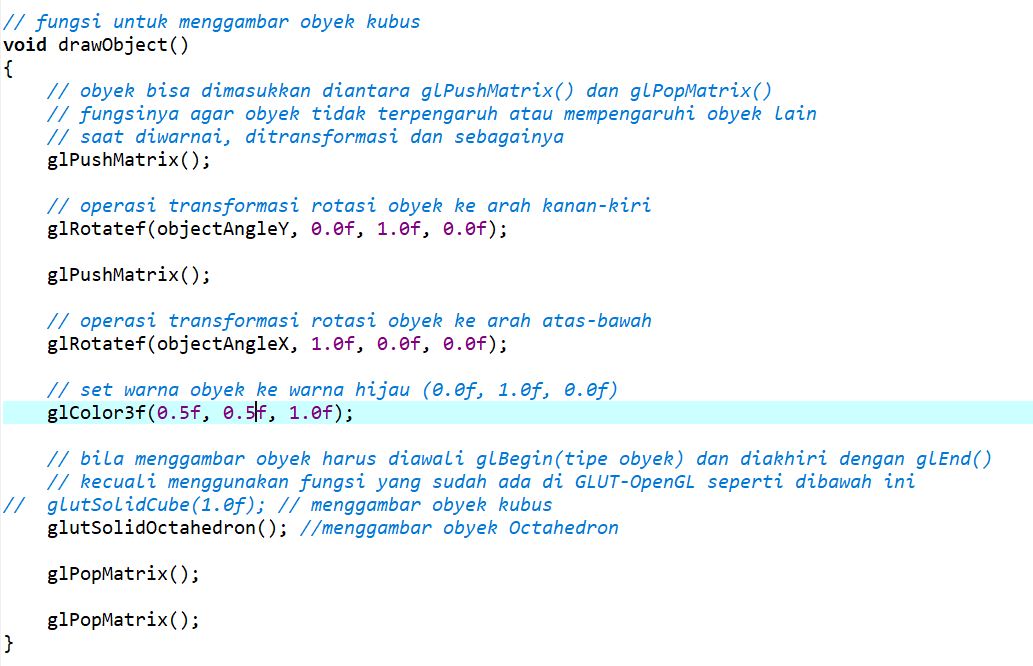
// looping

glutMainLoop();

return 0;

}

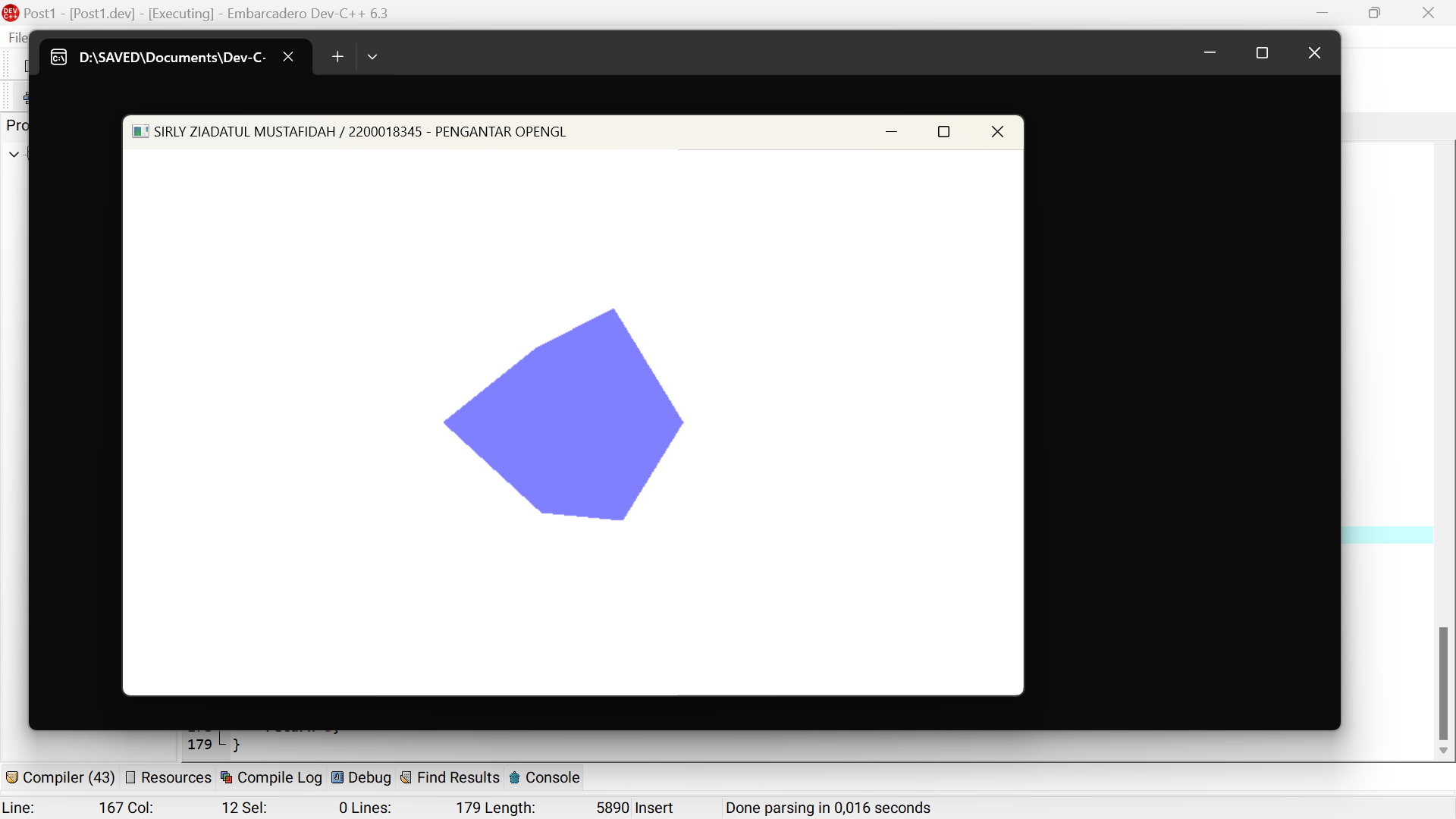
Penjelasan



Untuk mengubah warna obyek menggunakan fungsi glColor3f yang berupa warna RGB

Menambahkan fungsi glutSolidDodecahedron untuk membuat obyek Dodecahedron

Output



Gambar 1. 3

# BAB II. ALGORITMA GARIS

1. Pretest 2
2. Jelaskan tahapan pembangkitan garis dengan algoritma DDA!

Ketika slope berada pada nilai -1 <= m <= 1 maka koordinat x naik satu demi satu dan koordinat y naik berdasarkan slope dari garis

Dengan menaikkan koordiant x dengan 1, maka y dapat dihitung: yk+1 = yk+m

Bila m di luar nilai tersebut maka dilakukan sebaliknya

Dengan menaikkan koordinat y dengan 1, maka x dapat dihitung: xk+1 = xk+1/m

Selanjutnya nilai hasil perhitungan harus dibulatkan agar cocok dengan nilai piksel

1. Jelaskan tahapan pembangkitan garis dengan algoritma Bresenham!

Bila titik awal garis (x1, y1) dan akhir garis (x2, y2), nilai inisialisasi awal m hitung:

* Selisih lebar : ∆x = x2-x1
* Selisih tinggi : ∆y = y2-y1
* 2∆y = 2(y2-y1)

Inisial parameter keputusan: p0 = 2∆y-∆x

Setiap xk di sepanjang garis, mulai dari K = 0, cek kondisi berikut:

* Jika pk < 0, maka titik selanjutnya untuk digambar di: (xk+1, yk) pk+1 = pk+2∆y
* Selain itu maka titik selanjutnya untuk digambar di: (xk+1, yk+1) pk+1 = pk+2∆y-2∆x

Ulangi langkah diatas sebanyak ∆x

1. Langkah Praktikum 2

Algoritma DDA

Source Code

// kode dasar untuk semua praktikum grafika komputer

// adhi prahara. 2018

// deklarasikan semua header disini

#include <windows.h>

#include <GL/glut.h>

#include <stdio.h>

#include <iostream>

int SCREEN\_WIDTH = 400;

int SCREEN\_HEIGHT = 400;

// inisialisasi variabel untuk transformasi seperti translasi, rotasi atau scaling

float angle = 0.0f; // sudut transformasi kamera

float posX = 0.0f, rotX = 0.0f; // posisi kamera di sumbu X

float posY = 0.0f, rotY = 0.0f; // posisi kamera di sumbu Y

float posZ = 5.0f, rotZ = -1.0f; // posisi kamera di sumbu Z

float objectAngleX = 0.0f; // sudut tranformasi obyek di sumbu X

float objectAngleY = 0.0f; // sudut tranformasi obyek di sumbu Y

float objectAngleZ = 0.0f; // sudut tranformasi obyek di sumbu Z

// container untuk membuat tipe data 3D (X, Y, Z)

struct Vec3{

float X; float Y; float Z;

Vec3(float x, float y, float z) { X = x; Y = y; Z = z; }

//

Vec3() { }

~Vec3() { }

};

// fungsi untuk menggambar garis dengan algoritma DDA bila terhadap X

void lineDDAX(Vec3 point1, Vec3 point2)

{

// hitung gradient garis m

int dY = point2.Y - point1.Y;

int dX = point2.X - point1.X;

float m = (float)dY / dX;

float im = 1.0f/m;

// mulai menggambar titik-titik

glBegin(GL\_POINTS);

// koordinat titik awal

glVertex3f(point1.X, point1.Y, point1.Z);

float pX = point1.X, pY = point1.Y, pZ = point1.Z;

// kenaikan terhadap X

for (int i = point1.X; i < point2.X; i++)

{

pX = pX + 1; // Xn+1 = Xn + 1

pY = pY + m; // Yn+1 = Yn + m

glVertex3f(pX, pY, pZ);

}

// koordinat titik akhir

glVertex3f(point2.X, point2.Y, point2.Z);

glEnd();

}

// fungsi untuk menggambar garis dengan algoritma DDA bila terhadap Y

void lineDDAY(Vec3 point1, Vec3 point2)

{

// hitung gradient garis m

int dY = point2.Y - point1.Y;

int dX = point2.X - point1.X;

float m = (float)dY / dX;

float im = 1.0f/m;

// mulai menggambar titik-titik

glBegin(GL\_POINTS);

// koordinat titik awal

glVertex3f(point1.X, point1.Y, point1.Z);

float pX = point1.X, pY = point1.Y, pZ = point1.Z;

// kenaikan terhadap Y

for (int i = point1.Y; i < point2.Y; i++)

{

pX = pX + im; // Xn+1 = Xn + 1/m

pY = pY + 1; // Yn+1 = Yn + 1

glVertex3f(pX, pY, pZ);

}

// koordinat titik akhir

glVertex3f(point2.X, point2.Y, point2.Z);

glEnd();

}

// fungsi untuk menggambar garis dengan algoritma DDA

void lineDDA(Vec3 point1, Vec3 point2)

{

// hitung selisih panjang

int dY = point2.Y - point1.Y;

int dX = point2.X - point1.X;

// bila deltaY lebih pendek dari deltaX

if (abs(dY) < abs(dX))

{

if (point1.X < point2.X) // bila X1 < X2

lineDDAX(point1, point2);

else // bila X1 > X2 maka dibalik

lineDDAX(point2, point1);

}

else // bila deltaY lebih panjang dari deltaX

{

if (point1.Y < point2.Y) // bila Y1 < Y2

lineDDAY(point1, point2);

else // bila Y1 > Y2 maka dibalik

lineDDAY(point2, point1);

}

}

// fungsi untuk menggambar obyek

void drawObject()

{

glPushMatrix();

// operasi transformasi rotasi obyek ke arah kanan-kiri

glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);

glPushMatrix();

// operasi transformasi rotasi obyek ke arah atas-bawah

glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

// set warna obyek ke warna hijau (0.0f, 1.0f, 0.0f)

glColor3f(0.1f, 0.5f, 0.5f);

// gambar sumbu

Vec3 sbY1 = Vec3( 0.0f,-300.0f, 0.0f);

Vec3 sbY2 = Vec3( 0.0f, 300.0f, 0.0f);

Vec3 sbX1 = Vec3(-300.0f, 0.0f, 0.0f);

Vec3 sbX2 = Vec3( 300.0f, 0.0f, 0.0f);

lineDDA(sbX1, sbX2);

lineDDA(sbY1, sbY2);

// kuadran 1

Vec3 point1 = Vec3( 50.0f, 150.0f, 0.0f);

Vec3 point2 = Vec3( 200.0f, 200.0f, 0.0f);

lineDDA(point1, point2);

// kuadran 2

point1 = Vec3(-120.0f, 200.0f, 0.0f);

point2 = Vec3(-50.0f, 80.0f, 0.0f);

lineDDA(point1, point2);

// kuadran 3

point1 = Vec3(-120.0f, -10.0f, 0.0f);

point2 = Vec3(-210.0f, -90.0f, 0.0f);

lineDDA(point1, point2);

// kuadran 4

point1 = Vec3( 150.0f, -100.0f, 0.0f);

point2 = Vec3( 180.0f, -220.0f, 0.0f);

lineDDA(point1, point2);

glPopMatrix();

glPopMatrix();

}

// taruh semua obyek yang akan digambar di fungsi display()

void display()

{

// bersihkan dan reset layar dan buffer

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

// posisikan kamera pandang

// dalam hal ini sumbu Y ada diatas dan posisi kamera pandang di (posX, posY, posZ)

gluLookAt(posX, posY, posZ, posX + rotX, posY + rotY, posZ + rotZ, 0.0f, 1.0f, 0.0f);

// panggil fungsi untuk menggambar obyek

drawObject();

// tampilkan obyek ke layar

// gunakan glFlush() bila memakai single buffer

// gunakan glutSwapBuffers() bila memakai double buffer

glutSwapBuffers();

}

// inisialisasikan variabel, pencahayaan, tekstur dan pengaturan kamera pandang di fungsi init()

void init(void)

{

// inisialisasi warna latar belakang layar dalam hal ini warna putih (1.0, 1.0, 1.0, 0.0)

glClearColor(0.0, 0.0, 0.0, 0.0);

glEnable(GL\_DEPTH\_TEST); // mengaktifkan depth buffer

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

//gluPerspective(45.0, 1.0, 1.0, 100.0);

glOrtho((GLfloat)-SCREEN\_WIDTH/2, (GLfloat)SCREEN\_WIDTH/2, (GLfloat)-SCREEN\_HEIGHT/2, (GLfloat)SCREEN\_HEIGHT/2, 1.0, 100.0); // set proyeksi ke perspektif

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// inisialisasi kamera pandang

gluLookAt(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

}

// fungsi ini digunakan bila layar akan diresize (default)

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

//gluPerspective(45, (GLfloat)w / (GLfloat)h, 1.0, 100.0);

glOrtho((GLfloat)-w/2, (GLfloat)w/2, (GLfloat)-h/2, (GLfloat)h/2, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

}

// fungsi untuk mengatur masukan dari keyboard

// untuk arah kiri, kanan, atas, bawah, PgUp, dan PgDn

void keyboard(int key, int x, int y)

{

float fraction = 0.1f;

switch (key)

{

// masukkan perintah disini bila tombol kiri ditekan

case GLUT\_KEY\_LEFT:

// dalam hal ini perintah rotasi obyek ke kiri sebanyak 1 derajat

objectAngleY -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol kanan ditekan

case GLUT\_KEY\_RIGHT:

// dalam hal ini perintah rotasi obyek ke kanan sebanyak 1 derajat

objectAngleY += 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol atas ditekan

case GLUT\_KEY\_UP:

// dalam hal ini perintah rotasi obyek ke atas sebanyak 1 derajat

objectAngleX -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol bawah ditekan

case GLUT\_KEY\_DOWN:

// dalam hal ini perintah rotasi obyek ke bawah sebanyak 1 derajat

objectAngleX += 1.0f;

glutPostRedisplay(); // update obyek

break;

// zoom in

case GLUT\_KEY\_PAGE\_UP:

// masukkan perintah disini bila tombol PgUp ditekan

posX += rotX \* fraction;

posZ += rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

// zoom out

case GLUT\_KEY\_PAGE\_DOWN:

// masukkan perintah disini bila tombol PgDn ditekan

posX -= rotX \* fraction;

posZ -= rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

}

}

// timer untuk animasi (gunakan bila perlu)

void timer(int value)

{

glutPostRedisplay();

glutTimerFunc(55, timer, 0);

}

// program utama

int main(int argc, char\*\* argv)

{

// inisialisasi jendela OpenGL

// GLUT\_SINGLE berarti memakai single buffer

// GLUT\_DOUBLE berarti memakai double buffer

// GLUT\_RGB berarti mode tampilan yang dipakai RGB

// GLUT\_RGBA berarti mode tampilan yang dipakai RGBA

// GLUT\_DEPTH berarti memakai depth buffer

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA | GLUT\_DEPTH);

// set ukuran jendela tampilan

glutInitWindowSize(480, 480); // besarnya jendela dalam piksel dalam hal ini 300x300

glutInitWindowPosition(100, 100); // posisi jendela dilayar komputer dalam piksel

// judul jendela (wajib diubah dengan informasi NAMA / NIM - JUDUL PRAKTIKUM masing-masing)

glutCreateWindow("SIRLY ZIADATUL MUSTAFIDAH / 2200018345 - ALGORITMA GARIS DDA");

// panggil fungsi init untuk inisialisasi awal

init();

// event handler untuk display, reshape dan keyboard

glutDisplayFunc(display); // display

glutReshapeFunc(reshape); // reshape

glutSpecialFunc(keyboard); // keyboard

//glutTimerFunc(0, timer, 0); // aktifkan timer bila perlu

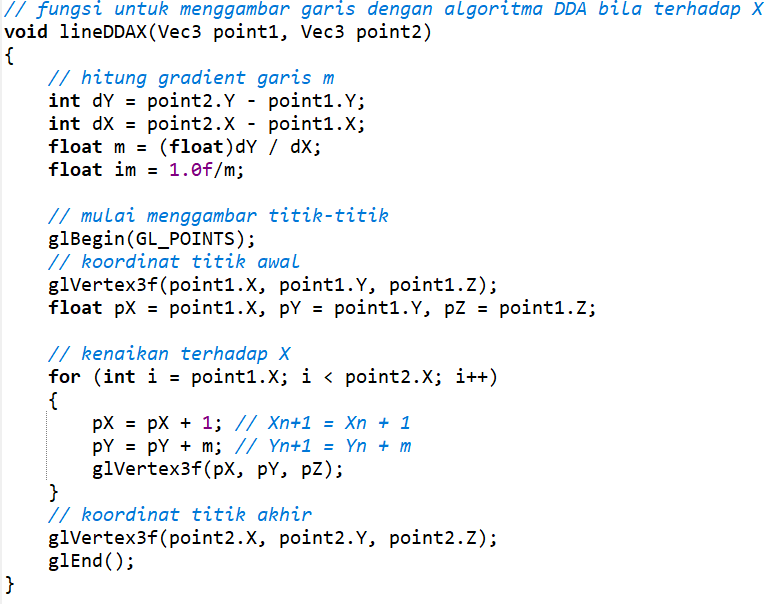
// looping

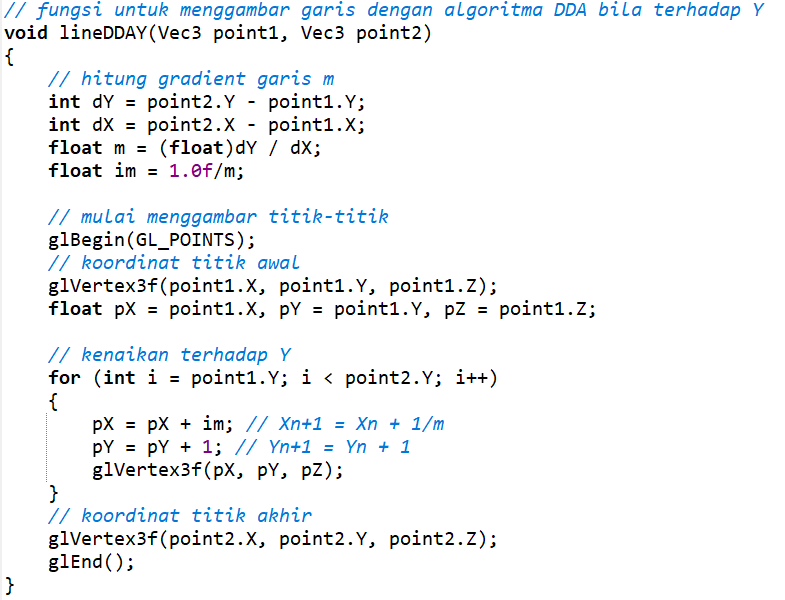
glutMainLoop();

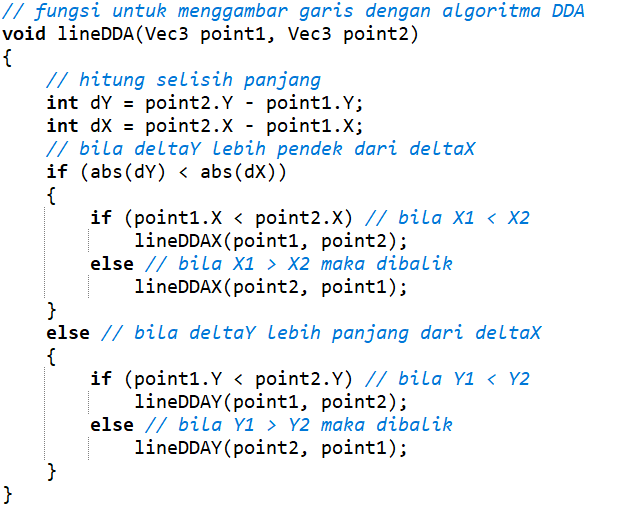
return 0;

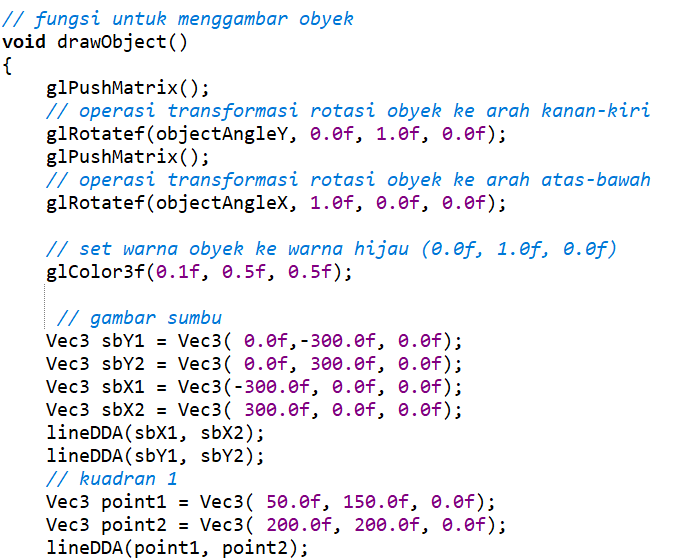
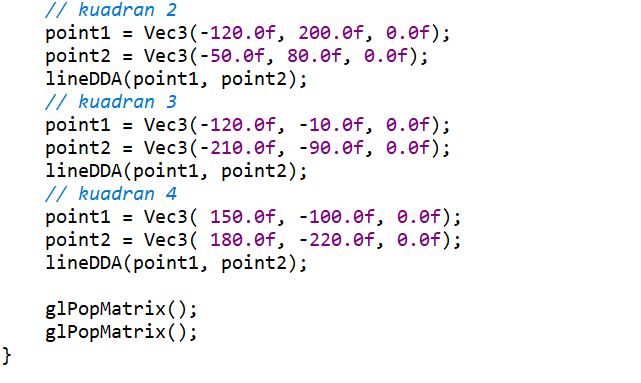
}

Penjelasan

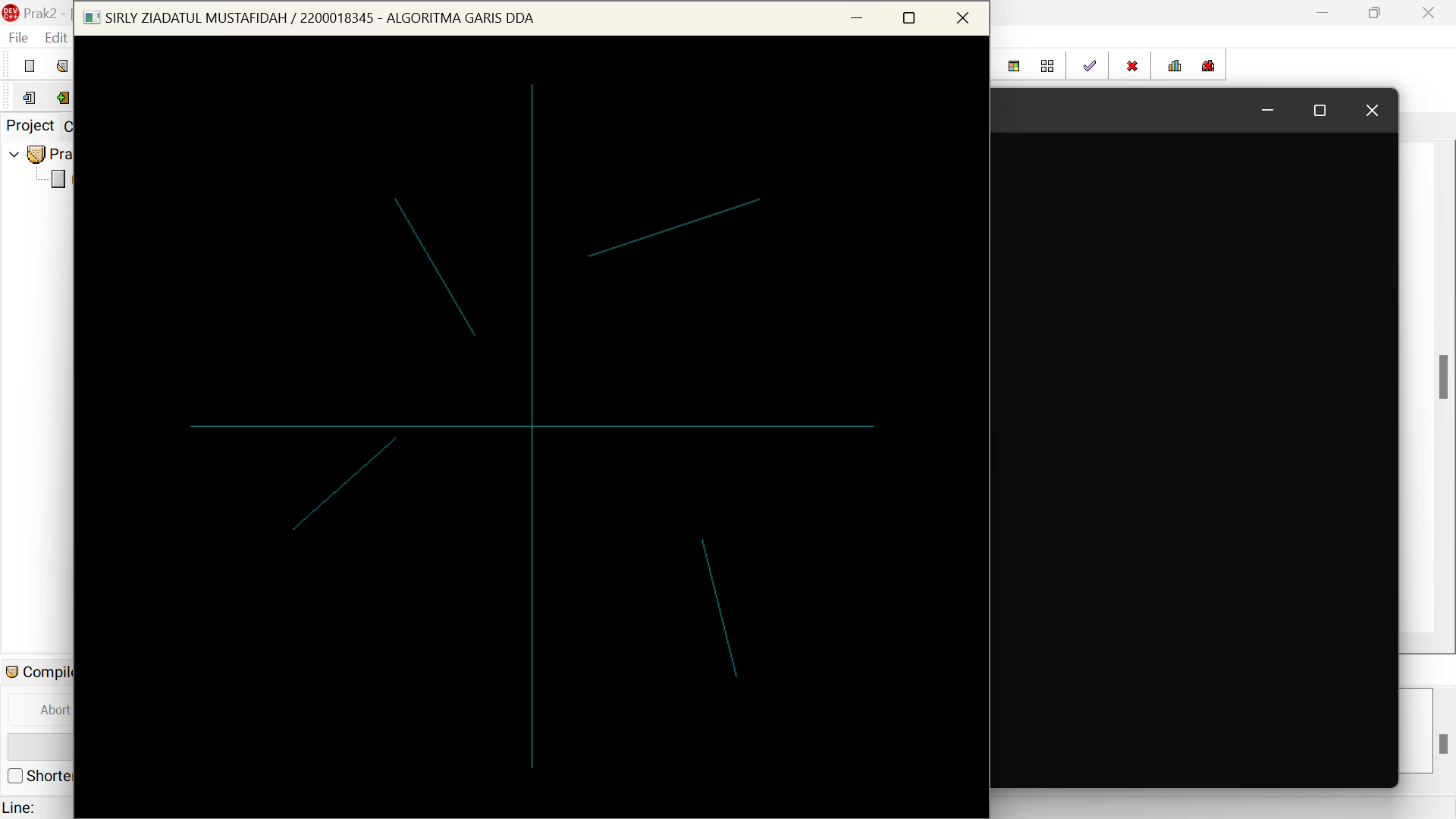






Output



Gambar 2. 1

Algoritma Bresenham

Source Code

// kode dasar untuk semua praktikum grafika komputer

// adhi prahara. 2018

// deklarasikan semua header disini

#include <windows.h>

#include <GL/glut.h>

#include <stdio.h>

#include <iostream>

int SCREEN\_WIDTH = 400;

int SCREEN\_HEIGHT = 400;

// inisialisasi variabel untuk transformasi seperti translasi, rotasi atau scaling

float angle = 0.0f; // sudut transformasi kamera

float posX = 0.0f, rotX = 0.0f; // posisi kamera di sumbu X

float posY = 0.0f, rotY = 0.0f; // posisi kamera di sumbu Y

float posZ = 5.0f, rotZ = -1.0f; // posisi kamera di sumbu Z

float objectAngleX = 0.0f; // sudut tranformasi obyek di sumbu X

float objectAngleY = 0.0f; // sudut tranformasi obyek di sumbu Y

float objectAngleZ = 0.0f; // sudut tranformasi obyek di sumbu Z

// container untuk membuat tipe data 3D (X, Y, Z)

struct Vec3{

float X; float Y; float Z;

Vec3(float x, float y, float z) { X = x; Y = y; Z = z; }

//

Vec3() { }

~Vec3() { }

};

// fungsi untuk menggambar garis dengan algoritma Bresenham

// bila slopenya terhadap X

void lineBresenhamX(Vec3 point1, Vec3 point2)

{

// hitung selisih panjang

int dY = point2.Y - point1.Y;

int dX = point2.X - point1.X;

int yi = 1; // skala penambahan

// bila delta Y kurang dari 0

if (dY < 0)

{

yi = -1;

dY = -dY;

}

// mulai menggambar titik-titik

glBegin(GL\_POINTS);

// koordinat titik awal

glVertex3f(point1.X, point1.Y, point1.Z);

int pX = point1.X, pY = point1.Y, pZ = point1.Z;

int dY2 = 2\*dY; // hitung 2\*deltaY

int dX2 = 2\*dX; // hitung 2\*deltaX

int pk = dY2 - dX; // hitung p0

// kenaikan terhadap X

for (int i = point1.X; i < point2.X; i++)

{

if (pk < 0) // bila p < 0

{

pk = pk + dY2; // update pk+1 = pk + 2dY

pX = pX + 1; // Xn+1 = Xn + 1

pY = pY; // Yn+1 = Yn

}

else // bila p >= 0

{

pk = pk + dY2 - dX2; // update pk+1 = pk + 2dY - 2dX

pX = pX + 1; // Xn+1 = Xn + 1

pY = pY + yi; // Yn+1 = Yn + yi

}

glVertex3f(pX, pY, pZ);

}

// koordinat titik akhir

glVertex3f(point2.X, point2.Y, point2.Z);

glEnd();

}

// fungsi untuk menggambar garis dengan algoritma Bresenham

// bila slopenya terhadap Y

void lineBresenhamY(Vec3 point1, Vec3 point2)

{

// hitung selisih panjang

int dY = point2.Y - point1.Y;

int dX = point2.X - point1.X;

int xi = 1; // skala penambahan

// bila delta X kurang dari 0

if (dX < 0)

{

xi = -1;

dX = -dX;

}

// mulai menggambar titik-titik

glBegin(GL\_POINTS);

// koordinat titik awal

glVertex3f(point1.X, point1.Y, point1.Z);

int pX = point1.X, pY = point1.Y, pZ = point1.Z;

int dY2 = 2\*dY; // hitung 2\*deltaY

int dX2 = 2\*dX; // hitung 2\*deltaX

int pk = dX2 - dY; // hitung p0

// kenaikan terhadap Y

for (int i = point1.Y; i < point2.Y; i++)

{

if (pk < 0) // bila p < 0

{

pk = pk + dX2; // update pk+1 = pk + 2dX

pX = pX; // Xn+1 = Xn

pY = pY + 1; // Yn+1 = Yn + 1

}

else // bila p >= 0

{

pk = pk + dX2 - dY2; // update pk+1 = pk + 2dX - 2dY

pX = pX + xi; // Xn+1 = Xn + xi

pY = pY + 1; // Yn+1 = Yn + 1

}

glVertex3f(pX, pY, pZ);

}

// koordinat titik akhir

glVertex3f(point2.X, point2.Y, point2.Z);

glEnd();

}

// fungsi untuk menggambar garis dengan algoritma Bresenham

void lineBresenham(Vec3 point1, Vec3 point2)

{

// hitung selisih panjang

int dY = point2.Y - point1.Y;

int dX = point2.X - point1.X;

if (abs(dY) < abs(dX)) // bila deltaY lebih pendek dari deltaX

{

if (point1.X < point2.X) // bila X1 < X2

lineBresenhamX(point1, point2);

else // bila X1 > X2 maka dibalik

lineBresenhamX(point2, point1);

}

else // bila deltaY lebih panjang dari deltaX

{

if (point1.Y < point2.Y) // bila Y1 < Y2

lineBresenhamY(point1, point2);

else // bila Y1 > Y2 maka dibalik

lineBresenhamY(point2, point1);

}

}

// fungsi untuk menggambar obyek

void drawObject()

{

glPushMatrix();

// operasi transformasi rotasi obyek ke arah kanan-kiri

glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);

glPushMatrix();

// operasi transformasi rotasi obyek ke arah atas-bawah

glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

// set warna obyek ke warna hijau (0.0f, 1.0f, 0.0f)

glColor3f(0.1f, 0.5f, 0.5f);

// gambar sumbu

Vec3 sbY1 = Vec3( 0.0f,-300.0f, 0.0f);

Vec3 sbY2 = Vec3( 0.0f, 300.0f, 0.0f);

Vec3 sbX1 = Vec3(-300.0f, 0.0f, 0.0f);

Vec3 sbX2 = Vec3( 300.0f, 0.0f, 0.0f);

lineBresenham(sbX1, sbX2);

lineBresenham(sbY1, sbY2);

// kuadran 1

Vec3 point1 = Vec3( 50.0f, 150.0f, 0.0f);

Vec3 point2 = Vec3( 200.0f, 200.0f, 0.0f);

lineBresenham(point1, point2);

// kuadran 2

point1 = Vec3(-120.0f, 200.0f, 0.0f);

point2 = Vec3(-50.0f, 80.0f, 0.0f);

lineBresenham(point1, point2);

// kuadran 3

point1 = Vec3(-120.0f, -10.0f, 0.0f);

point2 = Vec3(-210.0f, -90.0f, 0.0f);

lineBresenham(point1, point2);

// kuadran 4

point1 = Vec3( 150.0f, -100.0f, 0.0f);

point2 = Vec3( 180.0f, -220.0f, 0.0f);

lineBresenham(point1, point2);

glPopMatrix();

glPopMatrix();

}

// taruh semua obyek yang akan digambar di fungsi display()

void display()

{

// bersihkan dan reset layar dan buffer

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

// posisikan kamera pandang

// dalam hal ini sumbu Y ada diatas dan posisi kamera pandang di (posX, posY, posZ)

gluLookAt(posX, posY, posZ, posX + rotX, posY + rotY, posZ + rotZ, 0.0f, 1.0f, 0.0f);

// panggil fungsi untuk menggambar obyek

drawObject();

// tampilkan obyek ke layar

// gunakan glFlush() bila memakai single buffer

// gunakan glutSwapBuffers() bila memakai double buffer

glutSwapBuffers();

}

// inisialisasikan variabel, pencahayaan, tekstur dan pengaturan kamera pandang di fungsi init()

void init(void)

{

// inisialisasi warna latar belakang layar dalam hal ini warna putih (1.0, 1.0, 1.0, 0.0)

glClearColor(0.0, 0.0, 0.0, 0.0);

glEnable(GL\_DEPTH\_TEST); // mengaktifkan depth buffer

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

//gluPerspective(45.0, 1.0, 1.0, 100.0);

glOrtho((GLfloat)-SCREEN\_WIDTH/2, (GLfloat)SCREEN\_WIDTH/2, (GLfloat)-SCREEN\_HEIGHT/2, (GLfloat)SCREEN\_HEIGHT/2, 1.0, 100.0); // set proyeksi ke perspektif

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// inisialisasi kamera pandang

gluLookAt(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

}

// fungsi ini digunakan bila layar akan diresize (default)

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

//gluPerspective(45, (GLfloat)w / (GLfloat)h, 1.0, 100.0);

glOrtho((GLfloat)-w/2, (GLfloat)w/2, (GLfloat)-h/2, (GLfloat)h/2, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

}

// fungsi untuk mengatur masukan dari keyboard

// untuk arah kiri, kanan, atas, bawah, PgUp, dan PgDn

void keyboard(int key, int x, int y)

{

float fraction = 0.1f;

switch (key)

{

// masukkan perintah disini bila tombol kiri ditekan

case GLUT\_KEY\_LEFT:

// dalam hal ini perintah rotasi obyek ke kiri sebanyak 1 derajat

objectAngleY -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol kanan ditekan

case GLUT\_KEY\_RIGHT:

// dalam hal ini perintah rotasi obyek ke kanan sebanyak 1 derajat

objectAngleY += 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol atas ditekan

case GLUT\_KEY\_UP:

// dalam hal ini perintah rotasi obyek ke atas sebanyak 1 derajat

objectAngleX -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol bawah ditekan

case GLUT\_KEY\_DOWN:

// dalam hal ini perintah rotasi obyek ke bawah sebanyak 1 derajat

objectAngleX += 1.0f;

glutPostRedisplay(); // update obyek

break;

// zoom in

case GLUT\_KEY\_PAGE\_UP:

// masukkan perintah disini bila tombol PgUp ditekan

posX += rotX \* fraction;

posZ += rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

// zoom out

case GLUT\_KEY\_PAGE\_DOWN:

// masukkan perintah disini bila tombol PgDn ditekan

posX -= rotX \* fraction;

posZ -= rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

}

}

// timer untuk animasi (gunakan bila perlu)

void timer(int value)

{

glutPostRedisplay();

glutTimerFunc(55, timer, 0);

}

// program utama

int main(int argc, char\*\* argv)

{

// inisialisasi jendela OpenGL

// GLUT\_SINGLE berarti memakai single buffer

// GLUT\_DOUBLE berarti memakai double buffer

// GLUT\_RGB berarti mode tampilan yang dipakai RGB

// GLUT\_RGBA berarti mode tampilan yang dipakai RGBA

// GLUT\_DEPTH berarti memakai depth buffer

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA | GLUT\_DEPTH);

// set ukuran jendela tampilan

glutInitWindowSize(480, 480); // besarnya jendela dalam piksel dalam hal ini 300x300

glutInitWindowPosition(100, 100); // posisi jendela dilayar komputer dalam piksel

// judul jendela (wajib diubah dengan informasi NAMA / NIM - JUDUL PRAKTIKUM masing-masing)

glutCreateWindow("SIRLY ZIADATUL MUSTAFIDAH / 2200018345 - ALGORITMA GARIS BRESENHAM");

// panggil fungsi init untuk inisialisasi awal

init();

// event handler untuk display, reshape dan keyboard

glutDisplayFunc(display); // display

glutReshapeFunc(reshape); // reshape

glutSpecialFunc(keyboard); // keyboard

//glutTimerFunc(0, timer, 0); // aktifkan timer bila perlu

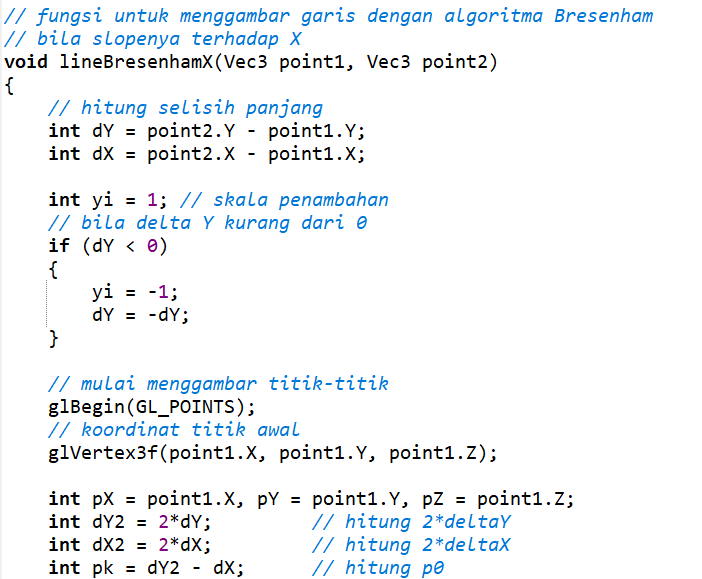
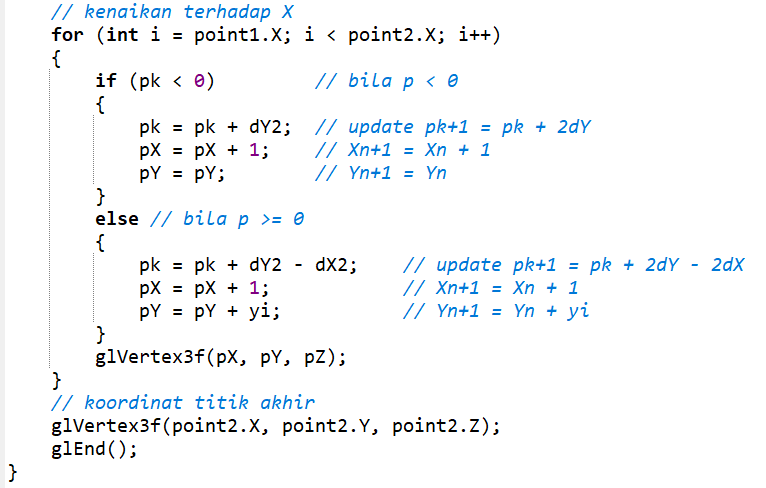
// looping

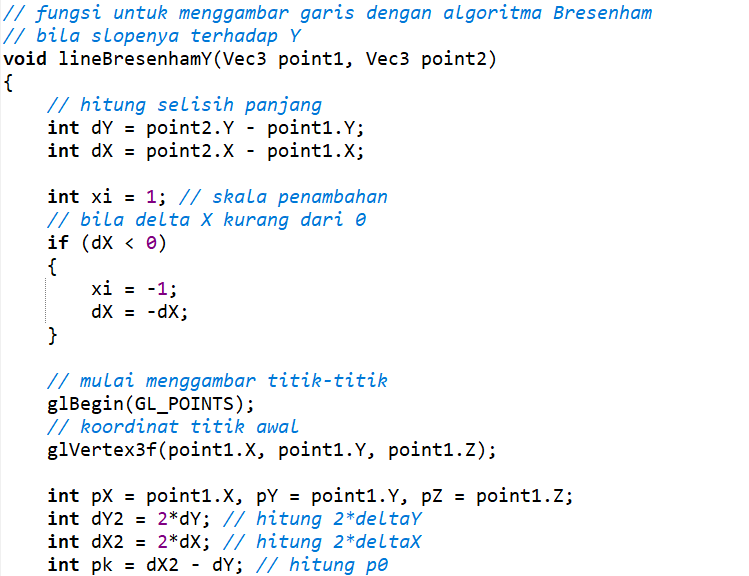
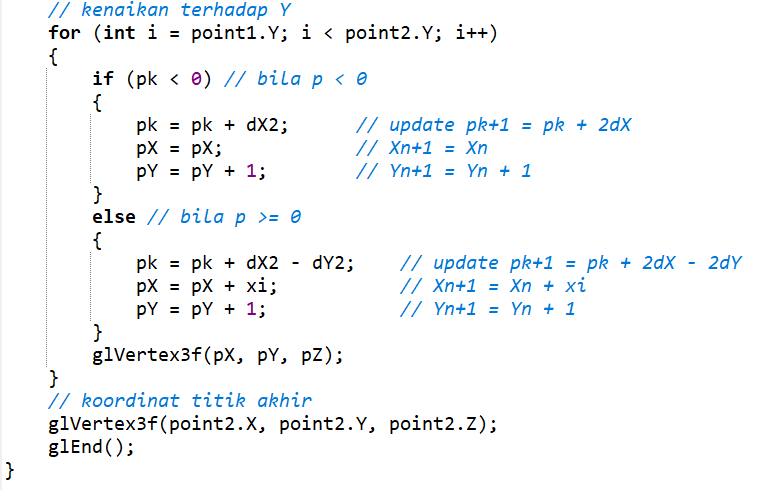
glutMainLoop();

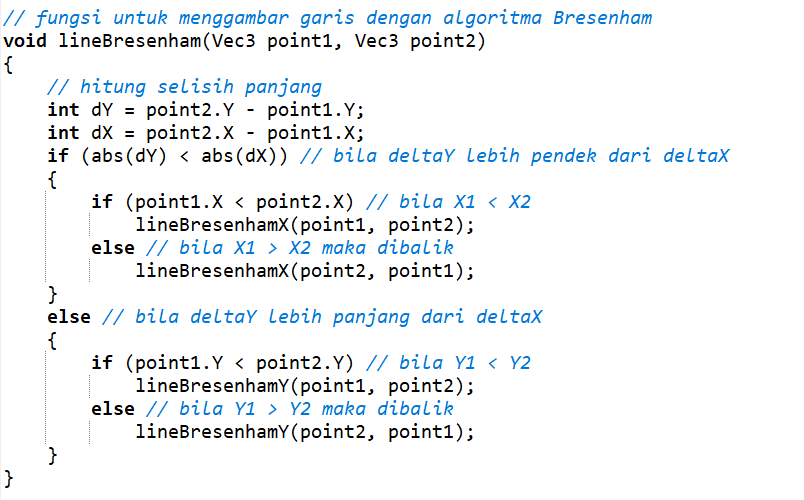
return 0;

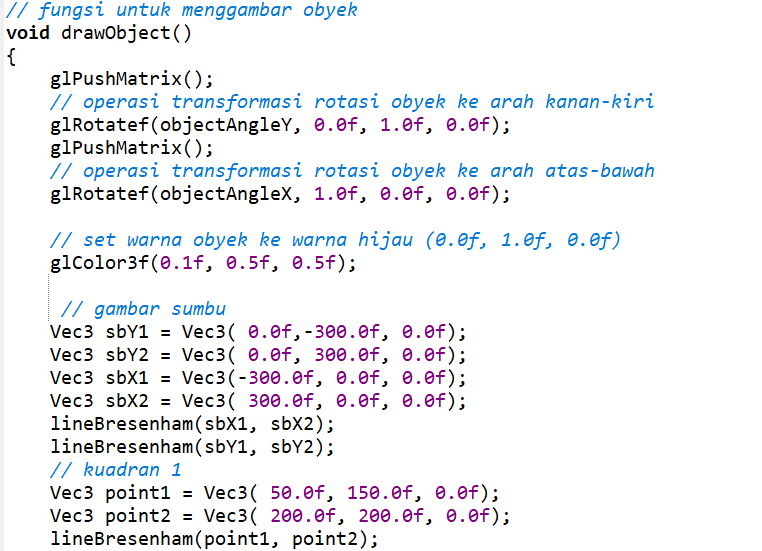
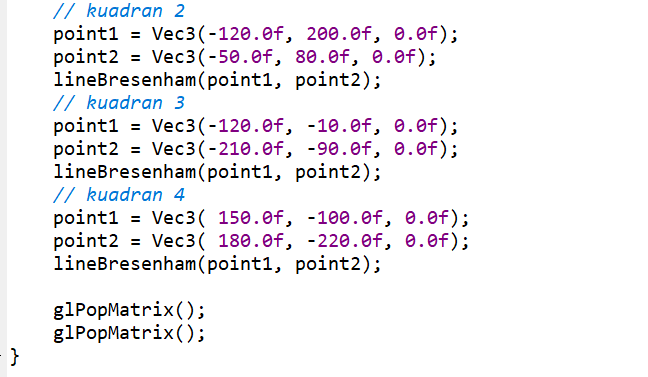
}

Penjelasan

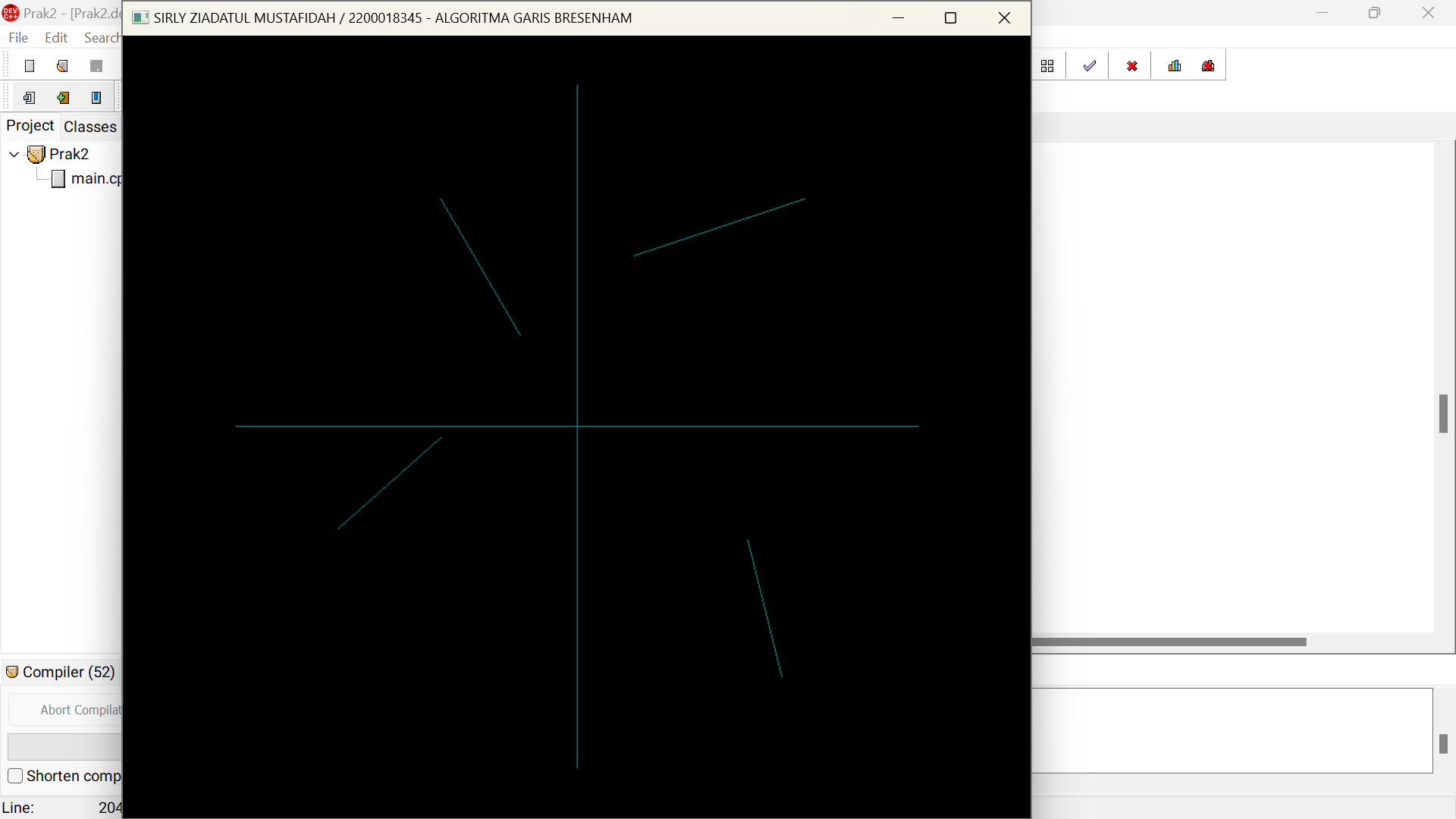
 



Output



Gambar 2. 2

1. Posttes 2

Source Code

// kode dasar untuk semua praktikum grafika komputer

// adhi prahara. 2018

// deklarasikan semua header disini

#include <windows.h>

#include <GL/glut.h>

#include <stdio.h>

#include <iostream>

int SCREEN\_WIDTH = 400;

int SCREEN\_HEIGHT = 400;

// inisialisasi variabel untuk transformasi seperti translasi, rotasi atau scaling

float angle = 0.0f; // sudut transformasi kamera

float posX = 0.0f, rotX = 0.0f; // posisi kamera di sumbu X

float posY = 0.0f, rotY = 0.0f; // posisi kamera di sumbu Y

float posZ = 5.0f, rotZ = -1.0f; // posisi kamera di sumbu Z

float objectAngleX = 0.0f; // sudut tranformasi obyek di sumbu X

float objectAngleY = 0.0f; // sudut tranformasi obyek di sumbu Y

float objectAngleZ = 0.0f; // sudut tranformasi obyek di sumbu Z

// container untuk membuat tipe data 3D (X, Y, Z)

struct Vec3{

float X; float Y; float Z;

Vec3(float x, float y, float z) { X = x; Y = y; Z = z; }

//

Vec3() { }

~Vec3() { }

};

// fungsi untuk menggambar garis dengan algoritma Bresenham

// bila slopenya terhadap X

void lineBresenhamX(Vec3 point1, Vec3 point2)

{

// hitung selisih panjang

int dY = point2.Y - point1.Y;

int dX = point2.X - point1.X;

int yi = 1; // skala penambahan

// bila delta Y kurang dari 0

if (dY < 0)

{

yi = -1;

dY = -dY;

}

// mulai menggambar titik-titik

glBegin(GL\_POINTS);

// koordinat titik awal

glVertex3f(point1.X, point1.Y, point1.Z);

int pX = point1.X, pY = point1.Y, pZ = point1.Z;

int dY2 = 2\*dY; // hitung 2\*deltaY

int dX2 = 2\*dX; // hitung 2\*deltaX

int pk = dY2 - dX; // hitung p0

// kenaikan terhadap X

for (int i = point1.X; i < point2.X; i++)

{

if (pk < 0) // bila p < 0

{

pk = pk + dY2; // update pk+1 = pk + 2dY

pX = pX + 1; // Xn+1 = Xn + 1

pY = pY; // Yn+1 = Yn

}

else // bila p >= 0

{

pk = pk + dY2 - dX2; // update pk+1 = pk + 2dY - 2dX

pX = pX + 1; // Xn+1 = Xn + 1

pY = pY + yi; // Yn+1 = Yn + yi

}

glVertex3f(pX, pY, pZ);

}

// koordinat titik akhir

glVertex3f(point2.X, point2.Y, point2.Z);

glEnd();

}

// fungsi untuk menggambar garis dengan algoritma Bresenham

// bila slopenya terhadap Y

void lineBresenhamY(Vec3 point1, Vec3 point2)

{

// hitung selisih panjang

int dY = point2.Y - point1.Y;

int dX = point2.X - point1.X;

int xi = 1; // skala penambahan

// bila delta X kurang dari 0

if (dX < 0)

{

xi = -1;

dX = -dX;

}

// mulai menggambar titik-titik

glBegin(GL\_POINTS);

// koordinat titik awal

glVertex3f(point1.X, point1.Y, point1.Z);

int pX = point1.X, pY = point1.Y, pZ = point1.Z;

int dY2 = 2\*dY; // hitung 2\*deltaY

int dX2 = 2\*dX; // hitung 2\*deltaX

int pk = dX2 - dY; // hitung p0

// kenaikan terhadap Y

for (int i = point1.Y; i < point2.Y; i++)

{

if (pk < 0) // bila p < 0

{

pk = pk + dX2; // update pk+1 = pk + 2dX

pX = pX; // Xn+1 = Xn

pY = pY + 1; // Yn+1 = Yn + 1

}

else // bila p >= 0

{

pk = pk + dX2 - dY2; // update pk+1 = pk + 2dX - 2dY

pX = pX + xi; // Xn+1 = Xn + xi

pY = pY + 1; // Yn+1 = Yn + 1

}

glVertex3f(pX, pY, pZ);

}

// koordinat titik akhir

glVertex3f(point2.X, point2.Y, point2.Z);

glEnd();

}

// fungsi untuk menggambar garis dengan algoritma Bresenham

void lineBresenham(Vec3 point1, Vec3 point2)

{

// hitung selisih panjang

int dY = point2.Y - point1.Y;

int dX = point2.X - point1.X;

if (abs(dY) < abs(dX)) // bila deltaY lebih pendek dari deltaX

{

if (point1.X < point2.X) // bila X1 < X2

lineBresenhamX(point1, point2);

else // bila X1 > X2 maka dibalik

lineBresenhamX(point2, point1);

}

else // bila deltaY lebih panjang dari deltaX

{

if (point1.Y < point2.Y) // bila Y1 < Y2

lineBresenhamY(point1, point2);

else // bila Y1 > Y2 maka dibalik

lineBresenhamY(point2, point1);

}

}

// fungsi untuk menggambar obyek

void drawObject()

{

glPushMatrix();

// operasi transformasi rotasi obyek ke arah kanan-kiri

glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);

glPushMatrix();

// operasi transformasi rotasi obyek ke arah atas-bawah

glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

// set warna obyek ke warna hijau (0.0f, 1.0f, 0.0f)

glColor3f(0.1f, 0.5f, 0.5f);

// gambar sumbu

Vec3 sbY1 = Vec3( 0.0f,-300.0f, 0.0f);

Vec3 sbY2 = Vec3( 0.0f, 300.0f, 0.0f);

Vec3 sbX1 = Vec3(-300.0f, 0.0f, 0.0f);

Vec3 sbX2 = Vec3( 300.0f, 0.0f, 0.0f);

lineBresenham(sbX1, sbX2);

lineBresenham(sbY1, sbY2);

// kuadran 1

Vec3 point1 = Vec3( 50.0f, 100.0f, 0.0f);

Vec3 point2 = Vec3( 250.0f, 100.0f, 0.0f);

Vec3 point3 = Vec3( 250.0f, 200.0f, 0.0f);

Vec3 point4 = Vec3( 50.0f, 200.0f, 0.0f);

lineBresenham(point1, point2);

lineBresenham(point2, point3);

lineBresenham(point3, point4);

lineBresenham(point4, point1);

//kuadran 2

point1 = Vec3(-70.0f, 100.0f, 0.0f);

point2 = Vec3(-50.0f, 200.0f, 0.0f);

point3 = Vec3(-230.0f, 200.0f, 0.0f);

point4 = Vec3(-250.0f, 100.0f, 0.0f);

lineBresenham(point1, point2);

lineBresenham(point2, point3);

lineBresenham(point3, point4);

lineBresenham(point4, point1);

// kuadran 3

point1 = Vec3(-150.0f, -100.0f, 0.0f);

point2 = Vec3(-100.0f, -200.0f, 0.0f);

point3 = Vec3(-200.0f, -200.0f, 0.0f);

lineBresenham(point1, point2);

lineBresenham(point2, point3);

lineBresenham(point3, point1);

// kuadran 4

point1 = Vec3( 100.0f, -100.0f, 0.0f);

point2 = Vec3( 200.0f, -100.0f, 0.0f);

point3 = Vec3( 150.0f, -200.0f, 0.0f);

lineBresenham(point1, point2);

lineBresenham(point2, point3);

lineBresenham(point3, point1);

glPopMatrix();

glPopMatrix();

}

// taruh semua obyek yang akan digambar di fungsi display()

void display()

{

// bersihkan dan reset layar dan buffer

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

// posisikan kamera pandang

// dalam hal ini sumbu Y ada diatas dan posisi kamera pandang di (posX, posY, posZ)

gluLookAt(posX, posY, posZ, posX + rotX, posY + rotY, posZ + rotZ, 0.0f, 1.0f, 0.0f);

// panggil fungsi untuk menggambar obyek

drawObject();

// tampilkan obyek ke layar

// gunakan glFlush() bila memakai single buffer

// gunakan glutSwapBuffers() bila memakai double buffer

glutSwapBuffers();

}

// inisialisasikan variabel, pencahayaan, tekstur dan pengaturan kamera pandang di fungsi init()

void init(void)

{

// inisialisasi warna latar belakang layar dalam hal ini warna putih (1.0, 1.0, 1.0, 0.0)

glClearColor(0.0, 0.0, 0.0, 0.0);

glEnable(GL\_DEPTH\_TEST); // mengaktifkan depth buffer

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

//gluPerspective(45.0, 1.0, 1.0, 100.0);

glOrtho((GLfloat)-SCREEN\_WIDTH/2, (GLfloat)SCREEN\_WIDTH/2, (GLfloat)-SCREEN\_HEIGHT/2, (GLfloat)SCREEN\_HEIGHT/2, 1.0, 100.0); // set proyeksi ke perspektif

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// inisialisasi kamera pandang

gluLookAt(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

}

// fungsi ini digunakan bila layar akan diresize (default)

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

//gluPerspective(45, (GLfloat)w / (GLfloat)h, 1.0, 100.0);

glOrtho((GLfloat)-w/2, (GLfloat)w/2, (GLfloat)-h/2, (GLfloat)h/2, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

}

// fungsi untuk mengatur masukan dari keyboard

// untuk arah kiri, kanan, atas, bawah, PgUp, dan PgDn

void keyboard(int key, int x, int y)

{

float fraction = 0.1f;

switch (key)

{

// masukkan perintah disini bila tombol kiri ditekan

case GLUT\_KEY\_LEFT:

// dalam hal ini perintah rotasi obyek ke kiri sebanyak 1 derajat

objectAngleY -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol kanan ditekan

case GLUT\_KEY\_RIGHT:

// dalam hal ini perintah rotasi obyek ke kanan sebanyak 1 derajat

objectAngleY += 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol atas ditekan

case GLUT\_KEY\_UP:

// dalam hal ini perintah rotasi obyek ke atas sebanyak 1 derajat

objectAngleX -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol bawah ditekan

case GLUT\_KEY\_DOWN:

// dalam hal ini perintah rotasi obyek ke bawah sebanyak 1 derajat

objectAngleX += 1.0f;

glutPostRedisplay(); // update obyek

break;

// zoom in

case GLUT\_KEY\_PAGE\_UP:

// masukkan perintah disini bila tombol PgUp ditekan

posX += rotX \* fraction;

posZ += rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

// zoom out

case GLUT\_KEY\_PAGE\_DOWN:

// masukkan perintah disini bila tombol PgDn ditekan

posX -= rotX \* fraction;

posZ -= rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

}

}

// timer untuk animasi (gunakan bila perlu)

void timer(int value)

{

glutPostRedisplay();

glutTimerFunc(55, timer, 0);

}

// program utama

int main(int argc, char\*\* argv)

{

// inisialisasi jendela OpenGL

// GLUT\_SINGLE berarti memakai single buffer

// GLUT\_DOUBLE berarti memakai double buffer

// GLUT\_RGB berarti mode tampilan yang dipakai RGB

// GLUT\_RGBA berarti mode tampilan yang dipakai RGBA

// GLUT\_DEPTH berarti memakai depth buffer

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA | GLUT\_DEPTH);

// set ukuran jendela tampilan

glutInitWindowSize(480, 480); // besarnya jendela dalam piksel dalam hal ini 300x300

glutInitWindowPosition(100, 100); // posisi jendela dilayar komputer dalam piksel

// judul jendela (wajib diubah dengan informasi NAMA / NIM - JUDUL PRAKTIKUM masing-masing)

glutCreateWindow("SIRLY ZIADATUL MUSTAFIDAH / 2200018345 - ALGORITMA GARIS BRESENHAM");

// panggil fungsi init untuk inisialisasi awal

init();

// event handler untuk display, reshape dan keyboard

glutDisplayFunc(display); // display

glutReshapeFunc(reshape); // reshape

glutSpecialFunc(keyboard); // keyboard

//glutTimerFunc(0, timer, 0); // aktifkan timer bila perlu

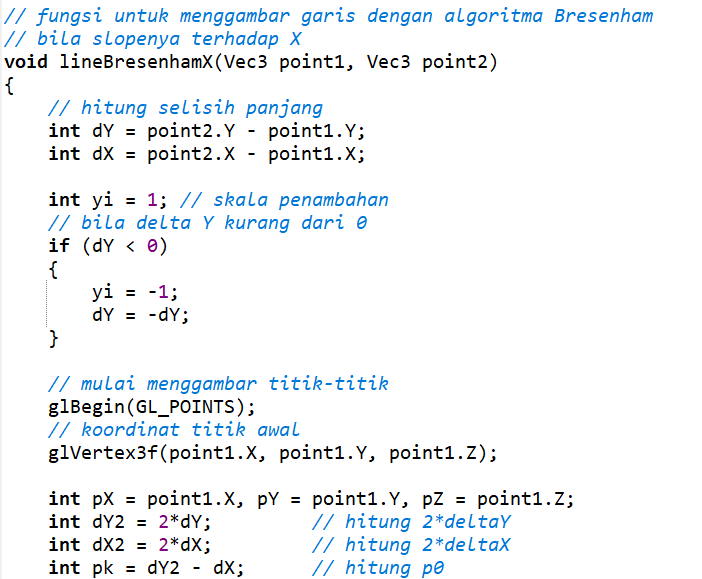
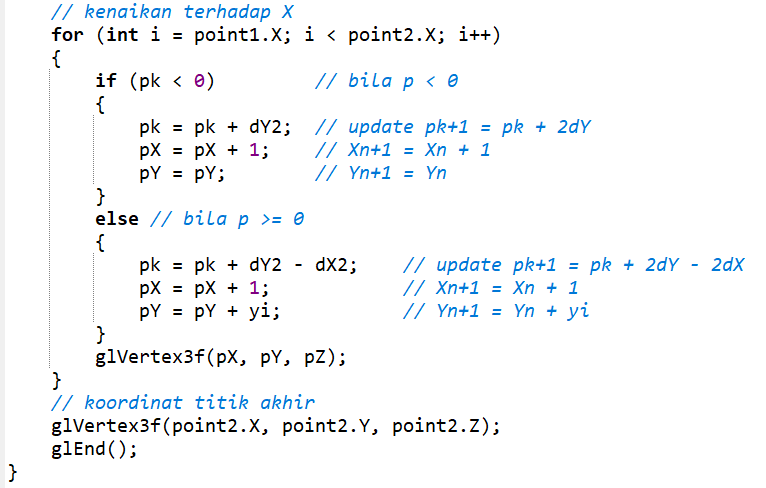
// looping

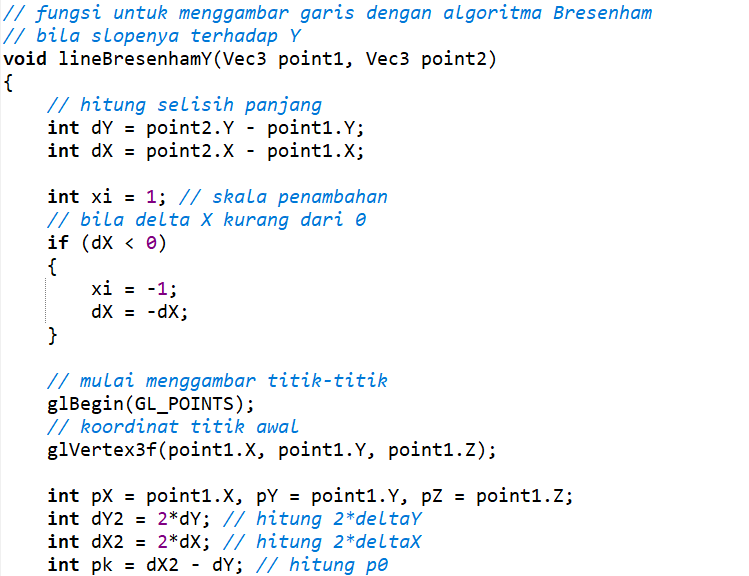
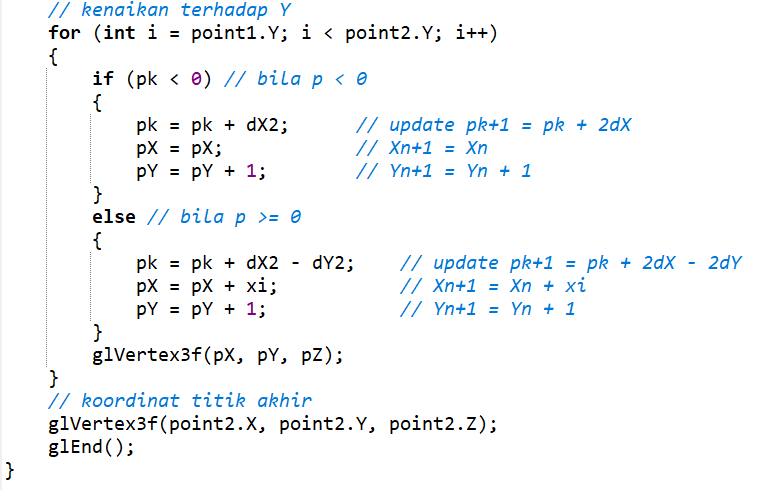
glutMainLoop();

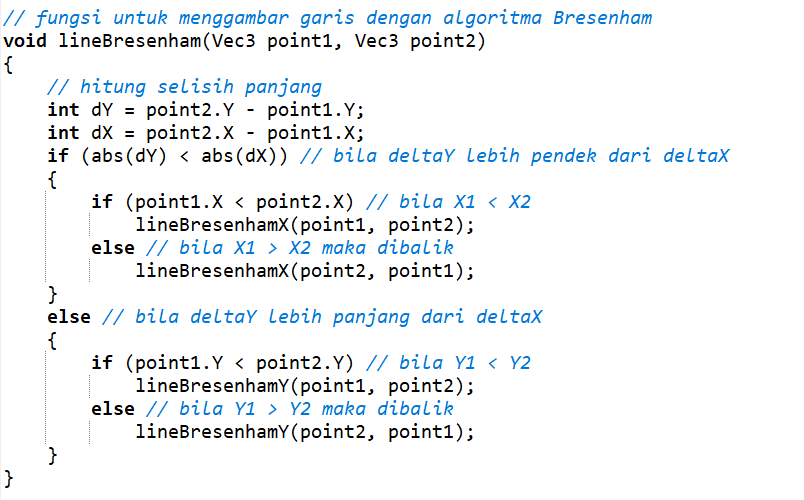
return 0;

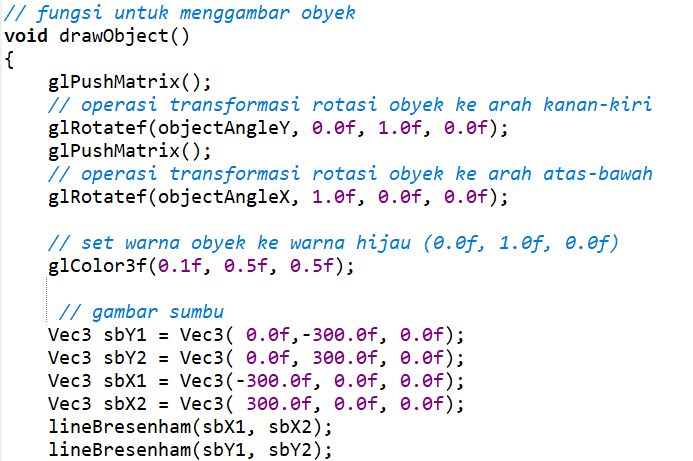
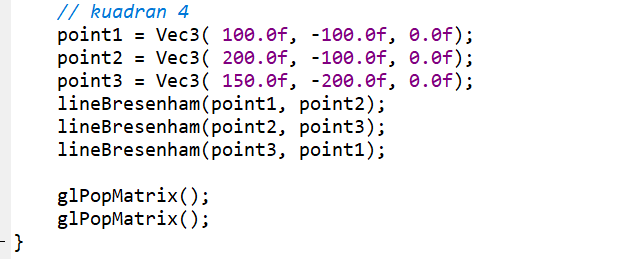
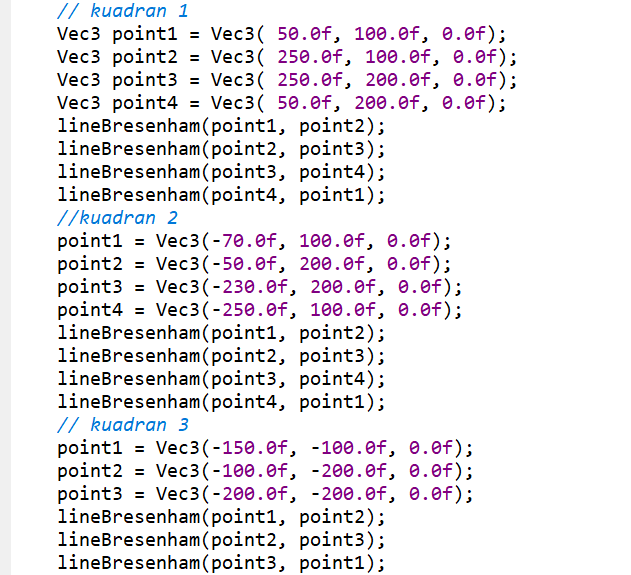
}

Penjelasan

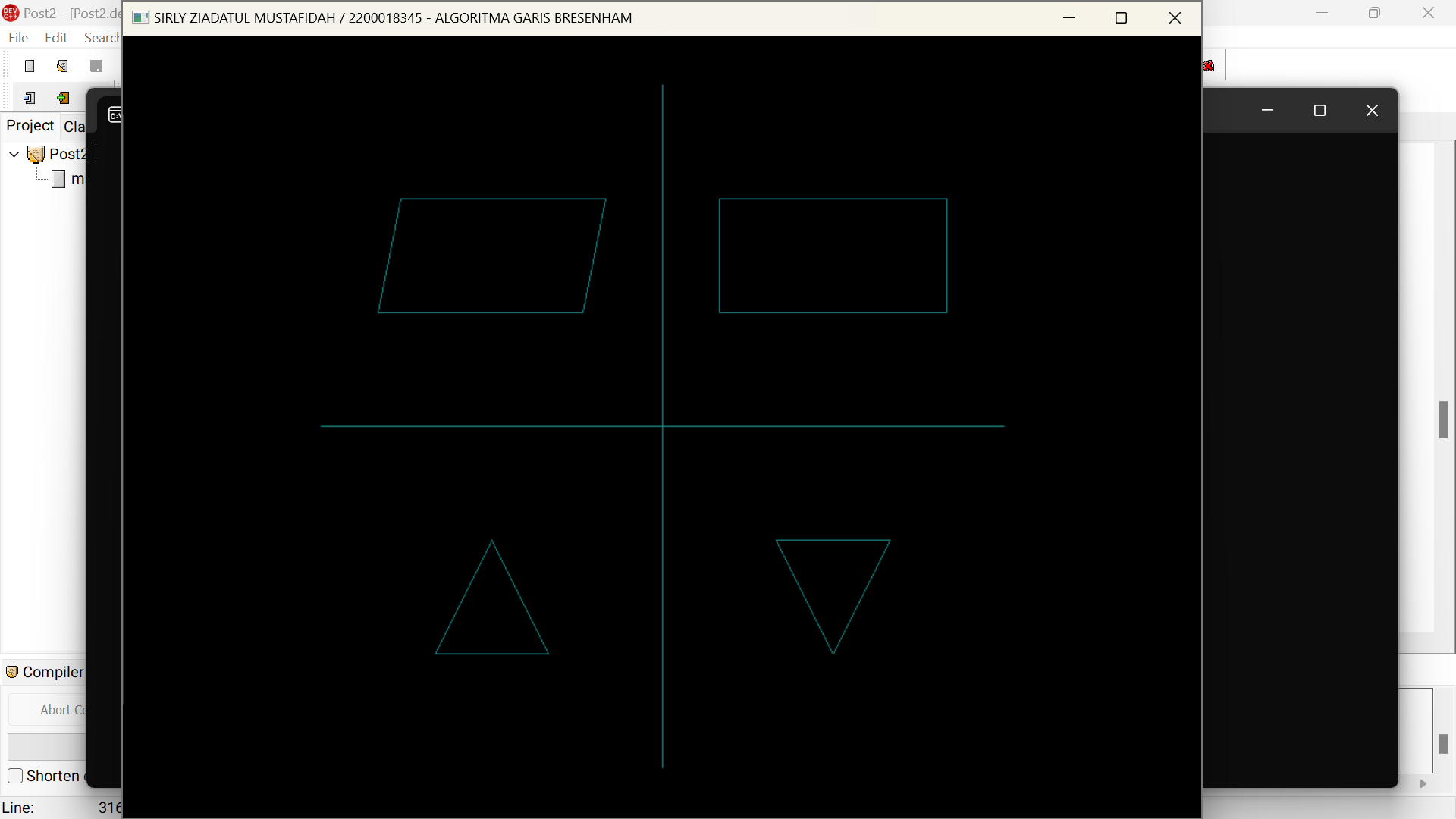
 



Output



Gambar 2. 3

# BAB III. INTERPOLASI DAN KURVA

1. Pretest 3
2. Jelaskan tahapan interpolasi linear!

Bila terdapat dua titik yang akan diinterpolasi yaitu (y0, y0) sampai (x1, y1). Bila jarak (x0, y0) sampai (x1, y1) dimisalkan 1 (dinormalisasikan) dan diketahui jarak awal (x0, y0) sampai titik sela (x, y) adalah u maka:

Dimana

1. Jelaskan tahapan interpolasi kubik!

Tahapan interpolasi kubik memerlukan 2 titik tambahan di ujung 2 titik utama untuk interpolasi. Jika terdapat 4 titik yang akan diinterpolasi yaitu (x0, y0), (x1, y1), (x2, y2) sampai (x3, y3) dan jarak tersebut dinormalisasikan menjadi 1 sedangkan jarak titik awal (x0, y0) sampai titk sela (x, y) adalah u dari dua titik tersebut, maka persamaannya:

Dimana:

* a = y3 + y2 – y0 + y1
* b = 2y0 – 2y1 – y3 + y2
* c = y2 – y0
* d = y1

1. Langkah Praktikum 3

Source Code

#include <windows.h>

#include <GL/glut.h>

#include <math.h>

// inisialisasi variabel untuk transformasi seperti translasi, rotasi atau scaling

float angle = 0.0f; // sudut transformasi kamera

float posX = 0.0f, rotX = 0.0f; // posisi kamera di sumbu X

float posY = 0.0f, rotY = 0.0f; // posisi kamera di sumbu Y

float posZ = 5.0f, rotZ = -1.0f; // posisi kamera di sumbu Z

float objectAngleX = 0.0f; // sudut tranformasi obyek di sumbu X

float objectAngleY = 0.0f; // sudut tranformasi obyek di sumbu Y

float objectAngleZ = 0.0f; // sudut tranformasi obyek di sumbu Z

#define PHI 3.141592654

#define SCREEN\_WIDTH 480

#define SCREEN\_HEIGHT 480

// container untuk membuat tipe data 3D (X, Y, Z)

struct Vec3

{

float X; float Y; float Z;

Vec3(float x, float y, float z) { X = x; Y = y; Z = z; }

//

Vec3() { }

~Vec3() { }

};

//langkah 5

// enumerate untuk tipe interpolation

enum INTERP\_TYPE

{

INTERP\_POINTS = 0,

INTERP\_LINES = 1,

INTERP\_LINEAR = 2,

INTERP\_COSINE = 3,

INTERP\_CUBIC = 4

};

//langkah 6

// fungsi untuk melakukan interpolasi linear dari dua titik

float linearInterpolate(float y0, float y1, float u)

{

return (y0 \* (1 - u) + y1 \* u);

}

//langkah 7

// fungsi untuk melakukan interpolasi cosine dari dua titik

float cosineInterpolate(float y0, float y1, float u)

{

float cosineU = (1 - cos(u \* PHI)) / 2;

return (y0 \* (1 - cosineU) + y1 \* cosineU);

}

//langkah 8

// fungsi untuk melakukan interpolasi cubic dari dua titik

float cubicInterpolate(float y0, float y1, float y2, float y3, float u)

{

float a = y3 - y2 - y0 + y1;

float b = 2 \* y0 - 2 \* y1 - y3 + y2;

float c = y2 - y0;

float d = y1;

return(a \* u \* u \* u + b \* u \* u + c \* u + d);

}

//langkah 9

// gambar garis hasil interpolasi

// n adalah jumlah titik yang dibuat

// type adalah tipe interpolasi yang digunakan

void drawInterpolation(

Vec3 point0,

Vec3 point1,// point1 adalah titik awal

Vec3 point2,// point2 adalah titik akhir

Vec3 point3,

int n,

INTERP\_TYPE type1,

INTERP\_TYPE type2)

{

float u = 0;

float stepU = 1.0f / n; // kenaikan u

float stepX = fabs(point2.X - point1.X) / n; // kenaikan x

float pX = point1.X, pY = point1.Y, pZ = point1.Z; // titik awal

// mulai menggambar titik-titik

glPointSize(5);

// kondisi menggambar titik

if (type1 == INTERP\_POINTS)

glBegin(GL\_POINTS);

// kondisi menggambar garis

else if (type1 == INTERP\_LINES)

glBegin(GL\_LINES);

for (int i = 0; i < n; i++)

{

glVertex3f(pX, pY, pZ);

pX = pX + stepX;

u = u + stepU;

// bila interpolasi linear

if (type2 == INTERP\_LINEAR)

pY = linearInterpolate(point1.Y, point2.Y, u);

// bila interpolasi cosine

else if (type2 == INTERP\_COSINE)

pY = cosineInterpolate(point1.Y, point2.Y, u);

// bila interpolasi cubic

else if (type2 == INTERP\_CUBIC)

pY = cubicInterpolate(point0.Y, point1.Y, point2.Y, point3.Y, u);

glVertex3f(pX, pY, pZ);

}

glEnd();

}

// fungsi untuk menggambar obyek

void drawObject()

{

glPushMatrix();

// operasi transformasi rotasi obyek ke arah kanan-kiri

glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);

glPushMatrix();

// operasi transformasi rotasi obyek ke arah atas-bawah

glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

// set warna obyek ke warna hijau (0.0f, 1.0f, 0.0f)

glColor3f(0.3f, 0.2f, 0.5f);

// Mendefinisikan banyak titik untuk menghasilkan kurva

Vec3 point0 = Vec3(-300.0f, 0.0f, 0.0f);

Vec3 point1 = Vec3(-200.0f, 100.0f, 0.0f);

Vec3 point2 = Vec3( 0.0f, -50.0f, 0.0f);

Vec3 point3 = Vec3( 200.0f, 100.0f, 0.0f);

Vec3 point4 = Vec3( 300.0f, 0.0f, 0.0f);

//langkah 10

// drawInterpolation(point0, point1, point2, point3, 10, INTERP\_POINTS,

// INTERP\_LINEAR);

// drawInterpolation(point0, point1, point2, point3, 10, INTERP\_LINES,

// INTERP\_LINEAR);

//langkah 12

// drawInterpolation(point0, point1, point2, point3, 10, INTERP\_POINTS,

// INTERP\_COSINE);

// drawInterpolation(point0, point1, point2, point3, 10, INTERP\_LINES,

// INTERP\_COSINE);

//Langkah 14 Menggambar kurva menggunakan interpolasi cubic

drawInterpolation(point0, point1, point2, point3, 50, INTERP\_POINTS, INTERP\_CUBIC);

drawInterpolation(point1, point2, point3, point4, 50, INTERP\_POINTS, INTERP\_CUBIC);

drawInterpolation(point0, point1, point2, point3, 50, INTERP\_LINES, INTERP\_CUBIC);

drawInterpolation(point1, point2, point3, point4, 50, INTERP\_LINES, INTERP\_CUBIC);

glPopMatrix();

glPopMatrix();

}

// taruh semua obyek yang akan digambar di fungsi display()

void display()

{

// bersihkan dan reset layar dan buffer

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

// posisikan kamera pandang

// dalam hal ini sumbu Y ada diatas dan posisi kamera pandang di (posX, posY, posZ)

gluLookAt(posX, posY, posZ, posX + rotX, posY + rotY, posZ + rotZ, 0.0f, 1.0f, 0.0f);

// panggil fungsi untuk menggambar obyek

drawObject();

// tampilkan obyek ke layar

// gunakan glFlush() bila memakai single buffer

// gunakan glutSwapBuffers() bila memakai double buffer

glutSwapBuffers();

}

// inisialisasikan variabel, pencahayaan, tekstur dan pengaturan kamera pandang di fungsi init()

void init(void)

{

// inisialisasi warna latar belakang layar dalam hal ini warna putih (1.0, 1.0, 1.0, 0.0)

//langkah 4

glClearColor(0.0, 0.0, 0.0, 0.0);

glEnable(GL\_DEPTH\_TEST); // mengaktifkan depth buffer

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

//langkah 2

glOrtho((GLfloat)-SCREEN\_WIDTH / 2, (GLfloat)SCREEN\_WIDTH / 2, (GLfloat)-SCREEN\_HEIGHT / 2, (GLfloat)SCREEN\_HEIGHT / 2, 1.0, 100.0);

// set proyeksi ke perspektif

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// inisialisasi kamera pandang

gluLookAt(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

}

// fungsi ini digunakan bila layar akan diresize (default)

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

//langkah 3

glOrtho((GLfloat)-w / 2, (GLfloat)w / 2, (GLfloat)-h / 2, (GLfloat)h / 2, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

}

// fungsi untuk mengatur masukan dari keyboard

// untuk arah kiri, kanan, atas, bawah, PgUp, dan PgDn

void keyboard(int key, int x, int y)

{

float fraction = 0.1f;

switch (key)

{

// masukkan perintah disini bila tombol kiri ditekan

case GLUT\_KEY\_LEFT:

// dalam hal ini perintah rotasi obyek ke kiri sebanyak 1 derajat

objectAngleY -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol kanan ditekan

case GLUT\_KEY\_RIGHT:

// dalam hal ini perintah rotasi obyek ke kanan sebanyak 1 derajat

objectAngleY += 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol atas ditekan

case GLUT\_KEY\_UP:

// dalam hal ini perintah rotasi obyek ke atas sebanyak 1 derajat

objectAngleX -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol bawah ditekan

case GLUT\_KEY\_DOWN:

// dalam hal ini perintah rotasi obyek ke bawah sebanyak 1 derajat

objectAngleX += 1.0f;

glutPostRedisplay(); // update obyek

break;

// zoom in

case GLUT\_KEY\_PAGE\_UP:

// masukkan perintah disini bila tombol PgUp ditekan

posX += rotX \* fraction;

posZ += rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

// zoom out

case GLUT\_KEY\_PAGE\_DOWN:

// masukkan perintah disini bila tombol PgDn ditekan

posX -= rotX \* fraction;

posZ -= rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

}

}

// timer untuk animasi (gunakan bila perlu)

void timer(int value)

{

glutPostRedisplay();

glutTimerFunc(55, timer, 0);

}

// program utama

int main(int argc, char\*\* argv)

{

// inisialisasi jendela OpenGL

// GLUT\_SINGLE berarti memakai single buffer

// GLUT\_DOUBLE berarti memakai double buffer

// GLUT\_RGB berarti mode tampilan yang dipakai RGB

// GLUT\_RGBA berarti mode tampilan yang dipakai RGBA

// GLUT\_DEPTH berarti memakai depth buffer

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA | GLUT\_DEPTH);

// set ukuran jendela tampilan

glutInitWindowSize(480, 480); // besarnya jendela dalam piksel dalam hal ini 480x480

glutInitWindowPosition(100, 100); // posisi jendela dilayar komputer dalam piksel

// judul jendela (wajib diubah dengan informasi NAMA / NIM - JUDUL PRAKTIKUM masing-masing)

glutCreateWindow("SIRLY ZIADATUL MUSTAFIDAH / 2200018345 - INTERPOLASI DAN KURVA");

// panggil fungsi init untuk inisialisasi awal

init();

// event handler untuk display, reshape dan keyboard

glutDisplayFunc(display); // display

glutReshapeFunc(reshape); // reshape

glutSpecialFunc(keyboard); // keyboard

//glutTimerFunc(0, timer, 0); // aktifkan timer bila perlu

// looping

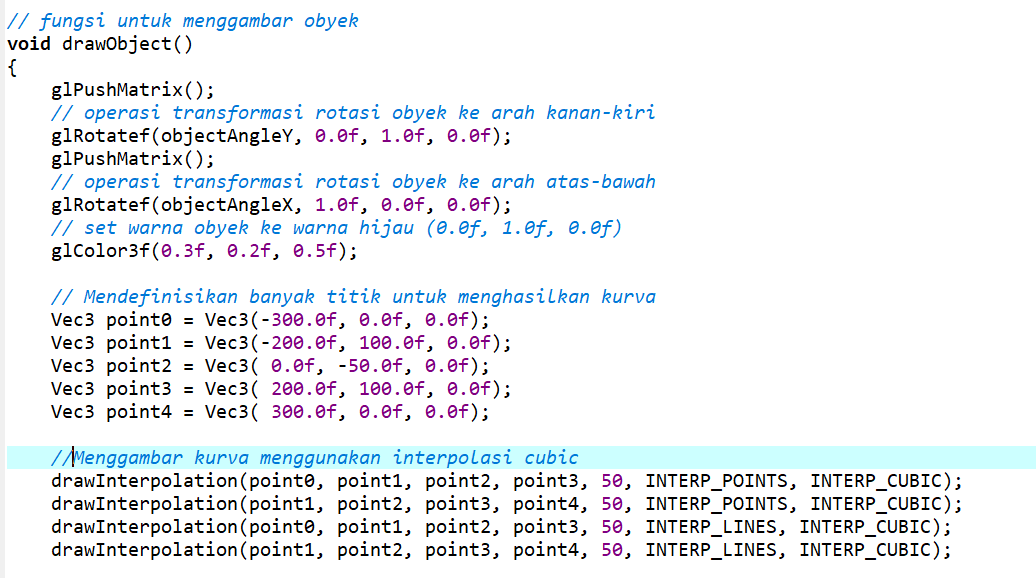
glutMainLoop();

//Mengembalikan nilai

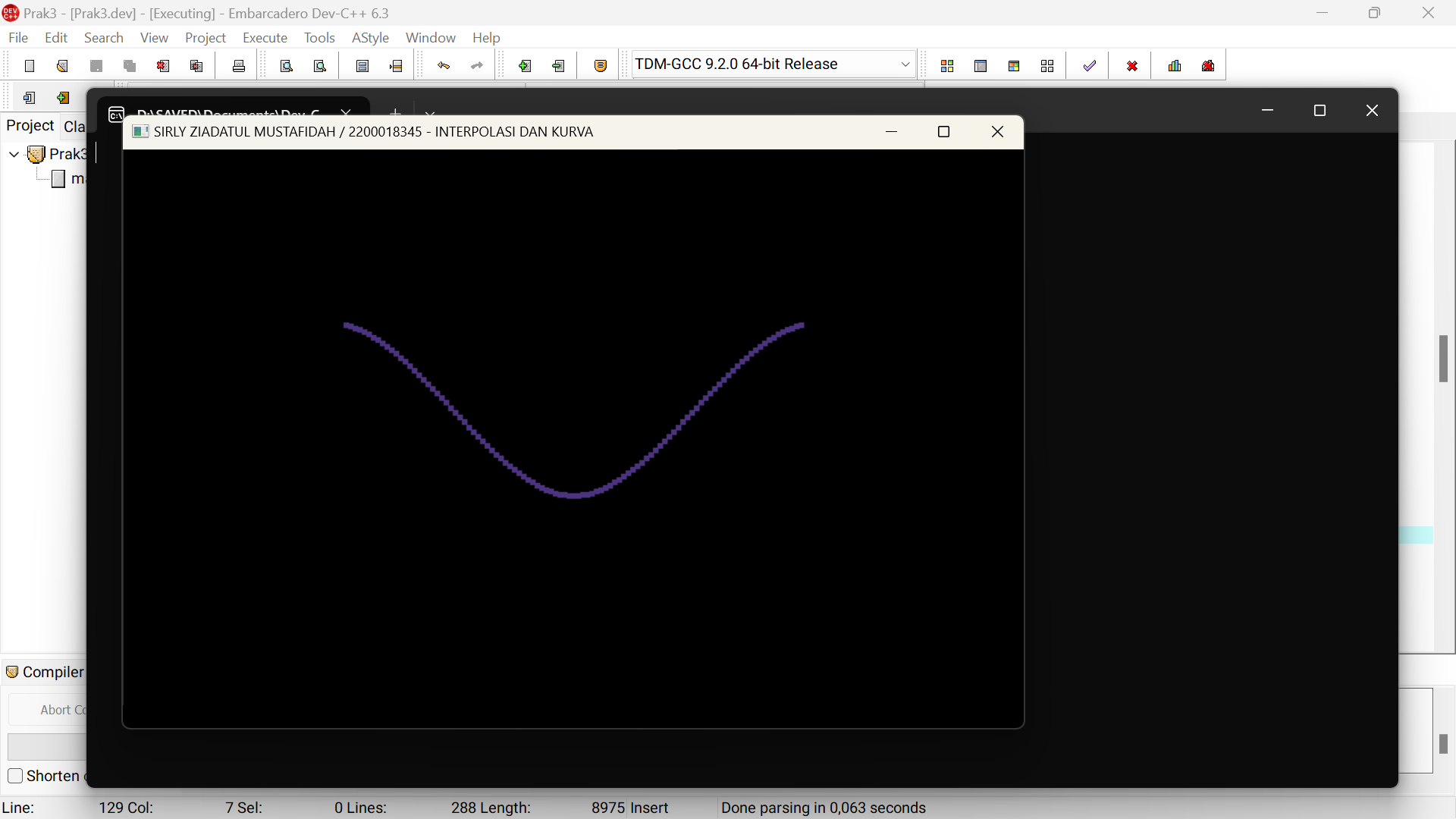
return 0;

}

Penjelasan



Output



Gambar 3. 1

1. Posttest 3

Source Code

#include <windows.h>

#include <GL/glut.h>

#include <math.h>

// inisialisasi variabel untuk transformasi seperti translasi, rotasi atau scaling

float angle = 0.0f; // sudut transformasi kamera

float posX = 0.0f, rotX = 0.0f; // posisi kamera di sumbu X

float posY = 0.0f, rotY = 0.0f; // posisi kamera di sumbu Y

float posZ = 5.0f, rotZ = -1.0f; // posisi kamera di sumbu Z

float objectAngleX = 0.0f; // sudut tranformasi obyek di sumbu X

float objectAngleY = 0.0f; // sudut tranformasi obyek di sumbu Y

float objectAngleZ = 0.0f; // sudut tranformasi obyek di sumbu Z

#define PHI 3.141592654

#define SCREEN\_WIDTH 480

#define SCREEN\_HEIGHT 480

// container untuk membuat tipe data 3D (X, Y, Z)

struct Vec3

{

float X; float Y; float Z;

Vec3(float x, float y, float z) { X = x; Y = y; Z = z; }

//

Vec3() { }

~Vec3() { }

};

//langkah 5

// enumerate untuk tipe interpolation

enum INTERP\_TYPE

{

INTERP\_POINTS = 0,

INTERP\_LINES = 1,

INTERP\_LINEAR = 2,

INTERP\_COSINE = 3,

INTERP\_CUBIC = 4

};

//langkah 6

// fungsi untuk melakukan interpolasi linear dari dua titik

float linearInterpolate(float y0, float y1, float u)

{

return (y0 \* (1 - u) + y1 \* u);

}

//langkah 7

// fungsi untuk melakukan interpolasi cosine dari dua titik

float cosineInterpolate(float y0, float y1, float u)

{

float cosineU = (1 - cos(u \* PHI)) / 2;

return (y0 \* (1 - cosineU) + y1 \* cosineU);

}

//langkah 8

// fungsi untuk melakukan interpolasi cubic dari dua titik

float cubicInterpolate(float y0, float y1, float y2, float y3, float u)

{

float a = y3 - y2 - y0 + y1;

float b = 2 \* y0 - 2 \* y1 - y3 + y2;

float c = y2 - y0;

float d = y1;

return(a \* u \* u \* u + b \* u \* u + c \* u + d);

}

//langkah 9

// gambar garis hasil interpolasi

// n adalah jumlah titik yang dibuat

// type adalah tipe interpolasi yang digunakan

void drawInterpolation(

Vec3 point0,

Vec3 point1,// point1 adalah titik awal

Vec3 point2,// point2 adalah titik akhir

Vec3 point3,

int n,

INTERP\_TYPE type1,

INTERP\_TYPE type2)

{

float u = 0;

float stepU = 1.0f / n; // kenaikan u

float stepX = fabs(point2.X - point1.X) / n; // kenaikan x

float pX = point1.X, pY = point1.Y, pZ = point1.Z; // titik awal

// mulai menggambar titik-titik

glPointSize(5);

// kondisi menggambar titik

if (type1 == INTERP\_POINTS)

glBegin(GL\_POINTS);

// kondisi menggambar garis

else if (type1 == INTERP\_LINES)

glBegin(GL\_LINES);

for (int i = 0; i < n; i++)

{

glVertex3f(pX, pY, pZ);

pX = pX + stepX;

u = u + stepU;

// bila interpolasi linear

if (type2 == INTERP\_LINEAR)

pY = linearInterpolate(point1.Y, point2.Y, u);

// bila interpolasi cosine

else if (type2 == INTERP\_COSINE)

pY = cosineInterpolate(point1.Y, point2.Y, u);

// bila interpolasi cubic

else if (type2 == INTERP\_CUBIC)

pY = cubicInterpolate(point0.Y, point1.Y, point2.Y, point3.Y, u);

glVertex3f(pX, pY, pZ);

}

glEnd();

}

// fungsi untuk menggambar obyek

void drawObject()

{

glPushMatrix();

// operasi transformasi rotasi obyek ke arah kanan-kiri

glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);

glPushMatrix();

// operasi transformasi rotasi obyek ke arah atas-bawah

glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

// set warna obyek ke warna hijau (0.0f, 1.0f, 0.0f)

glColor3f(0.3f, 0.2f, 0.5f);

// Mendefinisikan banyak titik untuk membentuk setengah lingkaran

Vec3 point0 = Vec3(-50.0f, 50.0f, 0.0f);

Vec3 point1 = Vec3(50.0f, 50.0f, 0.0f);

Vec3 point2 = Vec3(50.0f, -50.0f, 0.0f);

Vec3 point3 = Vec3(-50.0f, -50.0f, 0.0f);

Vec3 point4 = Vec3(-50.0f, 50.0f, 0.0f);

//Menggambar kurva menggunakan interpolasi cubic

drawInterpolation(point1, point2, point3, point4, 5, INTERP\_POINTS, INTERP\_CUBIC);

drawInterpolation(point1, point2, point3, point4, 5, INTERP\_LINES, INTERP\_CUBIC);

//langkah 10

// drawInterpolation(point0, point1, point2, point3, 10, INTERP\_POINTS,

// INTERP\_LINEAR);

// drawInterpolation(point0, point1, point2, point3, 10, INTERP\_LINES,

// INTERP\_LINEAR);

//langkah 12

// drawInterpolation(point0, point1, point2, point3, 10, INTERP\_POINTS,

// INTERP\_COSINE);

// drawInterpolation(point0, point1, point2, point3, 10, INTERP\_LINES,

// INTERP\_COSINE);

glPopMatrix();

glPopMatrix();

}

// taruh semua obyek yang akan digambar di fungsi display()

void display()

{

// bersihkan dan reset layar dan buffer

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

// posisikan kamera pandang

// dalam hal ini sumbu Y ada diatas dan posisi kamera pandang di (posX, posY, posZ)

gluLookAt(posX, posY, posZ, posX + rotX, posY + rotY, posZ + rotZ, 0.0f, 1.0f, 0.0f);

// panggil fungsi untuk menggambar obyek

drawObject();

// tampilkan obyek ke layar

// gunakan glFlush() bila memakai single buffer

// gunakan glutSwapBuffers() bila memakai double buffer

glutSwapBuffers();

}

// inisialisasikan variabel, pencahayaan, tekstur dan pengaturan kamera pandang di fungsi init()

void init(void)

{

// inisialisasi warna latar belakang layar dalam hal ini warna putih (1.0, 1.0, 1.0, 0.0)

//langkah 4

glClearColor(0.0, 0.0, 0.0, 0.0);

glEnable(GL\_DEPTH\_TEST); // mengaktifkan depth buffer

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

//langkah 2

glOrtho((GLfloat)-SCREEN\_WIDTH / 2, (GLfloat)SCREEN\_WIDTH / 2, (GLfloat)-SCREEN\_HEIGHT / 2, (GLfloat)SCREEN\_HEIGHT / 2, 1.0, 100.0);

// set proyeksi ke perspektif

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// inisialisasi kamera pandang

gluLookAt(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

}

// fungsi ini digunakan bila layar akan diresize (default)

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

//langkah 3

glOrtho((GLfloat)-w / 2, (GLfloat)w / 2, (GLfloat)-h / 2, (GLfloat)h / 2, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

}

// fungsi untuk mengatur masukan dari keyboard

// untuk arah kiri, kanan, atas, bawah, PgUp, dan PgDn

void keyboard(int key, int x, int y)

{

float fraction = 0.1f;

switch (key)

{

// masukkan perintah disini bila tombol kiri ditekan

case GLUT\_KEY\_LEFT:

// dalam hal ini perintah rotasi obyek ke kiri sebanyak 1 derajat

objectAngleY -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol kanan ditekan

case GLUT\_KEY\_RIGHT:

// dalam hal ini perintah rotasi obyek ke kanan sebanyak 1 derajat

objectAngleY += 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol atas ditekan

case GLUT\_KEY\_UP:

// dalam hal ini perintah rotasi obyek ke atas sebanyak 1 derajat

objectAngleX -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol bawah ditekan

case GLUT\_KEY\_DOWN:

// dalam hal ini perintah rotasi obyek ke bawah sebanyak 1 derajat

objectAngleX += 1.0f;

glutPostRedisplay(); // update obyek

break;

// zoom in

case GLUT\_KEY\_PAGE\_UP:

// masukkan perintah disini bila tombol PgUp ditekan

posX += rotX \* fraction;

posZ += rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

// zoom out

case GLUT\_KEY\_PAGE\_DOWN:

// masukkan perintah disini bila tombol PgDn ditekan

posX -= rotX \* fraction;

posZ -= rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

}

}

// timer untuk animasi (gunakan bila perlu)

void timer(int value)

{

glutPostRedisplay();

glutTimerFunc(55, timer, 0);

}

// program utama

int main(int argc, char\*\* argv)

{

// inisialisasi jendela OpenGL

// GLUT\_SINGLE berarti memakai single buffer

// GLUT\_DOUBLE berarti memakai double buffer

// GLUT\_RGB berarti mode tampilan yang dipakai RGB

// GLUT\_RGBA berarti mode tampilan yang dipakai RGBA

// GLUT\_DEPTH berarti memakai depth buffer

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA | GLUT\_DEPTH);

// set ukuran jendela tampilan

glutInitWindowSize(480, 480); // besarnya jendela dalam piksel dalam hal ini 480x480

glutInitWindowPosition(100, 100); // posisi jendela dilayar komputer dalam piksel

// judul jendela (wajib diubah dengan informasi NAMA / NIM - JUDUL PRAKTIKUM masing-masing)

glutCreateWindow("SIRLY ZIADATUL MUSTAFIDAH / 2200018345 - INTERPOLASI DAN KURVA");

// panggil fungsi init untuk inisialisasi awal

init();

// event handler untuk display, reshape dan keyboard

glutDisplayFunc(display); // display

glutReshapeFunc(reshape); // reshape

glutSpecialFunc(keyboard); // keyboard

//glutTimerFunc(0, timer, 0); // aktifkan timer bila perlu

// looping

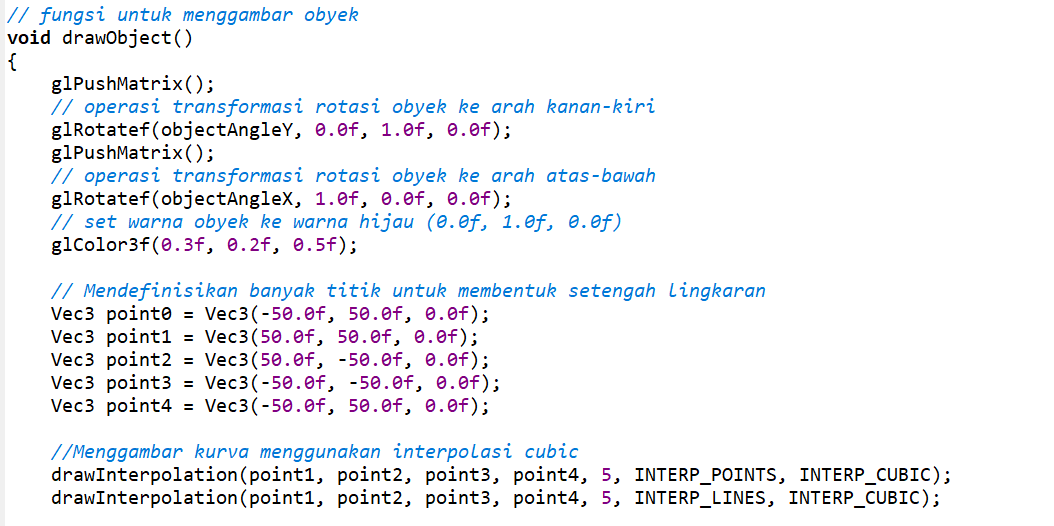
glutMainLoop();

//Mengembalikan nilai

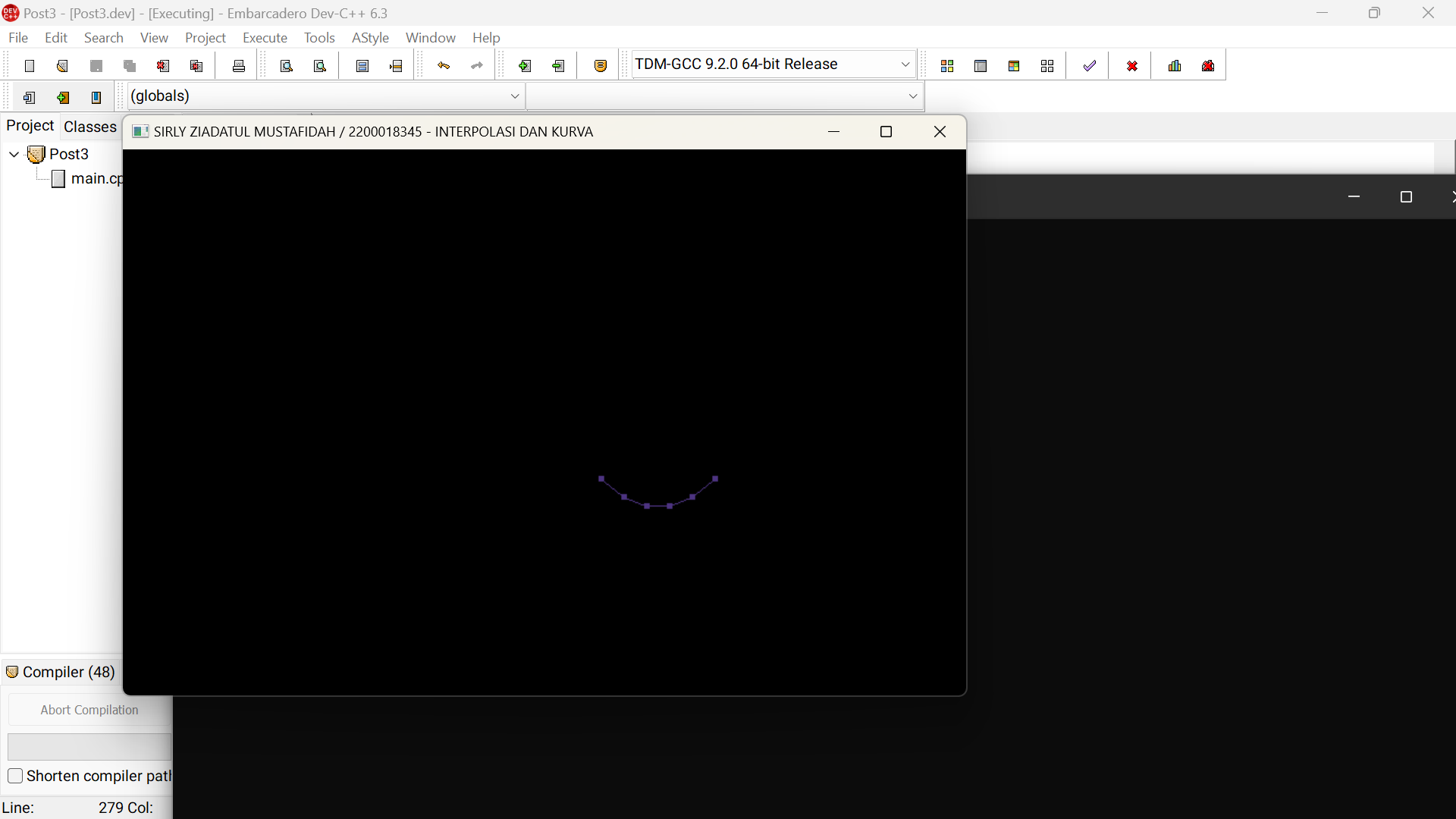
return 0;

}

Penjelasan



Output



Gambar 3. 2

# BAB IV. TRANSFORMASI 2D DAN 3D

1. Pretest 4

Diketahui suatu garis P dengan titik-titk ada di koordinat A(-1, -1) dan B(1, 1). Apabila garis P dikenakan transformasi berikut, Berapa koordinat titik-titik yang baru?

1. Garis P ditranslasi sejauh T(2,2)

Titik :

Maka

Titik :

Maka

Jadi, koordinat titik pada garis P setelah ditranslasi sejauh menjadi dan

1. Garis P discaling sebesar S(2,1)

Titik :

Maka

Titik :

Maka

Jadi, koordinat titik pada garis P setelah discaling sebesar menjadi dan

1. Garis P dirotasi sejauh 30 derajat

Titik :

Maka

Titik :

Maka

Jadi, koordinat titik pada garis P setelah ditranslasi sejauh menjadi dan

1. Langkah Praktikum 4

Source Code

// praktikum 4 transformasi 3D grafika komputer

// adhi prahara. 2018

// deklarasikan semua header disini

#include <windows.h>

#include <GL/glut.h>

// inisialisasi variabel untuk transformasi seperti translasi, rotasi atau scaling

float angle = 0.0f; // sudut transformasi kamera

float posX = 0.0f, rotX = 0.0f; // posisi kamera di sumbu X

float posY = 0.0f, rotY = 0.0f; // posisi kamera di sumbu Y

float posZ = 5.0f, rotZ = -1.0f; // posisi kamera di sumbu Z

float objectAngleX = 0.0f; // sudut tranformasi obyek di sumbu X

float objectAngleY = 0.0f; // sudut tranformasi obyek di sumbu Y

float objectAngleZ = 0.0f; // sudut tranformasi obyek di sumbu Z

float objectScaleX = 1.0f; // skala perbesaran obyek ke arah X

float objectScaleY = 1.0f; // skala perbesaran obyek ke arah Y

float objectScaleZ = 1.0f; // skala perbesaran obyek ke arah Z

float objectPositionX = 0.0f; // posisi obyek di sumbu X

float objectPositionY = 0.0f; // posisi obyek di sumbu Y

float objectPositionZ = 0.0f; // posisi obyek di sumbu Z

// fungsi untuk menggambar obyek

void drawObject()

{

// obyek bisa dimasukkan diantara glPushMatrix() dan glPopMatrix()

// fungsinya agar obyek tidak terpengaruh atau mempengaruhi obyek lain

// saat diwarnai, ditransformasi dan sebagainya

glPushMatrix();

// operasi transformasi translasi obyek

// ke arah sumbu X, Y atau Z

glTranslatef(objectPositionX, objectPositionY, objectPositionZ);

// operasi transformasi scaling obyek

// memperbesar atau mengecilkan obyek

// ke arah sumbu X, Y atau Z

glScalef(objectScaleX, objectScaleY, objectScaleZ);

// operasi transformasi rotasi obyek ke arah kanan-kiri

glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);

glPushMatrix();

// operasi transformasi rotasi obyek ke arah atas-bawah

glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

// set warna obyek ke warna hijau (0.0f, 1.0f, 0.0f)

glColor3f(0.2f, 0.3f, 0.5f);

// bila menggambar obyek harus diawali glBegin(tipe obyek) dan diakhiri dengan glEnd()

// kecuali menggunakan fungsi yang sudah ada di GLUT-OpenGL seperti dibawah ini

// glutSolidCube(1.0f); // menggambar obyek kubus

//Menggambar obyek segitiga

glBegin(GL\_TRIANGLES);

glVertex3f(0.0f, 1.0f, 0.0f);

glVertex3f(-1.0f, -1.0f, 0.0f);

glVertex3f(1.0f, -1.0f, 0.0f);

glEnd();

glPopMatrix();

glPopMatrix();

}

// taruh semua obyek yang akan digambar di fungsi display()

void display()

{

// bersihkan dan reset layar dan buffer

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

// posisikan kamera pandang

// dalam hal ini sumbu Y ada diatas dan posisi kamera pandang di (posX, posY, posZ)

gluLookAt(posX, posY, posZ, posX + rotX, posY + rotY, posZ + rotZ, 0.0f, 1.0f, 0.0f);

// panggil fungsi untuk menggambar obyek

drawObject();

// tampilkan obyek ke layar

// gunakan glFlush() bila memakai single buffer

// gunakan glutSwapBuffers() bila memakai double buffer

glutSwapBuffers();

}

// inisialisasikan variabel, pencahayaan, tekstur dan pengaturan kamera pandang di fungsi init()

void init(void)

{

// inisialisasi warna latar belakang layar dalam hal ini warna putih (1.0, 1.0, 1.0, 0.0)

glClearColor(1.0, 1.0, 1.0, 0.0);

glEnable(GL\_DEPTH\_TEST); // mengaktifkan depth buffer

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45.0, 1.0, 1.0, 100.0); // set proyeksi ke perspektif

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// inisialisasi kamera pandang

gluLookAt(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glEnable(GL\_COLOR\_MATERIAL);

}

// fungsi ini digunakan bila layar akan diresize (default)

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45, (GLfloat)w / (GLfloat)h, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

}

// fungsi untuk mengatur masukan dari keyboard

// untuk arah kiri, kanan, atas, bawah, PgUp, dan PgDn

void keyboard(int key, int x, int y)

{

float fraction = 0.1f;

switch (key)

{

// masukkan perintah disini bila tombol kiri ditekan

case GLUT\_KEY\_LEFT:

// dalam hal ini perintah rotasi obyek ke kiri sebanyak 1 derajat

objectAngleY -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol kanan ditekan

case GLUT\_KEY\_RIGHT:

// dalam hal ini perintah rotasi obyek ke kanan sebanyak 1 derajat

objectAngleY += 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol atas ditekan

case GLUT\_KEY\_UP:

// dalam hal ini perintah rotasi obyek ke atas sebanyak 1 derajat

objectAngleX -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol bawah ditekan

case GLUT\_KEY\_DOWN:

// dalam hal ini perintah rotasi obyek ke bawah sebanyak 1 derajat

objectAngleX += 1.0f;

glutPostRedisplay(); // update obyek

break;

// zoom in

case GLUT\_KEY\_PAGE\_UP:

// masukkan perintah disini bila tombol PgUp ditekan

posX += rotX \* fraction;

posZ += rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

// zoom out

case GLUT\_KEY\_PAGE\_DOWN:

// masukkan perintah disini bila tombol PgDn ditekan

posX -= rotX \* fraction;

posZ -= rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

}

}

// fungsi untuk mengatur masukan dari keyboard

void keyboard1(unsigned char key, int x, int y)

{

float fraction = 0.5f;

switch (key)

{

case 'w': // bila tombol 'w' pada keyboard ditekan

// translasi ke atas

objectPositionY += fraction;

glutPostRedisplay();

break;

case 's': // bila tombol 's' pada keyboard ditekan

// translasi ke bawah

objectPositionY -= fraction;

glutPostRedisplay();

break;

case 'a': // bila tombol 'a' pada keyboard ditekan

// translasi ke kiri

objectPositionX -= fraction;

glutPostRedisplay();

break;

case 'd': // bila tombol 'd' pada keyboard ditekan

// translasi ke kanan

objectPositionX += fraction;

glutPostRedisplay();

break;

case 'q': // bila tombol 'q' pada keyboard ditekan

// translasi ke depan

objectPositionZ += fraction;

glutPostRedisplay();

break;

case 'e': // bila tombol 'e' pada keyboard ditekan

// translasi ke belakang

objectPositionZ -= fraction;

glutPostRedisplay();

break;

case 't': // bila tombol 't' pada keyboard ditekan

// perbesar ke arah sumbu Y

objectScaleY += 0.1f;

glutPostRedisplay();

break;

case 'g': // bila tombol 'g' pada keyboard ditekan

// perkecil ke arah sumbu Y

objectScaleY = (objectScaleY - 0.1f, 1.0f);

glutPostRedisplay();

break;

case 'f': // bila tombol 'f' pada keyboard ditekan

// perbesar ke arah sumbu X

objectScaleX += 0.1f;

glutPostRedisplay();

break;

case 'h': // bila tombol 'h' pada keyboard ditekan

// perkecil ke arah sumbu X

objectScaleX = (objectScaleX - 0.1f, 1.0f);

glutPostRedisplay();

break;

case 'r': // bila tombol 'r' pada keyboard ditekan

// perbesar ke arah sumbu Z

objectScaleZ += 0.1f;

glutPostRedisplay();

break;

case 'y': // bila tombol 'y' pada keyboard ditekan

// perkecil ke arah sumbu Z

objectScaleZ = (objectScaleZ - 0.1f, 1.0f);

glutPostRedisplay();

break;

case 27: // bila tombol 'esc' pada keyboard ditekan

// keluar program

exit(0);

break;

}

}

// timer untuk animasi (gunakan bila perlu)

void timer(int value)

{

glutPostRedisplay();

glutTimerFunc(55, timer, 0);

}

// program utama

int main(int argc, char\*\* argv)

{

// inisialisasi jendela OpenGL

// GLUT\_SINGLE berarti memakai single buffer

// GLUT\_DOUBLE berarti memakai double buffer

// GLUT\_RGB berarti mode tampilan yang dipakai RGB

// GLUT\_RGBA berarti mode tampilan yang dipakai RGBA

// GLUT\_DEPTH berarti memakai depth buffer

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA | GLUT\_DEPTH);

// set ukuran jendela tampilan

glutInitWindowSize(480, 480); // besarnya jendela dalam piksel dalam hal ini 300x300

glutInitWindowPosition(100, 100); // posisi jendela dilayar komputer dalam piksel

// judul jendela (wajib diubah dengan informasi NAMA / NIM - JUDUL PRAKTIKUM masing-masing)

glutCreateWindow("SIRLY ZIADATUL MUSTAFIDAH / 2200018345 - TRANSFORMASI 2D DAN 3D");

// panggil fungsi init untuk inisialisasi awal

init();

// event handler untuk display, reshape dan keyboard

glutDisplayFunc(display); // display

glutReshapeFunc(reshape); // reshape

glutSpecialFunc(keyboard); // keyboard

glutKeyboardFunc(keyboard1); // keyboard

//glutTimerFunc(0, timer, 0); // aktifkan timer bila perlu

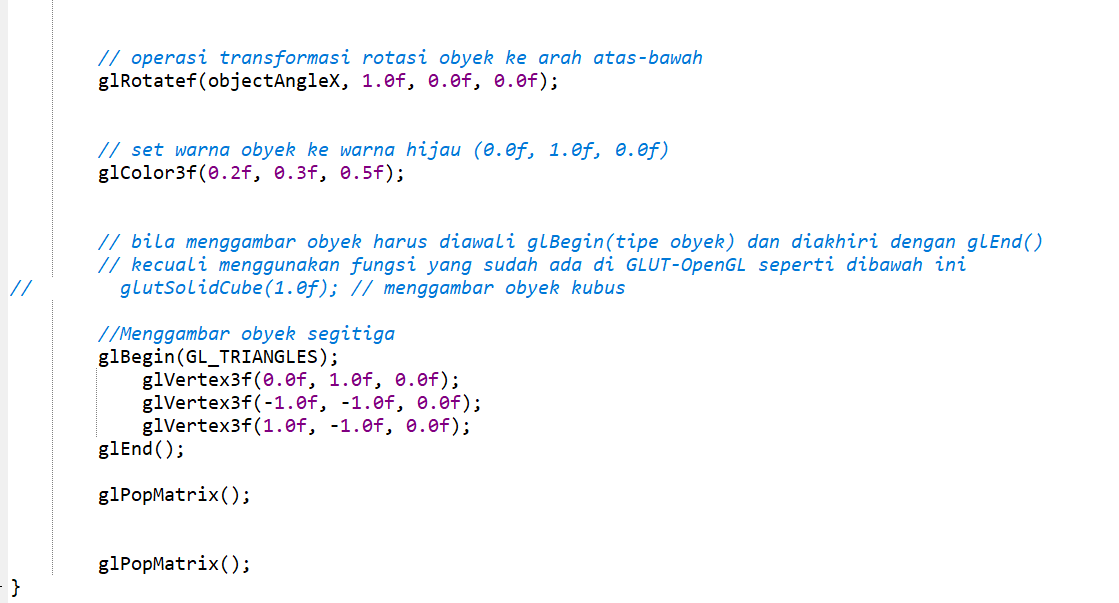
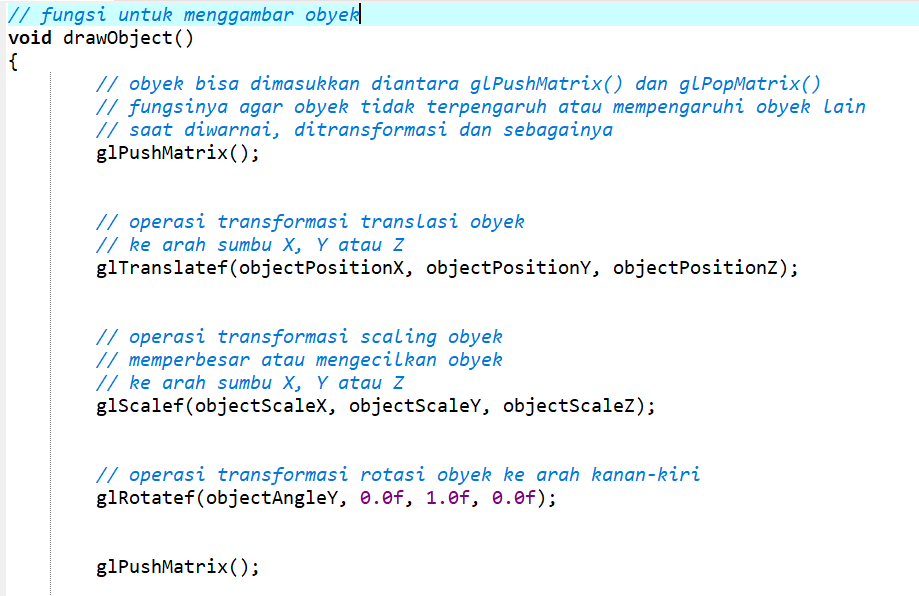
// looping

glutMainLoop();

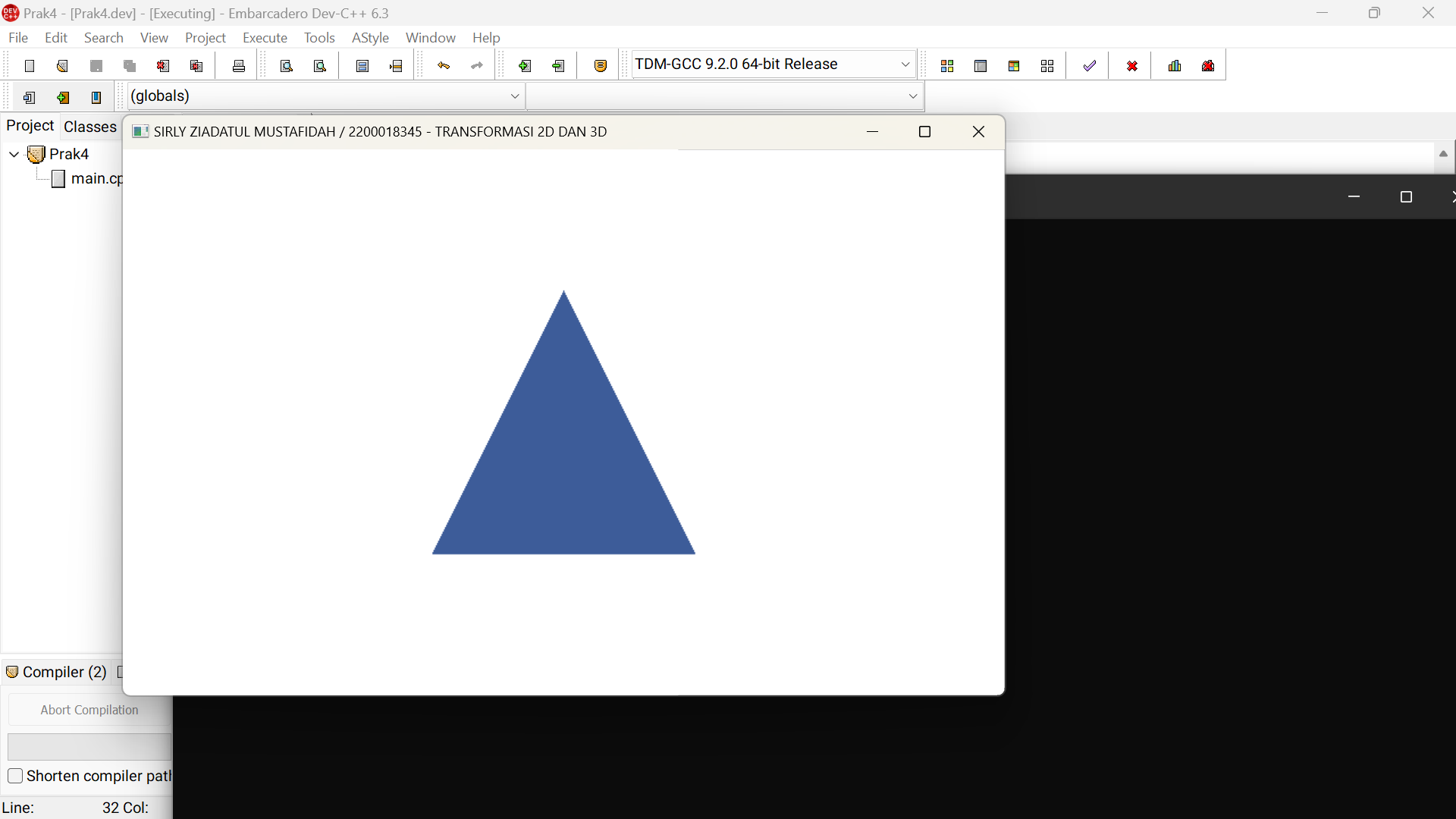
return 0;

}

Penjelasan



Output



Gambar 4. 1

1. Posttest 4

Source Code

// praktikum 4 transformasi 3D grafika komputer

// adhi prahara. 2018

// deklarasikan semua header disini

#include <windows.h>

#include <GL/glut.h>

// inisialisasi variabel untuk transformasi seperti translasi, rotasi atau scaling

float angle = 0.0f; // sudut transformasi kamera

float posX = 0.0f, rotX = 0.0f; // posisi kamera di sumbu X

float posY = 0.0f, rotY = 0.0f; // posisi kamera di sumbu Y

float posZ = 5.0f, rotZ = -1.0f; // posisi kamera di sumbu Z

float objectAngleX = 0.0f; // sudut tranformasi obyek di sumbu X

float objectAngleY = 0.0f; // sudut tranformasi obyek di sumbu Y

float objectAngleZ = 0.0f; // sudut tranformasi obyek di sumbu Z

float objectScaleX = 1.0f; // skala perbesaran obyek ke arah X

float objectScaleY = 1.0f; // skala perbesaran obyek ke arah Y

float objectScaleZ = 1.0f; // skala perbesaran obyek ke arah Z

float objectPositionX = 0.0f; // posisi obyek di sumbu X

float objectPositionY = 0.0f; // posisi obyek di sumbu Y

float objectPositionZ = 0.0f; // posisi obyek di sumbu Z

// fungsi untuk menggambar obyek

void drawObject()

{

// obyek bisa dimasukkan diantara glPushMatrix() dan glPopMatrix()

// fungsinya agar obyek tidak terpengaruh atau mempengaruhi obyek lain

// saat diwarnai, ditransformasi dan sebagainya

glPushMatrix();

// operasi transformasi translasi obyek

// ke arah sumbu X, Y atau Z

glTranslatef(objectPositionX, objectPositionY, objectPositionZ);

// operasi transformasi scaling obyek

// memperbesar atau mengecilkan obyek

// ke arah sumbu X, Y atau Z

glScalef(objectScaleX, objectScaleY, objectScaleZ);

// operasi transformasi rotasi obyek ke arah kanan-kiri

glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);

glPushMatrix();

// operasi transformasi rotasi obyek ke arah atas-bawah

glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

// bila menggambar obyek harus diawali glBegin(tipe obyek) dan diakhiri dengan glEnd()

// kecuali menggunakan fungsi yang sudah ada di GLUT-OpenGL seperti dibawah ini

// glutSolidCube(1.0f); // menggambar obyek kubus

//Menggambar obyek segitiga

glBegin(GL\_TRIANGLES);

//segmen pertama - warna biru

glColor3f(0.2f, 0.3f, 0.5f);

glVertex3f(0.0f, 1.0f, 0.0f);

glVertex3f(-0.5f, 0.0f, 0.0f);

glVertex3f(0.5f, 0.0f, 0.0f);

//segmen kedua - warna kuning

glColor3f(1.0f, 1.0f, 0.0f);

glVertex3f(0.0f, 1.0f, 0.0f);

glVertex3f(-0.5f, 0.0f, 0.0f);

glVertex3f(0.0f, -1.0f, 0.0f);

//segmen ketiga - warna merah

glColor3f(0.7f, 0.0f, 0.0f);

glVertex3f(0.0f, 1.0f, 0.0f);

glVertex3f(0.5f, 0.0f, 0.0f);

glVertex3f(0.0f, -1.0f, 0.0f);

glEnd();

glPopMatrix();

glPopMatrix();

}

// taruh semua obyek yang akan digambar di fungsi display()

void display()

{

// bersihkan dan reset layar dan buffer

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

// posisikan kamera pandang

// dalam hal ini sumbu Y ada diatas dan posisi kamera pandang di (posX, posY, posZ)

gluLookAt(posX, posY, posZ, posX + rotX, posY + rotY, posZ + rotZ, 0.0f, 1.0f, 0.0f);

// panggil fungsi untuk menggambar obyek

drawObject();

// tampilkan obyek ke layar

// gunakan glFlush() bila memakai single buffer

// gunakan glutSwapBuffers() bila memakai double buffer

glutSwapBuffers();

}

// inisialisasikan variabel, pencahayaan, tekstur dan pengaturan kamera pandang di fungsi init()

void init(void)

{

// inisialisasi warna latar belakang layar dalam hal ini warna putih (1.0, 1.0, 1.0, 0.0)

glClearColor(1.0, 1.0, 1.0, 0.0);

glEnable(GL\_DEPTH\_TEST); // mengaktifkan depth buffer

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45.0, 1.0, 1.0, 100.0); // set proyeksi ke perspektif

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// inisialisasi kamera pandang

gluLookAt(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glEnable(GL\_COLOR\_MATERIAL);

}

// fungsi ini digunakan bila layar akan diresize (default)

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45, (GLfloat)w / (GLfloat)h, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

}

// fungsi untuk mengatur masukan dari keyboard

// untuk arah kiri, kanan, atas, bawah, PgUp, dan PgDn

void keyboard(int key, int x, int y)

{

float fraction = 0.1f;

switch (key)

{

// masukkan perintah disini bila tombol kiri ditekan

case GLUT\_KEY\_LEFT:

// dalam hal ini perintah rotasi obyek ke kiri sebanyak 1 derajat

objectAngleY -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol kanan ditekan

case GLUT\_KEY\_RIGHT:

// dalam hal ini perintah rotasi obyek ke kanan sebanyak 1 derajat

objectAngleY += 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol atas ditekan

case GLUT\_KEY\_UP:

// dalam hal ini perintah rotasi obyek ke atas sebanyak 1 derajat

objectAngleX -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol bawah ditekan

case GLUT\_KEY\_DOWN:

// dalam hal ini perintah rotasi obyek ke bawah sebanyak 1 derajat

objectAngleX += 1.0f;

glutPostRedisplay(); // update obyek

break;

// zoom in

case GLUT\_KEY\_PAGE\_UP:

// masukkan perintah disini bila tombol PgUp ditekan

posX += rotX \* fraction;

posZ += rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

// zoom out

case GLUT\_KEY\_PAGE\_DOWN:

// masukkan perintah disini bila tombol PgDn ditekan

posX -= rotX \* fraction;

posZ -= rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

}

}

// fungsi untuk mengatur masukan dari keyboard

void keyboard1(unsigned char key, int x, int y)

{

float fraction = 0.5f;

switch (key)

{

case 'w': // bila tombol 'w' pada keyboard ditekan

// translasi ke atas

objectPositionY += fraction;

glutPostRedisplay();

break;

case 's': // bila tombol 's' pada keyboard ditekan

// translasi ke bawah

objectPositionY -= fraction;

glutPostRedisplay();

break;

case 'a': // bila tombol 'a' pada keyboard ditekan

// translasi ke kiri

objectPositionX -= fraction;

glutPostRedisplay();

break;

case 'd': // bila tombol 'd' pada keyboard ditekan

// translasi ke kanan

objectPositionX += fraction;

glutPostRedisplay();

break;

case 'q': // bila tombol 'q' pada keyboard ditekan

// translasi ke depan

objectPositionZ += fraction;

glutPostRedisplay();

break;

case 'e': // bila tombol 'e' pada keyboard ditekan

// translasi ke belakang

objectPositionZ -= fraction;

glutPostRedisplay();

break;

case 't': // bila tombol 't' pada keyboard ditekan

// perbesar ke arah sumbu Y

objectScaleY += 0.1f;

glutPostRedisplay();

break;

case 'g': // bila tombol 'g' pada keyboard ditekan

// perkecil ke arah sumbu Y

objectScaleY = (objectScaleY - 0.1f, 1.0f);

glutPostRedisplay();

break;

case 'f': // bila tombol 'f' pada keyboard ditekan

// perbesar ke arah sumbu X

objectScaleX += 0.1f;

glutPostRedisplay();

break;

case 'h': // bila tombol 'h' pada keyboard ditekan

// perkecil ke arah sumbu X

objectScaleX = (objectScaleX - 0.1f, 1.0f);

glutPostRedisplay();

break;

case 'r': // bila tombol 'r' pada keyboard ditekan

// perbesar ke arah sumbu Z

objectScaleZ += 0.1f;

glutPostRedisplay();

break;

case 'y': // bila tombol 'y' pada keyboard ditekan

// perkecil ke arah sumbu Z

objectScaleZ = (objectScaleZ - 0.1f, 1.0f);

glutPostRedisplay();

break;

case 27: // bila tombol 'esc' pada keyboard ditekan

// keluar program

exit(0);

break;

}

}

// timer untuk animasi (gunakan bila perlu)

void timer(int value)

{

glutPostRedisplay();

glutTimerFunc(55, timer, 0);

}

// program utama

int main(int argc, char\*\* argv)

{

// inisialisasi jendela OpenGL

// GLUT\_SINGLE berarti memakai single buffer

// GLUT\_DOUBLE berarti memakai double buffer

// GLUT\_RGB berarti mode tampilan yang dipakai RGB

// GLUT\_RGBA berarti mode tampilan yang dipakai RGBA

// GLUT\_DEPTH berarti memakai depth buffer

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA | GLUT\_DEPTH);

// set ukuran jendela tampilan

glutInitWindowSize(480, 480); // besarnya jendela dalam piksel dalam hal ini 300x300

glutInitWindowPosition(100, 100); // posisi jendela dilayar komputer dalam piksel

// judul jendela (wajib diubah dengan informasi NAMA / NIM - JUDUL PRAKTIKUM masing-masing)

glutCreateWindow("SIRLY ZIADATUL MUSTAFIDAH / 2200018345 - TRANSFORMASI 2D DAN 3D");

// panggil fungsi init untuk inisialisasi awal

init();

// event handler untuk display, reshape dan keyboard

glutDisplayFunc(display); // display

glutReshapeFunc(reshape); // reshape

glutSpecialFunc(keyboard); // keyboard

glutKeyboardFunc(keyboard1); // keyboard

//glutTimerFunc(0, timer, 0); // aktifkan timer bila perlu

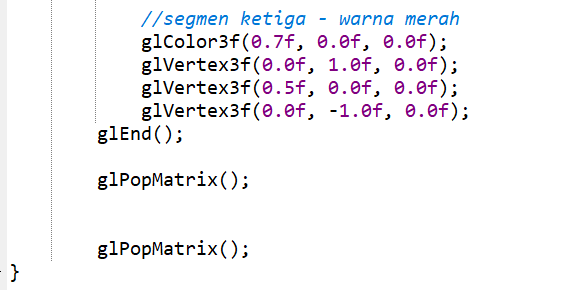
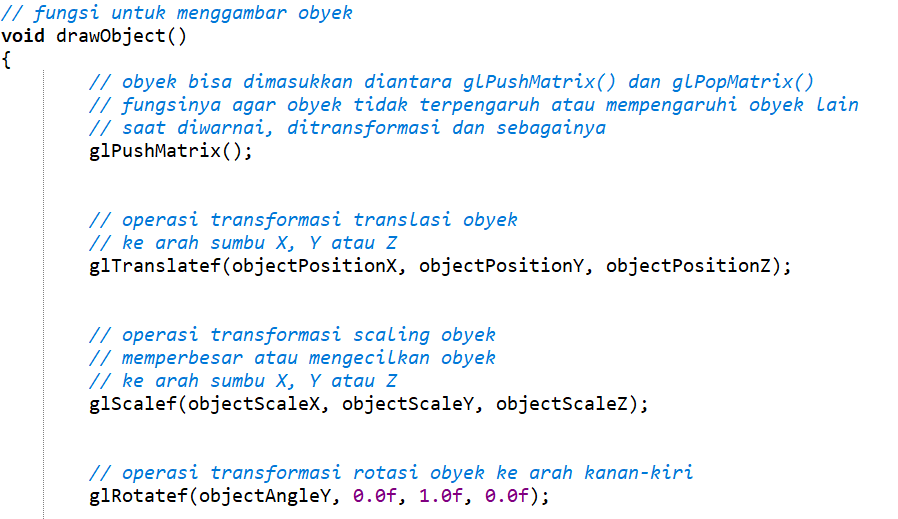
// looping

glutMainLoop();

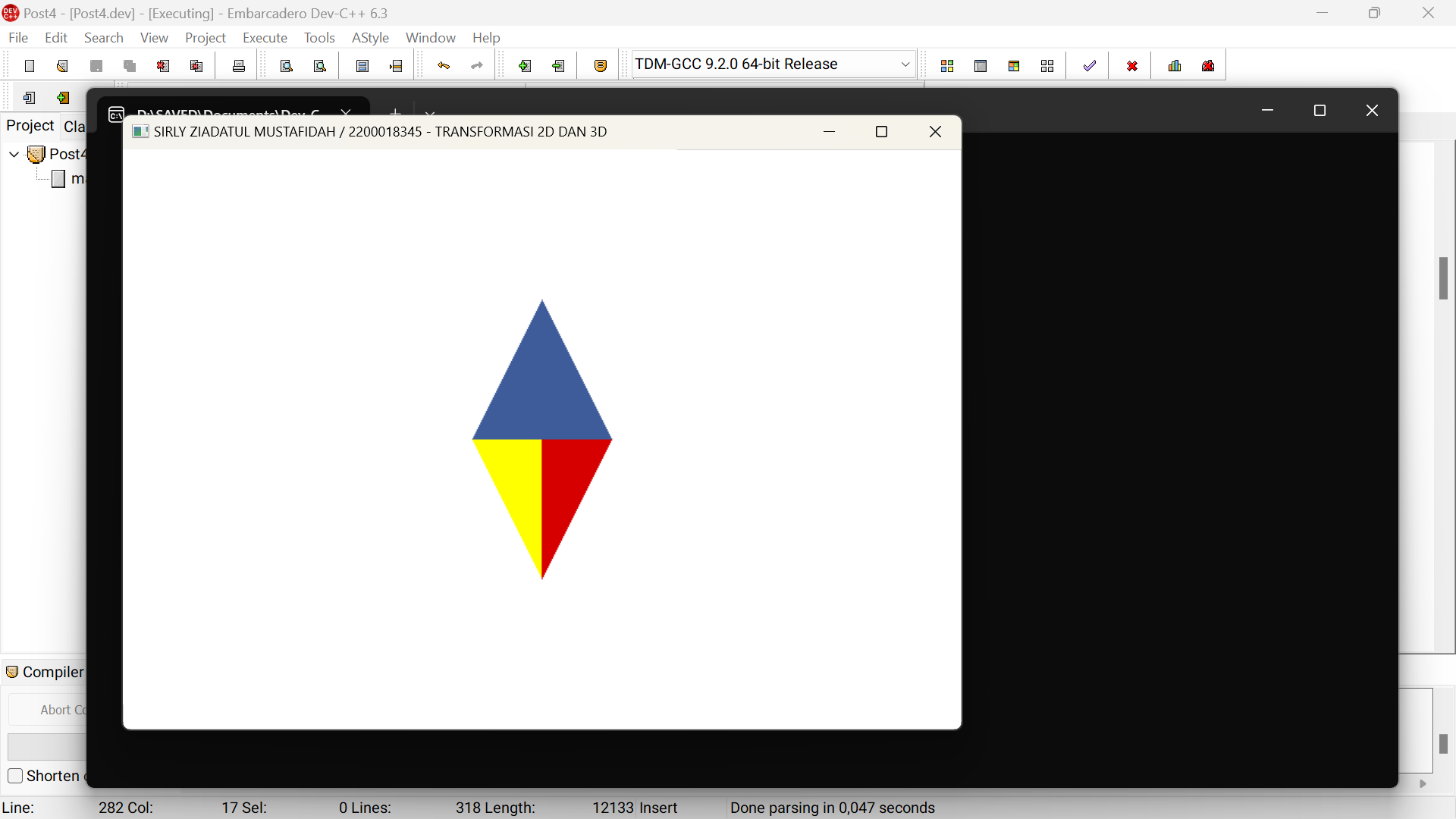
return 0;

}

Penjelasan



Output



Gambar 4. 2

# BAB V. PROYEKSI 3D

1. Pretest 5
2. Jelaskan yang dimaksud dengan proyeksi orthogonal!

Proyeksi yang bidang proyeksinya mempunyai sudut tegak lurus terhadap proyektornya. Proyeksi orthogonal digunakan dari depan, samping, dan atas

1. Jelaskan yang dimaksud dengan proyeksi perspektif!

Proyeksi bila posisi koordinat obyek ditransformasikan ke bidang pandang dengan garis-garis yang memusat di sebuah titik yang disebut titik referensi proyeksi atau Center of Projektion (COP)

1. Jelaskan perbedaan antara proyeksi orthogonal dengan perspektif!

Perbedaan antara proyeksi orthogonal dengan perspektif, sebagai berikut:

* Proyeksi perspektid obyeknya semakin mengecil ketika semakin menjauh, sedangkan proyeksi orthogonal dilihat dari berbagai arah
* Proyeksi perspektif itu proyeksinya dari suatu titik, sedangkan proyeksi orthogonal itu setara dengan perspektif dari jarak yang sangat jauh

1. Langkah Praktikum 5

Source Code

// kode dasar untuk semua praktikum grafika komputer

// adhi prahara. 2018

// deklarasikan semua header disini

#include <windows.h>

#include <GL/glut.h>

// posisi sumber cahaya

// posisi sumber cahaya utk perspektif

float position[] = {0.0f,5.0f,5.0f,1.0f};

// posisi sumber cahaya utk orthogonal

//float position[] = {0.0f,100.0f,-100.0f,1.0f};

// inisialisasi variabel untuk transformasi seperti translasi, rotasi atau scaling

float angle = 0.0f; // sudut transformasi kamera

float posX = 0.0f, rotX = 0.0f; // posisi kamera di sumbu X

float posY = 0.0f, rotY = 0.0f; // posisi kamera di sumbu Y

float posZ = 5.0f, rotZ = -1.0f; // posisi kamera di sumbu Z

float objectAngleX = 0.0f; // sudut tranformasi obyek di sumbu X

float objectAngleY = 0.0f; // sudut tranformasi obyek di sumbu Y

float objectAngleZ = 0.0f; // sudut tranformasi obyek di sumbu Z

// fungsi untuk menggambar obyek kubus

void drawObject()

{

// obyek bisa dimasukkan diantara glPushMatrix() dan glPopMatrix()

// fungsinya agar obyek tidak terpengaruh atau mempengaruhi obyek lain

// saat diwarnai, ditransformasi dan sebagainya

glPushMatrix();

// operasi transformasi rotasi obyek ke arah kanan-kiri

glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);

glPushMatrix();

// operasi transformasi rotasi obyek ke arah atas-bawah

glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

// set warna obyek ke warna hijau (0.0f, 1.0f, 0.0f)

glColor3f(0.4f, 0.2f, 0.8f);

// bila menggambar obyek harus diawali glBegin(tipe obyek) dan diakhiri dengan glEnd()

// kecuali menggunakan fungsi yang sudah ada di GLUT-OpenGL seperti dibawah ini

// glutSolidCube(1.0f); // menggambar obyek kubus

glutSolidSphere(1, 20, 10); //menggambar obyek bola

glPopMatrix();

glPopMatrix();

}

// taruh semua obyek yang akan digambar di fungsi display()

void display()

{

// bersihkan dan reset layar dan buffer

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

// posisikan kamera pandang

// dalam hal ini sumbu Y ada diatas dan posisi kamera pandang di (posX, posY, posZ)

gluLookAt(posX, posY, posZ, posX + rotX, posY + rotY, posZ + rotZ, 0.0f, 1.0f, 0.0f);

// panggil fungsi untuk menggambar obyek

drawObject();

// tampilkan obyek ke layar

// gunakan glFlush() bila memakai single buffer

// gunakan glutSwapBuffers() bila memakai double buffer

glutSwapBuffers();

}

// inisialisasikan variabel, pencahayaan, tekstur dan pengaturan kamera pandang di fungsi init()

void init(void)

{

// inisialisasi warna latar belakang layar dalam hal ini warna putih (1.0, 1.0, 1.0, 0.0)

glClearColor(1.0, 1.0, 1.0, 0.0);

glEnable(GL\_DEPTH\_TEST); // mengaktifkan depth buffer

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45.0, 1.0, 1.0, 100.0); // set proyeksi ke perspektif

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// inisialisasi kamera pandang

gluLookAt(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

// aktifkan pencahayaan

glEnable(GL\_LIGHTING);

glEnable(GL\_COLOR\_MATERIAL);

glEnable(GL\_LIGHT0);

}

// fungsi ini digunakan bila layar akan diresize (default)

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45, (GLfloat)w / (GLfloat)h, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

}

// fungsi untuk mengatur masukan dari keyboard

// untuk arah kiri, kanan, atas, bawah, PgUp, dan PgDn

void keyboard(int key, int x, int y)

{

float fraction = 0.1f;

switch (key)

{

// masukkan perintah disini bila tombol kiri ditekan

case GLUT\_KEY\_LEFT:

// dalam hal ini perintah rotasi obyek ke kiri sebanyak 1 derajat

objectAngleY -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol kanan ditekan

case GLUT\_KEY\_RIGHT:

// dalam hal ini perintah rotasi obyek ke kanan sebanyak 1 derajat

objectAngleY += 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol atas ditekan

case GLUT\_KEY\_UP:

// dalam hal ini perintah rotasi obyek ke atas sebanyak 1 derajat

objectAngleX -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol bawah ditekan

case GLUT\_KEY\_DOWN:

// dalam hal ini perintah rotasi obyek ke bawah sebanyak 1 derajat

objectAngleX += 1.0f;

glutPostRedisplay(); // update obyek

break;

// zoom in

case GLUT\_KEY\_PAGE\_UP:

// masukkan perintah disini bila tombol PgUp ditekan

posX += rotX \* fraction;

posZ += rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

// zoom out

case GLUT\_KEY\_PAGE\_DOWN:

// masukkan perintah disini bila tombol PgDn ditekan

posX -= rotX \* fraction;

posZ -= rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

}

}

// timer untuk animasi (gunakan bila perlu)

void timer(int value)

{

glutPostRedisplay();

glutTimerFunc(55, timer, 0);

}

// program utama

int main(int argc, char\*\* argv)

{

// inisialisasi jendela OpenGL

// GLUT\_SINGLE berarti memakai single buffer

// GLUT\_DOUBLE berarti memakai double buffer

// GLUT\_RGB berarti mode tampilan yang dipakai RGB

// GLUT\_RGBA berarti mode tampilan yang dipakai RGBA

// GLUT\_DEPTH berarti memakai depth buffer

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA | GLUT\_DEPTH);

// set ukuran jendela tampilan

glutInitWindowSize(480, 480); // besarnya jendela dalam piksel dalam hal ini 300x300

glutInitWindowPosition(100, 100); // posisi jendela dilayar komputer dalam piksel

// judul jendela (wajib diubah dengan informasi NAMA / NIM - JUDUL PRAKTIKUM masing-masing)

glutCreateWindow("SIRLY ZIADATUL MUSTAFIDAH / 2200018345 - PROYEKSI 3D");

// panggil fungsi init untuk inisialisasi awal

init();

// event handler untuk display, reshape dan keyboard

glutDisplayFunc(display); // display

glutReshapeFunc(reshape); // reshape

glutSpecialFunc(keyboard); // keyboard

//glutTimerFunc(0, timer, 0); // aktifkan timer bila perlu

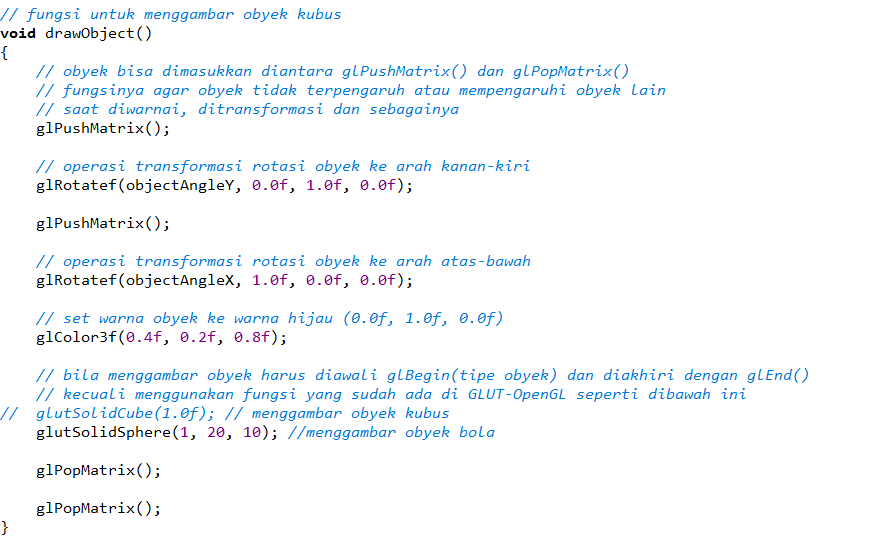
// looping

glutMainLoop();

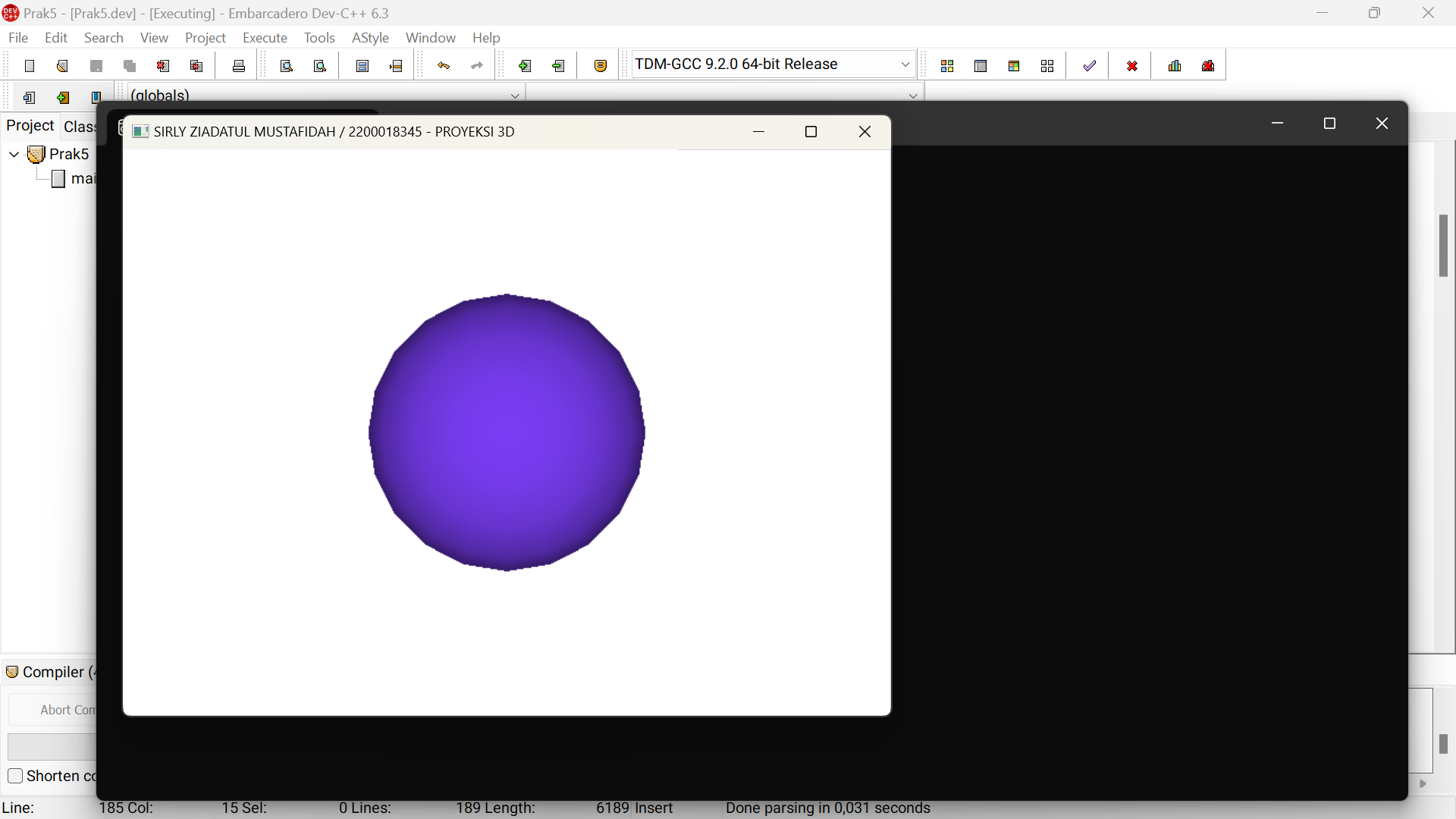
return 0;

}

Penjelasan



Output



Gambar 5. 1

1. Postest 5

Source Code

// kode dasar untuk semua praktikum grafika komputer

// adhi prahara. 2018

// deklarasikan semua header disini

#include <windows.h>

#include <GL/glut.h>

// posisi sumber cahaya

// posisi sumber cahaya utk perspektif

float position[] = {0.0f,5.0f,5.0f,1.0f};

// posisi sumber cahaya utk orthogonal

//float position[] = {0.0f,100.0f,-100.0f,1.0f};

// inisialisasi variabel untuk transformasi seperti translasi, rotasi atau scaling

float angle = 0.0f; // sudut transformasi kamera

float posX = 0.0f, rotX = 0.0f; // posisi kamera di sumbu X

float posY = 0.0f, rotY = 0.0f; // posisi kamera di sumbu Y

float posZ = 5.0f, rotZ = -1.0f; // posisi kamera di sumbu Z

float objectAngleX = 0.0f; // sudut tranformasi obyek di sumbu X

float objectAngleY = 0.0f; // sudut tranformasi obyek di sumbu Y

float objectAngleZ = 0.0f; // sudut tranformasi obyek di sumbu Z

// fungsi untuk menggambar obyek kubus

void drawObject()

{

// obyek bisa dimasukkan diantara glPushMatrix() dan glPopMatrix()

// fungsinya agar obyek tidak terpengaruh atau mempengaruhi obyek lain

// saat diwarnai, ditransformasi dan sebagainya

glPushMatrix();

// operasi transformasi rotasi obyek ke arah kanan-kiri

glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);

glPushMatrix();

// operasi transformasi rotasi obyek ke arah atas-bawah

glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

// set warna obyek ke warna hijau (0.0f, 1.0f, 0.0f)

glColor3f(0.4f, 0.2f, 0.8f);

// bila menggambar obyek harus diawali glBegin(tipe obyek) dan diakhiri dengan glEnd()

// kecuali menggunakan fungsi yang sudah ada di GLUT-OpenGL seperti dibawah ini

// glutSolidCube(1.0f); // menggambar obyek kubus

glutSolidCone(1,2,25,25); //menggambar obyek

glPopMatrix();

glPopMatrix();

}

// taruh semua obyek yang akan digambar di fungsi display()

void display()

{

// bersihkan dan reset layar dan buffer

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

// posisikan kamera pandang

// dalam hal ini sumbu Y ada diatas dan posisi kamera pandang di (posX, posY, posZ)

gluLookAt(posX, posY, posZ, posX + rotX, posY + rotY, posZ + rotZ, 0.0f, 1.0f, 0.0f);

// panggil fungsi untuk menggambar obyek

drawObject();

// tampilkan obyek ke layar

// gunakan glFlush() bila memakai single buffer

// gunakan glutSwapBuffers() bila memakai double buffer

glutSwapBuffers();

}

// inisialisasikan variabel, pencahayaan, tekstur dan pengaturan kamera pandang di fungsi init()

void init(void)

{

// inisialisasi warna latar belakang layar dalam hal ini warna putih (1.0, 1.0, 1.0, 0.0)

glClearColor(1.0, 1.0, 1.0, 0.0);

glEnable(GL\_DEPTH\_TEST); // mengaktifkan depth buffer

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45.0, 1.0, 1.0, 100.0); // set proyeksi ke perspektif

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// inisialisasi kamera pandang

gluLookAt(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

// aktifkan pencahayaan

glEnable(GL\_LIGHTING);

glEnable(GL\_COLOR\_MATERIAL);

glEnable(GL\_LIGHT0);

}

// fungsi ini digunakan bila layar akan diresize (default)

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45, (GLfloat)w / (GLfloat)h, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

}

// fungsi untuk mengatur masukan dari keyboard

// untuk arah kiri, kanan, atas, bawah, PgUp, dan PgDn

void keyboard(int key, int x, int y)

{

float fraction = 0.1f;

switch (key)

{

// masukkan perintah disini bila tombol kiri ditekan

case GLUT\_KEY\_LEFT:

// dalam hal ini perintah rotasi obyek ke kiri sebanyak 1 derajat

objectAngleY -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol kanan ditekan

case GLUT\_KEY\_RIGHT:

// dalam hal ini perintah rotasi obyek ke kanan sebanyak 1 derajat

objectAngleY += 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol atas ditekan

case GLUT\_KEY\_UP:

// dalam hal ini perintah rotasi obyek ke atas sebanyak 1 derajat

objectAngleX -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol bawah ditekan

case GLUT\_KEY\_DOWN:

// dalam hal ini perintah rotasi obyek ke bawah sebanyak 1 derajat

objectAngleX += 1.0f;

glutPostRedisplay(); // update obyek

break;

// zoom in

case GLUT\_KEY\_PAGE\_UP:

// masukkan perintah disini bila tombol PgUp ditekan

posX += rotX \* fraction;

posZ += rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

// zoom out

case GLUT\_KEY\_PAGE\_DOWN:

// masukkan perintah disini bila tombol PgDn ditekan

posX -= rotX \* fraction;

posZ -= rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

}

}

// timer untuk animasi (gunakan bila perlu)

void timer(int value)

{

glutPostRedisplay();

glutTimerFunc(55, timer, 0);

}

// program utama

int main(int argc, char\*\* argv)

{

// inisialisasi jendela OpenGL

// GLUT\_SINGLE berarti memakai single buffer

// GLUT\_DOUBLE berarti memakai double buffer

// GLUT\_RGB berarti mode tampilan yang dipakai RGB

// GLUT\_RGBA berarti mode tampilan yang dipakai RGBA

// GLUT\_DEPTH berarti memakai depth buffer

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA | GLUT\_DEPTH);

// set ukuran jendela tampilan

glutInitWindowSize(480, 480); // besarnya jendela dalam piksel dalam hal ini 300x300

glutInitWindowPosition(100, 100); // posisi jendela dilayar komputer dalam piksel

// judul jendela (wajib diubah dengan informasi NAMA / NIM - JUDUL PRAKTIKUM masing-masing)

glutCreateWindow("SIRLY ZIADATUL MUSTAFIDAH / 2200018345 - PROYEKSI 3D");

// panggil fungsi init untuk inisialisasi awal

init();

// event handler untuk display, reshape dan keyboard

glutDisplayFunc(display); // display

glutReshapeFunc(reshape); // reshape

glutSpecialFunc(keyboard); // keyboard

//glutTimerFunc(0, timer, 0); // aktifkan timer bila perlu

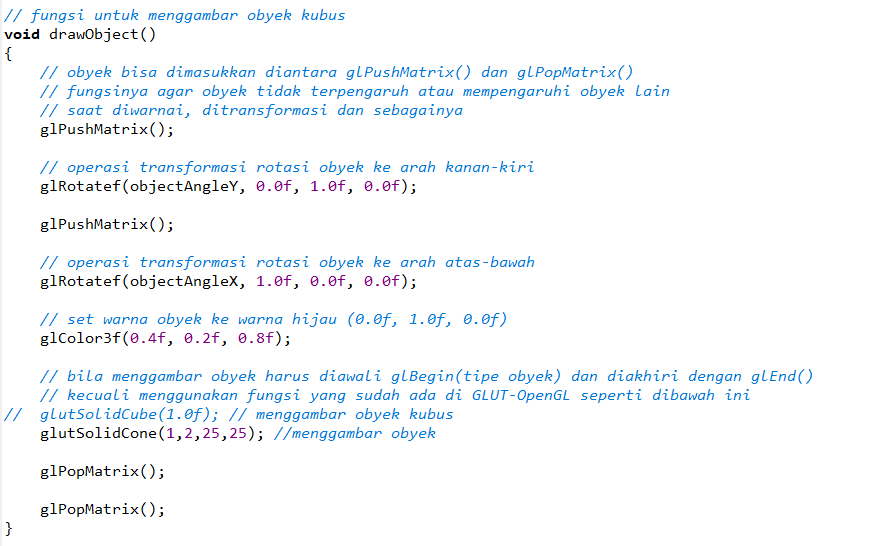
// looping

glutMainLoop();

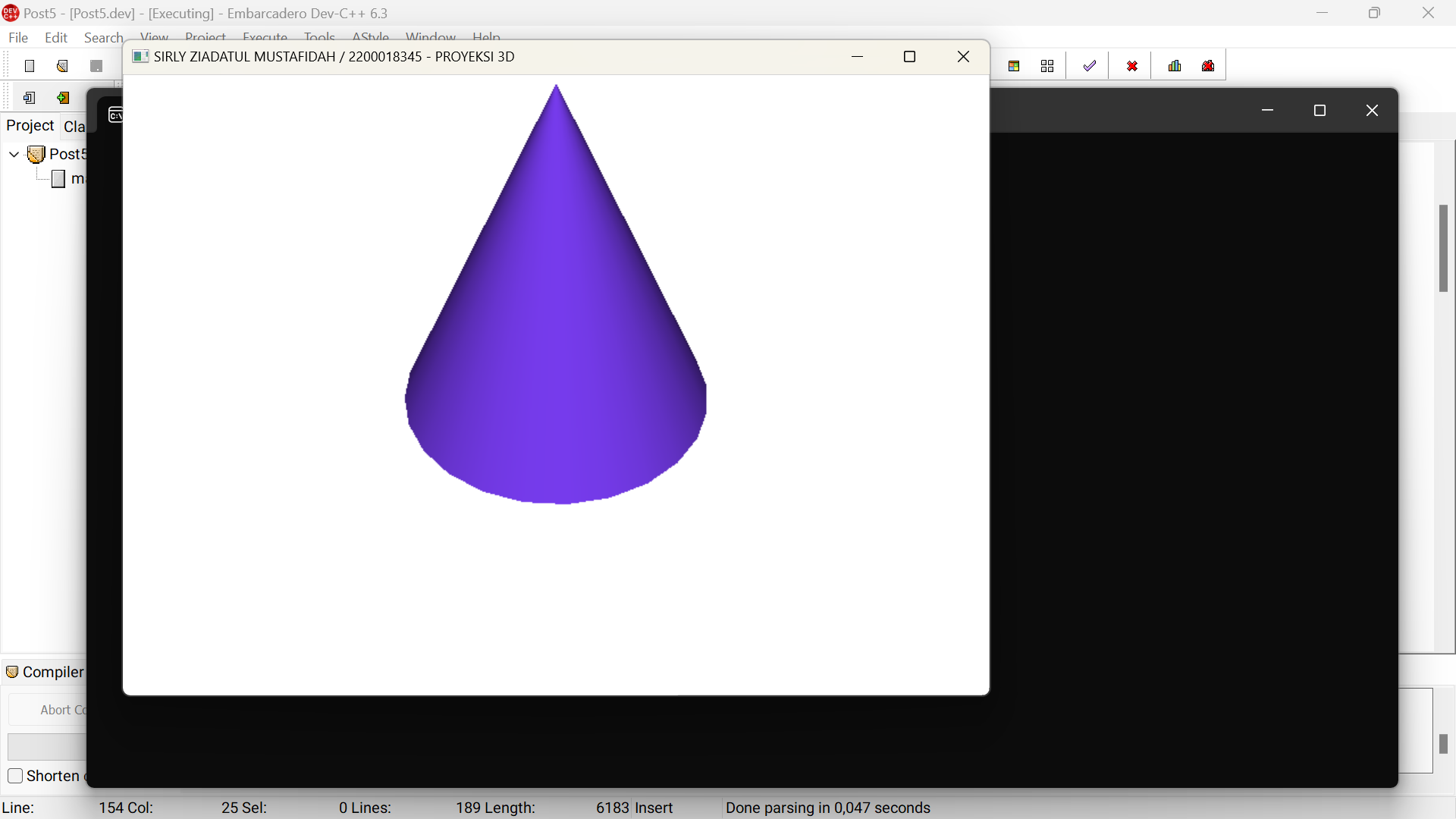
return 0;

}

Penjelasan



Output



Gambar 5. 2

# BAB VI. REPRESENTASI OBYEK 3D

1. Pretest 6
2. Sebutkan metode representasi obyek 3D yang anda ketahui!

* Wireframe
* Sweep representation
* Boundary representation
* Spatial paritional representation
* Constructive Solid Geometry (CSG)

1. Jelaskan setiap metode representasi obyek 3D yang anda sebutkan di soal nomor 1!

* Model wireframe adalah metode yang menggunakan kumpulan garis yang mewakili tepian obyek dan struktur obyek
* Sweep representation adalah metode ini mendeskripsikan obyek 3D dari batasan permukaannya yang membedakan obyek bagian dalam dan bagian luar
* Spantial partitioning representation adalah metode yang membagi ruang ke dalam kumpulan bagian yang lebih kecil
* CSG adalah metode yang mengombinasikan beberapa bentuk obyek dengan beberapa operasi

1. Langkah Praktikum 6

Source Code

// kode dasar untuk semua praktikum grafika komputer

// adhi prahara. 2018

// deklarasikan semua header disini

#include <windows.h>

#include <GL/glut.h>

// inisialisasi variabel untuk transformasi seperti translasi, rotasi atau scaling

float angle = 0.0f; // sudut transformasi kamera

float posX = 0.0f, rotX = 0.0f; // posisi kamera di sumbu X

float posY = 0.0f, rotY = 0.0f; // posisi kamera di sumbu Y

float posZ = 5.0f, rotZ = -1.0f; // posisi kamera di sumbu Z

float objectAngleX = 0.0f; // sudut tranformasi obyek di sumbu X

float objectAngleY = 0.0f; // sudut tranformasi obyek di sumbu Y

float objectAngleZ = 0.0f; // sudut tranformasi obyek di sumbu Z

// container untuk membuat tipe data 3D (X, Y, Z)

struct Vec3

{

float X; float Y; float Z;

Vec3(float x, float y, float z) { X = x; Y = y; Z = z; }

//

Vec3() { }

~Vec3() { }

};

void drawCube()

{

glBegin(GL\_QUADS);

// beri warna merah di sisi depan

glColor3f(1.0f, 0.0f, 0.0f);

// buat sisi depan

glVertex3f(-1.0f, -1.0f, 1.0f);

glVertex3f( 1.0f, -1.0f, 1.0f);

glVertex3f( 1.0f, 1.0f, 1.0f);

glVertex3f(-1.0f, 1.0f, 1.0f);

// beri warna hijau di sisi belakang

glColor3f(0.0f, 1.0f, 0.0f);

// buat sisi belakang

glVertex3f(-1.0f, -1.0f,-1.0f);

glVertex3f( 1.0f, -1.0f,-1.0f);

glVertex3f( 1.0f, 1.0f,-1.0f);

glVertex3f(-1.0f, 1.0f,-1.0f);

// beri warna biru di sisi kiri

glColor3f(0.0f, 0.0f, 1.0f);

// buat sisi kiri

glVertex3f(-1.0f, -1.0f, 1.0f);

glVertex3f(-1.0f, -1.0f,-1.0f);

glVertex3f(-1.0f, 1.0f,-1.0f);

glVertex3f(-1.0f, 1.0f, 1.0f);

// beri warna cyan di sisi kanan

glColor3f(0.0f, 1.0f, 1.0f);

// buat sisi kanan

glVertex3f( 1.0f, -1.0f, 1.0f);

glVertex3f( 1.0f, -1.0f,-1.0f);

glVertex3f( 1.0f, 1.0f,-1.0f);

glVertex3f( 1.0f, 1.0f, 1.0f);

// beri warna kuning di sisi atas

glColor3f(1.0f, 1.0f, 0.0f);

// buat sisi atas

glVertex3f(-1.0f, 1.0f, 1.0f);

glVertex3f( 1.0f, 1.0f, 1.0f);

glVertex3f( 1.0f, 1.0f,-1.0f);

glVertex3f(-1.0f, 1.0f,-1.0f);

// beri warna magenta di sisi bawah

glColor3f(1.0f, 0.0f, 1.0f);

// buat sisi bawah

glVertex3f(-1.0f, -1.0f, 1.0f);

glVertex3f( 1.0f, -1.0f, 1.0f);

glVertex3f( 1.0f, -1.0f,-1.0f);

glVertex3f(-1.0f, -1.0f,-1.0f);

glEnd();

}

// fungsi untuk menggambar silinder

void drawCylinder(float radius, float height, int slices, int stacks)

{

glPushMatrix();

GLUquadricObj\* cyl = gluNewQuadric();

gluQuadricDrawStyle(cyl, GLU\_FILL);

gluQuadricNormals(cyl, GLU\_SMOOTH);

gluQuadricOrientation(cyl, GLU\_INSIDE);

// buat tutup atas silinder

glTranslatef(0.0f, -height/2, 0.0f);

glRotatef(-90, 1.0f, 0.0f, 0.0f);

glColor3f(1.0f, 1.0f, 0.0f); // warna kuning

gluDisk(cyl, 0.0f, radius, slices, stacks);

// buat badan silinder

glColor3f(1.0f, 0.0f, 0.0f); // warna merah

gluCylinder(cyl, radius, radius, height, slices, stacks);

// buat tutup bawah silinder

glColor3f(1.0f, 1.0f, 0.0f); // warna kuning

glTranslatef(0.0f, 0.0f, height);

gluDisk(cyl, 0.0f, radius, slices, stacks);

glPopMatrix();

}

// fungsi untuk menggambar bola

void drawSphere(float radius, int slices, int stacks)

{

glPushMatrix();

glColor3f(1.0f, 0.0f, 0.0f); // warna merah

GLUquadric \*sphere = gluNewQuadric();

gluQuadricDrawStyle(sphere, GLU\_FILL);

gluQuadricNormals(sphere, GLU\_SMOOTH);

gluSphere(sphere, radius, slices, stacks);

glPopMatrix();

}

void drawTriangles()

{

glBegin(GL\_TRIANGLES);

// Depan

glColor3f(1.0f, 0.0f, 0.0f);

glVertex3f( 0.0f, 1.0f, 0.0f);

glVertex3f(-1.0f, -1.0f, 1.0f);

glVertex3f(1.0f, -1.0f, 1.0f);

// Kanan

glColor3f(0.0f, 0.0f, 1.0f);

glVertex3f(0.0f, 1.0f, 0.0f);

glVertex3f(1.0f, -1.0f, 1.0f);

glVertex3f(1.0f, -1.0f, -1.0f);

// Belakang

glColor3f(0.0f, 1.0f, 0.0f);

glVertex3f(0.0f, 1.0f, 0.0f);

glVertex3f(1.0f, -1.0f, -1.0f);

glVertex3f(-1.0f, -1.0f, -1.0f);

// Kiri

glColor3f(1.0f, 1.0f,0.0f);

glVertex3f( 0.0f, 1.0f, 0.0f);

glVertex3f(-1.0f,-1.0f,-1.0f);

glVertex3f(-1.0f,-1.0f, 1.0f);

glEnd();

}

// fungsi untuk menggambar obyek

void drawObject()

{

// obyek bisa dimasukkan diantara glPushMatrix() dan glPopMatrix()

// fungsinya agar obyek tidak terpengaruh atau mempengaruhi obyek lain

// saat diwarnai, ditransformasi dan sebagainya

glPushMatrix();

// operasi transformasi rotasi obyek ke arah kanan-kiri

glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);

glPushMatrix();

// operasi transformasi rotasi obyek ke arah atas-bawah

glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

// set warna obyek ke warna hijau (0.0f, 1.0f, 0.0f)

glColor3f(0.0f, 1.0f, 0.0f);

// bila menggambar obyek harus diawali glBegin(tipe obyek) dan diakhiri dengan glEnd()

// kecuali menggunakan fungsi yang sudah ada di GLUT-OpenGL seperti dibawah ini

//glutSolidCube(1.0f); // menggambar obyek kubus

//glutWireTeapot(1.0f);

// drawCube(); //panggil fungsi untuk membuat obyek kubus

//drawCylinder(1.0f, 2.0f, 20, 20); // fungsi untuk membuat obyek silinder

//drawSphere(1.0f, 50, 50); // fungsi untuk membuat obyek bola

drawTriangles();

glPopMatrix();

glPopMatrix();

// membuat obyek polyhedron

//glutSolidTetrahedron();

//glutSolidOctahedron();

//glutSolidDodecahedron();

//glutSolidIcosahedron();

//glutSolidCube(1.0f);

//glutSolidCone(1.0f, 1.0f, 50, 50);

//glutSolidSphere(1.0f, 50, 50);

//glutSolidTeapot(1.0f);

//glutSolidTorus(0.5f, 1.0f, 20, 20);

}

// taruh semua obyek yang akan digambar di fungsi display()

void display()

{

// bersihkan dan reset layar dan buffer

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

// posisikan kamera pandang

// dalam hal ini sumbu Y ada diatas dan posisi kamera pandang di (posX, posY, posZ)

gluLookAt(posX, posY, posZ, posX + rotX, posY + rotY, posZ + rotZ, 0.0f, 1.0f, 0.0f);

// panggil fungsi untuk menggambar obyek

drawObject();

// tampilkan obyek ke layar

// gunakan glFlush() bila memakai single buffer

// gunakan glutSwapBuffers() bila memakai double buffer

glutSwapBuffers();

}

// inisialisasikan variabel, pencahayaan, tekstur dan pengaturan kamera pandang di fungsi init()

void init(void)

{

// inisialisasi warna latar belakang layar dalam hal ini warna putih (1.0, 1.0, 1.0, 0.0)

glClearColor(1.0, 1.0, 1.0, 0.0);

glEnable(GL\_DEPTH\_TEST); // mengaktifkan depth buffer

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45.0, 1.0, 1.0, 100.0); // set proyeksi ke perspektif

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// inisialisasi kamera pandang

gluLookAt(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

glEnable(GL\_COLOR\_MATERIAL);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

}

// fungsi ini digunakan bila layar akan diresize (default)

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45, (GLfloat)w / (GLfloat)h, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

}

// fungsi untuk mengatur masukan dari keyboard

// untuk arah kiri, kanan, atas, bawah, PgUp, dan PgDn

void keyboard(int key, int x, int y)

{

float fraction = 0.1f;

switch (key)

{

// masukkan perintah disini bila tombol kiri ditekan

case GLUT\_KEY\_LEFT:

// dalam hal ini perintah rotasi obyek ke kiri sebanyak 1 derajat

objectAngleY -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol kanan ditekan

case GLUT\_KEY\_RIGHT:

// dalam hal ini perintah rotasi obyek ke kanan sebanyak 1 derajat

objectAngleY += 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol atas ditekan

case GLUT\_KEY\_UP:

// dalam hal ini perintah rotasi obyek ke atas sebanyak 1 derajat

objectAngleX -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol bawah ditekan

case GLUT\_KEY\_DOWN:

// dalam hal ini perintah rotasi obyek ke bawah sebanyak 1 derajat

objectAngleX += 1.0f;

glutPostRedisplay(); // update obyek

break;

// zoom in

case GLUT\_KEY\_PAGE\_UP:

// masukkan perintah disini bila tombol PgUp ditekan

posX += rotX \* fraction;

posZ += rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

// zoom out

case GLUT\_KEY\_PAGE\_DOWN:

// masukkan perintah disini bila tombol PgDn ditekan

posX -= rotX \* fraction;

posZ -= rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

}

}

// timer untuk animasi (gunakan bila perlu)

void timer(int value)

{

glutPostRedisplay();

glutTimerFunc(55, timer, 0);

}

// program utama

int main(int argc, char\*\* argv)

{

// inisialisasi jendela OpenGL

// GLUT\_SINGLE berarti memakai single buffer

// GLUT\_DOUBLE berarti memakai double buffer

// GLUT\_RGB berarti mode tampilan yang dipakai RGB

// GLUT\_RGBA berarti mode tampilan yang dipakai RGBA

// GLUT\_DEPTH berarti memakai depth buffer

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA | GLUT\_DEPTH);

// set ukuran jendela tampilan

glutInitWindowSize(480, 480); // besarnya jendela dalam piksel dalam hal ini 300x300

glutInitWindowPosition(100, 100); // posisi jendela dilayar komputer dalam piksel

// judul jendela (wajib diubah dengan informasi NAMA / NIM - JUDUL PRAKTIKUM masing-masing)

glutCreateWindow("SIRLY ZIADATUL MUSTAFIDAH / 2200018345 - REPRESENTASI OBYEK 3D");

// panggil fungsi init untuk inisialisasi awal

init();

// event handler untuk display, reshape dan keyboard

glutDisplayFunc(display); // display

glutReshapeFunc(reshape); // reshape

glutSpecialFunc(keyboard); // keyboard

//glutTimerFunc(0, timer, 0); // aktifkan timer bila perlu

// looping

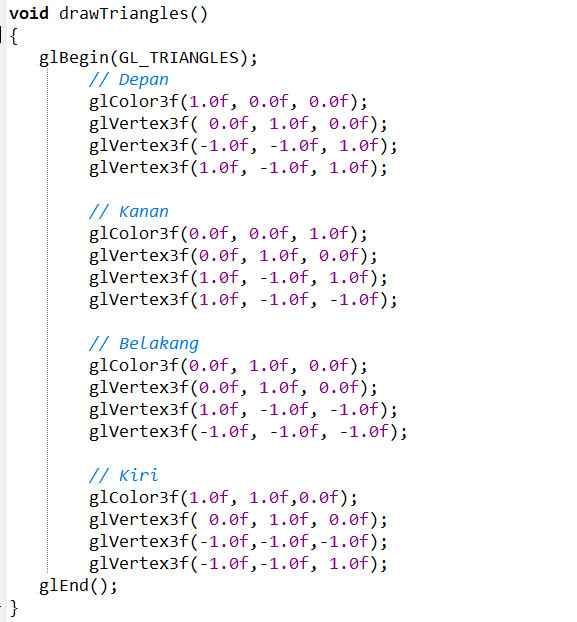
glutMainLoop();

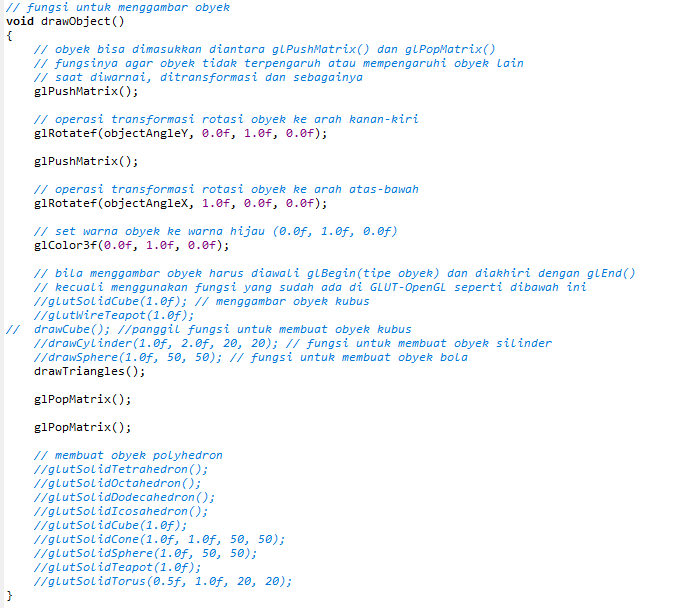
return 0;

}

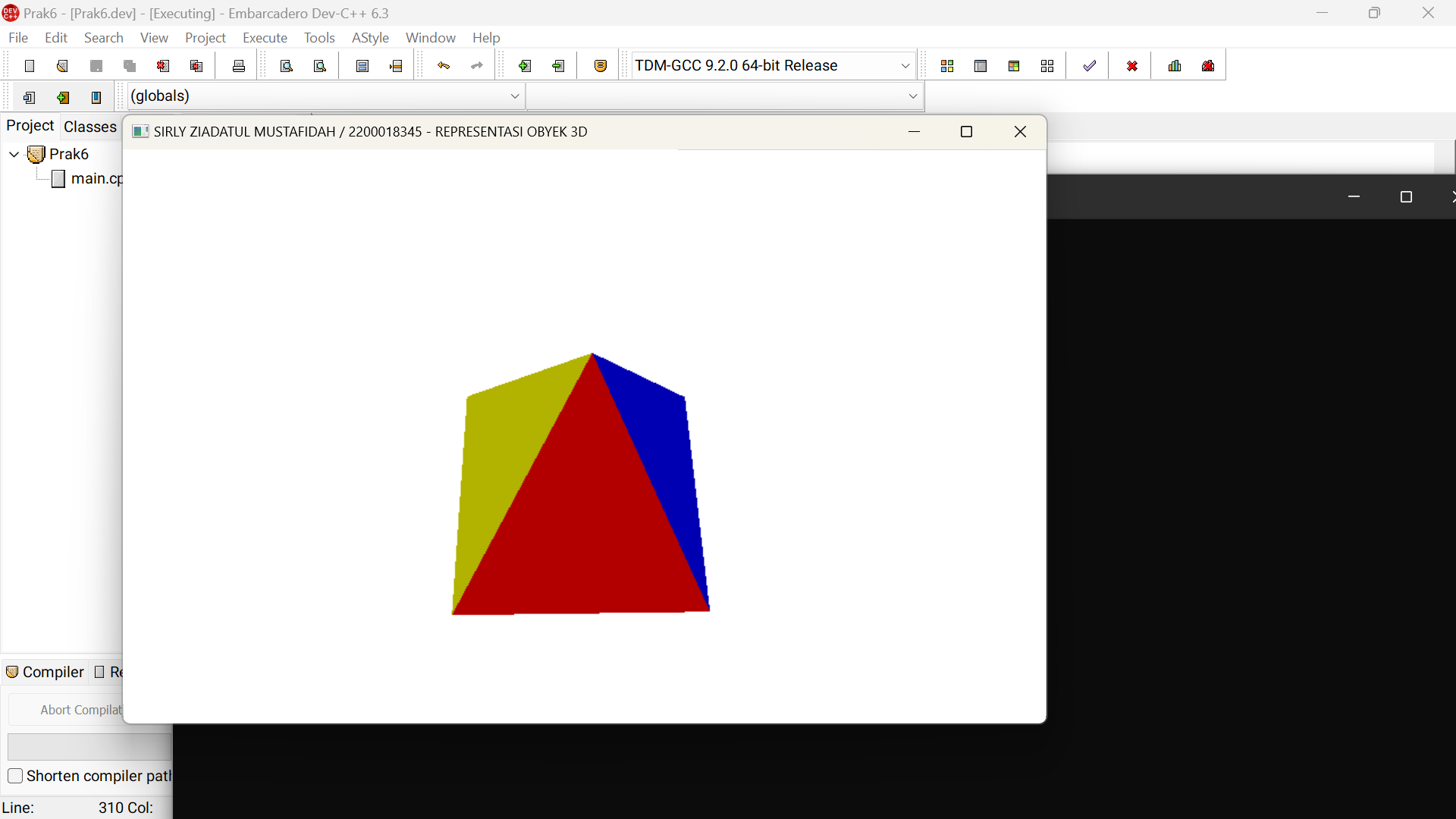
Penjelasan

Membuat fungsi untuk membuat piramid





Output



Gambar 6. 1

1. Posttest 6

Source Code

// kode dasar untuk semua praktikum grafika komputer

// adhi prahara. 2018

// deklarasikan semua header disini

#include <windows.h>

#include <GL/glut.h>

// inisialisasi variabel untuk transformasi seperti translasi, rotasi atau scaling

float angle = 0.0f; // sudut transformasi kamera

float posX = 0.0f, rotX = 0.0f; // posisi kamera di sumbu X

float posY = 0.0f, rotY = 0.0f; // posisi kamera di sumbu Y

float posZ = 5.0f, rotZ = -1.0f; // posisi kamera di sumbu Z

float objectAngleX = 0.0f; // sudut tranformasi obyek di sumbu X

float objectAngleY = 0.0f; // sudut tranformasi obyek di sumbu Y

float objectAngleZ = 0.0f; // sudut tranformasi obyek di sumbu Z

// container untuk membuat tipe data 3D (X, Y, Z)

struct Vec3

{

float X; float Y; float Z;

Vec3(float x, float y, float z) { X = x; Y = y; Z = z; }

//

Vec3() { }

~Vec3() { }

};

void drawCube()

{

glBegin(GL\_QUADS);

// beri warna merah di sisi depan

glColor3f(1.0f, 0.0f, 0.0f);

// buat sisi depan

glVertex3f(-1.0f, -1.0f, 1.0f);

glVertex3f( 1.0f, -1.0f, 1.0f);

glVertex3f( 1.0f, 1.0f, 1.0f);

glVertex3f(-1.0f, 1.0f, 1.0f);

// beri warna hijau di sisi belakang

glColor3f(0.0f, 1.0f, 0.0f);

// buat sisi belakang

glVertex3f(-1.0f, -1.0f,-1.0f);

glVertex3f( 1.0f, -1.0f,-1.0f);

glVertex3f( 1.0f, 1.0f,-1.0f);

glVertex3f(-1.0f, 1.0f,-1.0f);

// beri warna biru di sisi kiri

glColor3f(0.0f, 0.0f, 1.0f);

// buat sisi kiri

glVertex3f(-1.0f, -1.0f, 1.0f);

glVertex3f(-1.0f, -1.0f,-1.0f);

glVertex3f(-1.0f, 1.0f,-1.0f);

glVertex3f(-1.0f, 1.0f, 1.0f);

// beri warna cyan di sisi kanan

glColor3f(0.0f, 1.0f, 1.0f);

// buat sisi kanan

glVertex3f( 1.0f, -1.0f, 1.0f);

glVertex3f( 1.0f, -1.0f,-1.0f);

glVertex3f( 1.0f, 1.0f,-1.0f);

glVertex3f( 1.0f, 1.0f, 1.0f);

// beri warna kuning di sisi atas

glColor3f(1.0f, 1.0f, 0.0f);

// buat sisi atas

glVertex3f(-1.0f, 1.0f, 1.0f);

glVertex3f( 1.0f, 1.0f, 1.0f);

glVertex3f( 1.0f, 1.0f,-1.0f);

glVertex3f(-1.0f, 1.0f,-1.0f);

// beri warna magenta di sisi bawah

glColor3f(1.0f, 0.0f, 1.0f);

// buat sisi bawah

glVertex3f(-1.0f, -1.0f, 1.0f);

glVertex3f( 1.0f, -1.0f, 1.0f);

glVertex3f( 1.0f, -1.0f,-1.0f);

glVertex3f(-1.0f, -1.0f,-1.0f);

glEnd();

}

// fungsi untuk menggambar silinder

void drawCylinder(float radius, float height, int slices, int stacks)

{

glPushMatrix();

GLUquadricObj\* cyl = gluNewQuadric();

gluQuadricDrawStyle(cyl, GLU\_FILL);

gluQuadricNormals(cyl, GLU\_SMOOTH);

gluQuadricOrientation(cyl, GLU\_INSIDE);

// buat tutup atas silinder

glTranslatef(0.0f, -height/2, 0.0f);

glRotatef(-90, 1.0f, 0.0f, 0.0f);

glColor3f(1.0f, 1.0f, 0.0f); // warna kuning

gluDisk(cyl, 0.0f, radius, slices, stacks);

// buat badan silinder

glColor3f(1.0f, 0.0f, 0.0f); // warna merah

gluCylinder(cyl, radius, radius, height, slices, stacks);

// buat tutup bawah silinder

glColor3f(1.0f, 1.0f, 0.0f); // warna kuning

glTranslatef(0.0f, 0.0f, height);

gluDisk(cyl, 0.0f, radius, slices, stacks);

glPopMatrix();

}

// fungsi untuk menggambar bola

void drawSphere(float radius, int slices, int stacks)

{

glPushMatrix();

glColor3f(1.0f, 0.0f, 0.0f); // warna merah

GLUquadric \*sphere = gluNewQuadric();

gluQuadricDrawStyle(sphere, GLU\_FILL);

gluQuadricNormals(sphere, GLU\_SMOOTH);

gluSphere(sphere, radius, slices, stacks);

glPopMatrix();

}

void drawTriangles()

{

glBegin(GL\_TRIANGLES);

// Front

glColor3f(1.0f, 0.0f, 0.0f); // Red

glVertex3f( 0.0f, 1.0f, 0.0f);

glColor3f(0.0f, 1.0f, 0.0f); // Green

glVertex3f(-1.0f, -1.0f, 1.0f);

glColor3f(0.0f, 0.0f, 1.0f); // Blue

glVertex3f(1.0f, -1.0f, 1.0f);

// Right

glColor3f(1.0f, 0.0f, 0.0f); // Red

glVertex3f(0.0f, 1.0f, 0.0f);

glColor3f(0.0f, 0.0f, 1.0f); // Blue

glVertex3f(1.0f, -1.0f, 1.0f);

glColor3f(0.0f, 1.0f, 0.0f); // Green

glVertex3f(1.0f, -1.0f, -1.0f);

// Back

glColor3f(1.0f, 0.0f, 0.0f); // Red

glVertex3f(0.0f, 1.0f, 0.0f);

glColor3f(0.0f, 1.0f, 0.0f); // Green

glVertex3f(1.0f, -1.0f, -1.0f);

glColor3f(0.0f, 0.0f, 1.0f); // Blue

glVertex3f(-1.0f, -1.0f, -1.0f);

// Left

glColor3f(1.0f,0.0f,0.0f); // Red

glVertex3f( 0.0f, 1.0f, 0.0f);

glColor3f(0.0f,0.0f,1.0f); // Blue

glVertex3f(-1.0f,-1.0f,-1.0f);

glColor3f(0.0f,1.0f,0.0f); // Green

glVertex3f(-1.0f,-1.0f, 1.0f);

glEnd();

}

// fungsi untuk menggambar obyek

void drawObject()

{

// obyek bisa dimasukkan diantara glPushMatrix() dan glPopMatrix()

// fungsinya agar obyek tidak terpengaruh atau mempengaruhi obyek lain

// saat diwarnai, ditransformasi dan sebagainya

glPushMatrix();

// operasi transformasi rotasi obyek ke arah kanan-kiri

glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);

glPushMatrix();

// operasi transformasi rotasi obyek ke arah atas-bawah

glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

// set warna obyek ke warna hijau (0.0f, 1.0f, 0.0f)

glColor3f(0.0f, 1.0f, 0.0f);

// bila menggambar obyek harus diawali glBegin(tipe obyek) dan diakhiri dengan glEnd()

// kecuali menggunakan fungsi yang sudah ada di GLUT-OpenGL seperti dibawah ini

//glutSolidCube(1.0f); // menggambar obyek kubus

//glutWireTeapot(1.0f);

// drawCube(); //panggil fungsi untuk membuat obyek kubus

//drawCylinder(1.0f, 2.0f, 20, 20); // fungsi untuk membuat obyek silinder

//drawSphere(1.0f, 50, 50); // fungsi untuk membuat obyek bola

drawTriangles();

glPopMatrix();

glPopMatrix();

// membuat obyek polyhedron

//glutSolidTetrahedron();

//glutSolidOctahedron();

//glutSolidDodecahedron();

//glutSolidIcosahedron();

//glutSolidCube(1.0f);

//glutSolidCone(1.0f, 1.0f, 50, 50);

//glutSolidSphere(1.0f, 50, 50);

//glutSolidTeapot(1.0f);

//glutSolidTorus(0.5f, 1.0f, 20, 20);

}

// taruh semua obyek yang akan digambar di fungsi display()

void display()

{

// bersihkan dan reset layar dan buffer

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

// posisikan kamera pandang

// dalam hal ini sumbu Y ada diatas dan posisi kamera pandang di (posX, posY, posZ)

gluLookAt(posX, posY, posZ, posX + rotX, posY + rotY, posZ + rotZ, 0.0f, 1.0f, 0.0f);

// panggil fungsi untuk menggambar obyek

drawObject();

// tampilkan obyek ke layar

// gunakan glFlush() bila memakai single buffer

// gunakan glutSwapBuffers() bila memakai double buffer

glutSwapBuffers();

}

// inisialisasikan variabel, pencahayaan, tekstur dan pengaturan kamera pandang di fungsi init()

void init(void)

{

// inisialisasi warna latar belakang layar dalam hal ini warna putih (1.0, 1.0, 1.0, 0.0)

glClearColor(1.0, 1.0, 1.0, 0.0);

glEnable(GL\_DEPTH\_TEST); // mengaktifkan depth buffer

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45.0, 1.0, 1.0, 100.0); // set proyeksi ke perspektif

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// inisialisasi kamera pandang

gluLookAt(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

glEnable(GL\_COLOR\_MATERIAL);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

}

// fungsi ini digunakan bila layar akan diresize (default)

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45, (GLfloat)w / (GLfloat)h, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

}

// fungsi untuk mengatur masukan dari keyboard

// untuk arah kiri, kanan, atas, bawah, PgUp, dan PgDn

void keyboard(int key, int x, int y)

{

float fraction = 0.1f;

switch (key)

{

// masukkan perintah disini bila tombol kiri ditekan

case GLUT\_KEY\_LEFT:

// dalam hal ini perintah rotasi obyek ke kiri sebanyak 1 derajat

objectAngleY -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol kanan ditekan

case GLUT\_KEY\_RIGHT:

// dalam hal ini perintah rotasi obyek ke kanan sebanyak 1 derajat

objectAngleY += 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol atas ditekan

case GLUT\_KEY\_UP:

// dalam hal ini perintah rotasi obyek ke atas sebanyak 1 derajat

objectAngleX -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol bawah ditekan

case GLUT\_KEY\_DOWN:

// dalam hal ini perintah rotasi obyek ke bawah sebanyak 1 derajat

objectAngleX += 1.0f;

glutPostRedisplay(); // update obyek

break;

// zoom in

case GLUT\_KEY\_PAGE\_UP:

// masukkan perintah disini bila tombol PgUp ditekan

posX += rotX \* fraction;

posZ += rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

// zoom out

case GLUT\_KEY\_PAGE\_DOWN:

// masukkan perintah disini bila tombol PgDn ditekan

posX -= rotX \* fraction;

posZ -= rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

}

}

// timer untuk animasi (gunakan bila perlu)

void timer(int value)

{

glutPostRedisplay();

glutTimerFunc(55, timer, 0);

}

// program utama

int main(int argc, char\*\* argv)

{

// inisialisasi jendela OpenGL

// GLUT\_SINGLE berarti memakai single buffer

// GLUT\_DOUBLE berarti memakai double buffer

// GLUT\_RGB berarti mode tampilan yang dipakai RGB

// GLUT\_RGBA berarti mode tampilan yang dipakai RGBA

// GLUT\_DEPTH berarti memakai depth buffer

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA | GLUT\_DEPTH);

// set ukuran jendela tampilan

glutInitWindowSize(480, 480); // besarnya jendela dalam piksel dalam hal ini 300x300

glutInitWindowPosition(100, 100); // posisi jendela dilayar komputer dalam piksel

// judul jendela (wajib diubah dengan informasi NAMA / NIM - JUDUL PRAKTIKUM masing-masing)

glutCreateWindow("SIRLY ZIADATUL MUSTAFIDAH / 2200018345 - REPRESENTASI OBYEK 3D");

// panggil fungsi init untuk inisialisasi awal

init();

// event handler untuk display, reshape dan keyboard

glutDisplayFunc(display); // display

glutReshapeFunc(reshape); // reshape

glutSpecialFunc(keyboard); // keyboard

//glutTimerFunc(0, timer, 0); // aktifkan timer bila perlu

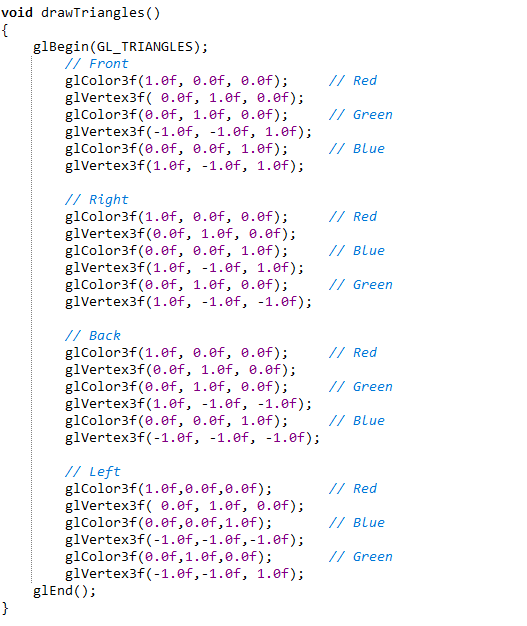
// looping

glutMainLoop();

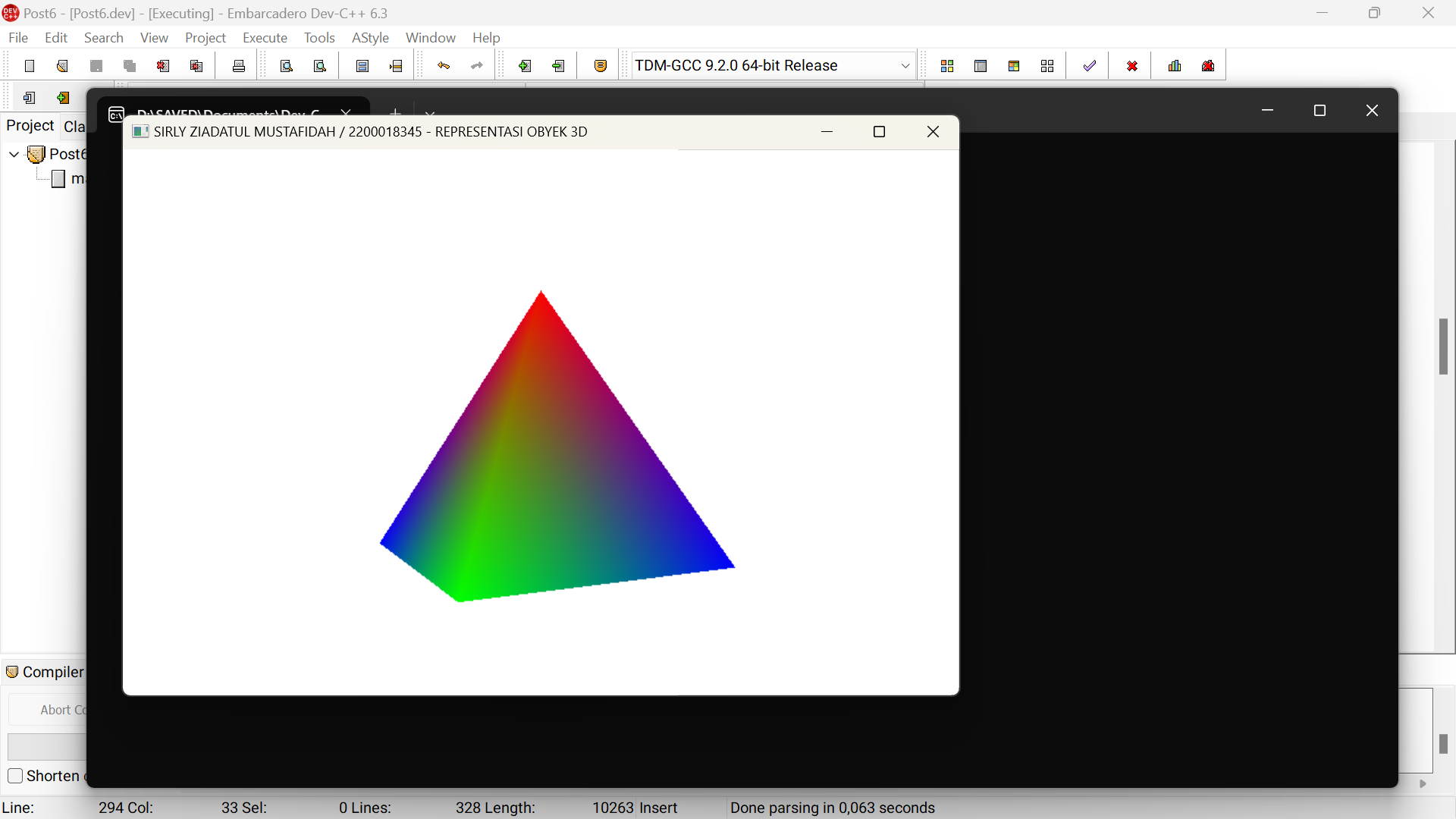
return 0;

}

Penjelasan



Output



Gambar 6. 2

# BAB VII. KURVA SPLINE

1. Pretest 7
2. Jelaskan perbedaan antara kurva spline Cubic, Catmull-Rom, Hermit, dan Bezier!

* Spline cubic menggunakan polynomial pangkat 3 dengan masukan 4 titik kontrol
* Catmull-rom menginterpolasi titk tengah pada titik kontrol
* Hermit menggunakan 2 titik ujung untuk interpolasi, kemudian tengah dihitung pada setiap titik untuk menentukan arah kurva
* Bezier menggunakan titik ujung dari titik kontrol untuk interpolasi, kemudian menghitung arah tengah untuk menentukan arah kurva

1. Langkah Praktikum 7

Source Code

// kode dasar untuk semua praktikum grafika komputer

// adhi prahara. 2018

// deklarasikan semua header disini

#include <windows.h>

#include <math.h>

#include <GL/glut.h>

#define SCREEN\_WIDTH 480

#define SCREEN\_HEIGHT 480

// inisialisasi variabel untuk transformasi seperti translasi, rotasi atau scaling

float angle = 0.0f; // sudut transformasi kamera

float posX = 0.0f, rotX = 0.0f; // posisi kamera di sumbu X

float posY = 0.0f, rotY = 0.0f; // posisi kamera di sumbu Y

float posZ = 5.0f, rotZ = -1.0f; // posisi kamera di sumbu Z

float objectAngleX = 0.0f; // sudut tranformasi obyek di sumbu X

float objectAngleY = 0.0f; // sudut tranformasi obyek di sumbu Y

float objectAngleZ = 0.0f; // sudut tranformasi obyek di sumbu Z

// container untuk membuat tipe data 3D (X, Y, Z)

struct Vec3

{

float X; float Y; float Z;

Vec3(float x, float y, float z) { X = x; Y = y; Z = z; }

//

Vec3() { }

~Vec3() { }

};

// menggambar setiap titik kontrol kurva

void markPoint(Vec3 points, Vec3 colors, float width)

{

// tandai setiap titik dengan warna

glPushMatrix();

glColor3f(colors.X, colors.Y, colors.Z);

glBegin(GL\_QUADS);

glVertex3f(points.X - width, points.Y - width, points.Z);

glVertex3f(points.X + width, points.Y - width, points.Z);

glVertex3f(points.X + width, points.Y + width, points.Z);

glVertex3f(points.X - width, points.Y + width, points.Z);

glEnd();

glPopMatrix();

}

// fungsi untuk menghitung invers matriks ordo 4x4

bool inverse(float inMat[16], float outMat[16])

{

float inv[16], det;

int i;

inv[0] =

inMat[5] \* inMat[10] \* inMat[15] -

inMat[5] \* inMat[11] \* inMat[14] -

inMat[9] \* inMat[6] \* inMat[15] +

inMat[9] \* inMat[7] \* inMat[14] +

inMat[13] \* inMat[6] \* inMat[11] -

inMat[13] \* inMat[7] \* inMat[10];

inv[4] =

-inMat[4] \* inMat[10] \* inMat[15] +

inMat[4] \* inMat[11] \* inMat[14] +

inMat[8] \* inMat[6] \* inMat[15] -

inMat[8] \* inMat[7] \* inMat[14] -

inMat[12] \* inMat[6] \* inMat[11] +

inMat[12] \* inMat[7] \* inMat[10];

inv[8] =

inMat[4] \* inMat[9] \* inMat[15] -

inMat[4] \* inMat[11] \* inMat[13] -

inMat[8] \* inMat[5] \* inMat[15] +

inMat[8] \* inMat[7] \* inMat[13] +

inMat[12] \* inMat[5] \* inMat[11] -

inMat[12] \* inMat[7] \* inMat[9];

inv[12] =

-inMat[4] \* inMat[9] \* inMat[14] +

inMat[4] \* inMat[10] \* inMat[13] +

inMat[8] \* inMat[5] \* inMat[14] -

inMat[8] \* inMat[6] \* inMat[13] -

inMat[12] \* inMat[5] \* inMat[10] +

inMat[12] \* inMat[6] \* inMat[9];

inv[1] =

-inMat[1] \* inMat[10] \* inMat[15] +

inMat[1] \* inMat[11] \* inMat[14] +

inMat[9] \* inMat[2] \* inMat[15] -

inMat[9] \* inMat[3] \* inMat[14] -

inMat[13] \* inMat[2] \* inMat[11] +

inMat[13] \* inMat[3] \* inMat[10];

inv[5] =

inMat[0] \* inMat[10] \* inMat[15] -

inMat[0] \* inMat[11] \* inMat[14] -

inMat[8] \* inMat[2] \* inMat[15] +

inMat[8] \* inMat[3] \* inMat[14] +

inMat[12] \* inMat[2] \* inMat[11] -

inMat[12] \* inMat[3] \* inMat[10];

inv[9] =

-inMat[0] \* inMat[9] \* inMat[15] +

inMat[0] \* inMat[11] \* inMat[13] +

inMat[8] \* inMat[1] \* inMat[15] -

inMat[8] \* inMat[3] \* inMat[13] -

inMat[12] \* inMat[1] \* inMat[11] +

inMat[12] \* inMat[3] \* inMat[9];

inv[13] =

inMat[0] \* inMat[9] \* inMat[14] -

inMat[0] \* inMat[10] \* inMat[13] -

inMat[8] \* inMat[1] \* inMat[14] +

inMat[8] \* inMat[2] \* inMat[13] +

inMat[12] \* inMat[1] \* inMat[10] -

inMat[12] \* inMat[2] \* inMat[9];

inv[2] =

inMat[1] \* inMat[6] \* inMat[15] -

inMat[1] \* inMat[7] \* inMat[14] -

inMat[5] \* inMat[2] \* inMat[15] +

inMat[5] \* inMat[3] \* inMat[14] +

inMat[13] \* inMat[2] \* inMat[7] -

inMat[13] \* inMat[3] \* inMat[6];

inv[6] =

-inMat[0] \* inMat[6] \* inMat[15] +

inMat[0] \* inMat[7] \* inMat[14] +

inMat[4] \* inMat[2] \* inMat[15] -

inMat[4] \* inMat[3] \* inMat[14] -

inMat[12] \* inMat[2] \* inMat[7] +

inMat[12] \* inMat[3] \* inMat[6];

inv[10] =

inMat[0] \* inMat[5] \* inMat[15] -

inMat[0] \* inMat[7] \* inMat[13] -

inMat[4] \* inMat[1] \* inMat[15] +

inMat[4] \* inMat[3] \* inMat[13] +

inMat[12] \* inMat[1] \* inMat[7] -

inMat[12] \* inMat[3] \* inMat[5];

inv[14] =

-inMat[0] \* inMat[5] \* inMat[14] +

inMat[0] \* inMat[6] \* inMat[13] +

inMat[4] \* inMat[1] \* inMat[14] -

inMat[4] \* inMat[2] \* inMat[13] -

inMat[12] \* inMat[1] \* inMat[6] +

inMat[12] \* inMat[2] \* inMat[5];

inv[3] =

-inMat[1] \* inMat[6] \* inMat[11] +

inMat[1] \* inMat[7] \* inMat[10] +

inMat[5] \* inMat[2] \* inMat[11] -

inMat[5] \* inMat[3] \* inMat[10] -

inMat[9] \* inMat[2] \* inMat[7] +

inMat[9] \* inMat[3] \* inMat[6];

inv[7] =

inMat[0] \* inMat[6] \* inMat[11] -

inMat[0] \* inMat[7] \* inMat[10] -

inMat[4] \* inMat[2] \* inMat[11] +

inMat[4] \* inMat[3] \* inMat[10] +

inMat[8] \* inMat[2] \* inMat[7] -

inMat[8] \* inMat[3] \* inMat[6];

inv[11] =

-inMat[0] \* inMat[5] \* inMat[11] +

inMat[0] \* inMat[7] \* inMat[9] +

inMat[4] \* inMat[1] \* inMat[11] -

inMat[4] \* inMat[3] \* inMat[9] -

inMat[8] \* inMat[1] \* inMat[7] +

inMat[8] \* inMat[3] \* inMat[5];

inv[15] =

inMat[0] \* inMat[5] \* inMat[10] -

inMat[0] \* inMat[6] \* inMat[9] -

inMat[4] \* inMat[1] \* inMat[10] +

inMat[4] \* inMat[2] \* inMat[9] +

inMat[8] \* inMat[1] \* inMat[6] -

inMat[8] \* inMat[2] \* inMat[5];

det = inMat[0] \* inv[0] + inMat[1] \* inv[4] + inMat[2] \* inv[8] + inMat[3] \* inv[12];

if (det == 0)

return false;

det = 1.0 / det;

for (i = 0; i < 16; i++)

outMat[i] = inv[i] \* det;

return true;

}

// fungsi untuk perkalian matriks 4x4 dengan 4x1

void DotMatrix(float inMat1[16], float inMat2[4], float outMat[4])

{

outMat[0] = inMat1[0] \* inMat2[0] + inMat1[1] \* inMat2[1] +

inMat1[2] \* inMat2[2] + inMat1[3] \* inMat2[3];

outMat[1] = inMat1[4] \* inMat2[0] + inMat1[5] \* inMat2[1] +

inMat1[6] \* inMat2[2] + inMat1[7] \* inMat2[3];

outMat[2] = inMat1[8] \* inMat2[0] + inMat1[9] \* inMat2[1] +

inMat1[10] \* inMat2[2] + inMat1[11] \* inMat2[3];

outMat[3] = inMat1[12] \* inMat2[0] + inMat1[13] \* inMat2[1] +

inMat1[14] \* inMat2[2] + inMat1[15] \* inMat2[3];

}

// fungsi untuk membuat kurva spline cubic dari 4 titik kontrol

// point1 sampai point4 = titik kontrol

// nPoint = jumlah titik interpolasi antara point1 sampai point4

void drawSplineCubic(Vec3 point1, Vec3 point2, Vec3 point3, Vec3 point4, int nPoint)

{

// hitung bobot jarak u di masing-masing titik

float utotal = (abs((int)(point2.X - point1.X)) + abs((int)(point3.X - point2.X)) + abs((int)(point4.X - point3.X)));

float u1 = 0;

float u2 = abs((int)(point2.X - point1.X)) / utotal;

float u3 = (abs((int)(point2.X - point1.X)) + abs((int)(point3.X - point2.X))) / utotal;

float u4 = 1;

// hitung inverse matriks dari koefisien u (lihat slide kuliah)

float inverseMat[16];

float coeffMat[16] = {

1.00f, 0.00f, 0.00f, 0.00f,

1.00f, u2, pow(u2, 2), pow(u2, 3),

1.00f, u3, pow(u3, 2), pow(u3, 3),

1.00f, 1.00f, 1.00f, 1.00f };

bool status = inverse(coeffMat, inverseMat);

// hitung koefisien cubic au^3 + bu^2 + cu + d

if (status == true)

{

float outMatX[4], outMatY[4], outMatZ[4];

float inMatX[4] = { point1.X, point2.X, point3.X, point4.X };

float inMatY[4] = { point1.Y, point2.Y, point3.Y, point4.Y };

float inMatZ[4] = { point1.Z, point2.Z, point3.Z, point4.Z };

DotMatrix(inverseMat, inMatX, outMatX);

DotMatrix(inverseMat, inMatY, outMatY);

DotMatrix(inverseMat, inMatZ, outMatZ);

// gambar kurva cubic spline dengan titik kontrol diatas

// hitung posisi y untuk setiap x di setiap point dengan persamaan diatas

for (int i=0; i<nPoint; i++)

{

float step = 1.0f / nPoint; // jeda setiap titik pd bobot u

// titik awal

float pX = point1.X, pY = point1.Y, pZ = point1.Z;

//

float u = 0.0f;

for (int i = 0; i < nPoint; i++)

{

// bentuk segment kurva cubic spline sebanyak nPoint

u = u + step;

glVertex3f(pX, pY, pZ); // gambar titik awal

// koordinat X pada kurva

pX = outMatX[3] \* pow(u, 3) + outMatX[2] \* pow(u, 2)

+ outMatX[1] \* u + outMatX[0];

// koordinat Y pada kurva

pY = outMatY[3] \* pow(u, 3) + outMatY[2] \* pow(u, 2)

+ outMatY[1] \* u + outMatY[0];

// koordinat Z pada kurva

pZ = outMatZ[3] \* pow(u, 3) + outMatZ[2] \* pow(u, 2)

+ outMatZ[1] \* u + outMatZ[0];

glVertex3f(pX, pY, pZ); // gambar titik akhir

}

}

}

}

// fungsi untuk membuat kurva spline bezier dari 4 titik kontrol

// point1 dan point4 = titik kontrol awal dan akhir

// point2 dan point3 = titik kontrol pembentuk kurva

// nPoint = jumlah titik interpolasi antara point1 sampai point4

void drawSplineBezier(Vec3 point1, Vec3 point2, Vec3 point3, Vec3 point4, int nPoint)

{

// hitung bobot jarak u di masing-masing titik

float utotal = (abs((int)(point2.X - point1.X)) + abs((int)(point3.X - point2.X)) + abs((int)(point4.X - point3.X)));

float u1 = 0;

float u2 = abs((int)(point2.X - point1.X)) / utotal;

float u3 = (abs((int)(point2.X - point1.X)) + abs((int)(point3.X - point2.X))) / utotal;

float u4 = 1;

// hitung inverse matriks dari koefisien u (lihat slide kuliah)

float inverseMat[16];

float coeffMat[16] = {

1.0000f, 0.0000f, 0.0000f, 0.0000f,

1.0000f, 1.0000f, 1.0000f, 1.0000f,

0.0000f, 1.0000f, 0.0000f, 0.0000f,

0.0000f, 1.0000f, 2.0000f, 3.0000f };

bool status = inverse(coeffMat, inverseMat);

// hitung koefisien

if (status == true)

{

float outMatX[4], outMatY[4], outMatZ[4];

float inMatX[4] = { point1.X, point4.X,

1.0f/(u2-u1)\*(point2.X - point1.X),

1.0f/(u4-u3)\*(point4.X - point3.X) };

float inMatY[4] = { point1.Y, point4.Y,

1.0f/(u2-u1)\*(point2.Y - point1.Y),

1.0f/(u4-u3)\*(point4.Y - point3.Y) };

float inMatZ[4] = { point1.Z, point4.Z,

1.0f/(u2-u1)\*(point2.Z - point1.Z),

1.0f/(u4-u3)\*(point4.Z - point3.Z) };

DotMatrix(inverseMat, inMatX, outMatX);

DotMatrix(inverseMat, inMatY, outMatY);

DotMatrix(inverseMat, inMatZ, outMatZ);

// gambar kurva spline dengan titik kontrol diatas

// hitung posisi y untuk setiap x di setiap point dengan persamaan diatas

for (int i=0; i<nPoint; i++)

{

float step = 1.0f / nPoint; // jeda setiap titik pd bobot u

// titik awal

float pX = point1.X, pY = point1.Y, pZ = point1.Z;

//

float u = 0.0f;

for (int i = 0; i < nPoint; i++)

{

// bentuk segment kurva spline sebanyak nPoint

u = u + step;

glVertex3f(pX, pY, pZ); // gambar titik awal

// koordinat X pada kurva

pX = outMatX[3] \* pow(u, 3) + outMatX[2] \* pow(u, 2)

+ outMatX[1] \* u + outMatX[0];

// koordinat Y pada kurva

pY = outMatY[3] \* pow(u, 3) + outMatY[2] \* pow(u, 2)

+ outMatY[1] \* u + outMatY[0];

// koordinat Z pada kurva

pZ = outMatZ[3] \* pow(u, 3) + outMatZ[2] \* pow(u, 2)

+ outMatZ[1] \* u + outMatZ[0];

glVertex3f(pX, pY, pZ); // gambar titik akhir

}

}

}

}

// fungsi untuk membuat kurva spline catmull-rom dari 4 titik kontrol

// point1 dan point4 = titik kontrol awal dan akhir

// point2 dan point3 = titik kontrol pembentuk kurva

// nPoint = jumlah titik interpolasi antara point1 sampai point4

void drawSplineCatmullRom(Vec3 point1, Vec3 point2, Vec3 point3, Vec3 point4, int nPoint)

{

// hitung bobot jarak u di masing-masing titik

float utotal = (abs((int)(point2.X - point1.X)) + abs((int)(point3.X - point2.X)) + abs((int)(point4.X - point3.X)));

float u1 = 0;

float u2 = abs((int)(point2.X - point1.X)) / utotal;

float u3 = (abs((int)(point2.X - point1.X)) + abs((int)(point3.X - point2.X))) / utotal;

float u4 = 1;

// hitung inverse matriks dari koefisien u (lihat slide kuliah)

float inverseMat[16];

float coeffMat[16] = {

1.0000f, 0.0000f, 0.0000f, 0.0000f,

1.0000f, 1.0000f, 1.0000f, 1.0000f,

0.0000f, 1.0000f, 0.0000f, 0.0000f,

0.0000f, 1.0000f, 2.0000f, 3.0000f };

bool status = inverse(coeffMat, inverseMat);

// hitung koefisien

if (status == true)

{

float outMatX[4], outMatY[4], outMatZ[4];

float inMatX[4] = { point1.X, point4.X,

1.0f/(u3-u1)\*(point3.X - point1.X),

1.0f/(u4-u2)\*(point4.X - point2.X) };

float inMatY[4] = { point1.Y, point4.Y,

1.0f/(u3-u1)\*(point3.Y - point1.Y),

1.0f/(u4-u2)\*(point4.Y - point2.Y) };

float inMatZ[4] = { point1.Z, point4.Z,

1.0f/(u3-u1)\*(point3.Z - point1.Z),

1.0f/(u4-u2)\*(point4.Z - point2.Z) };

DotMatrix(inverseMat, inMatX, outMatX);

DotMatrix(inverseMat, inMatY, outMatY);

DotMatrix(inverseMat, inMatZ, outMatZ);

// gambar kurva spline dengan titik kontrol diatas

// hitung posisi y untuk setiap x di setiap point dengan persamaan diatas

for (int i=0; i<nPoint; i++)

{

float step = 1.0f / nPoint; // jeda setiap titik pd bobot u

// titik awal

float pX = point1.X, pY = point1.Y, pZ = point1.Z;

//

float u = 0.0f;

for (int i = 0; i < nPoint; i++)

{

// bentuk segment kurva spline sebanyak nPoint

u = u + step;

glVertex3f(pX, pY, pZ); // gambar titik awal

// koordinat X pada kurva

pX = outMatX[3] \* pow(u, 3) + outMatX[2] \* pow(u, 2)

+ outMatX[1] \* u + outMatX[0];

// koordinat Y pada kurva

pY = outMatY[3] \* pow(u, 3) + outMatY[2] \* pow(u, 2)

+ outMatY[1] \* u + outMatY[0];

// koordinat Z pada kurva

pZ = outMatZ[3] \* pow(u, 3) + outMatZ[2] \* pow(u, 2)

+ outMatZ[1] \* u + outMatZ[0];

glVertex3f(pX, pY, pZ); // gambar titik akhir

}

}

}

}

// fungsi untuk menggambar obyek kubus

void drawObject()

{

// obyek bisa dimasukkan diantara glPushMatrix() dan glPopMatrix()

// fungsinya agar obyek tidak terpengaruh atau mempengaruhi obyek lain

// saat diwarnai, ditransformasi dan sebagainya

glPushMatrix();

// operasi transformasi rotasi obyek ke arah kanan-kiri

glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);

glPushMatrix();

// operasi transformasi rotasi obyek ke arah atas-bawah

glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

// membuat 4 titik kontrol kurva

Vec3 point1 = Vec3(-200.0f, -70.0f, 0.0f);

Vec3 point2 = Vec3( -50.0f, 50.0f, 0.0f);

Vec3 point3 = Vec3( 50.0f, 10.0f, 0.0f);

Vec3 point4 = Vec3( 150.0f, 50.0f, 0.0f);

// tandai setiap titik kontrol kurva dengan warna

markPoint(point1, Vec3(1.0f, 0.0f, 0.0f), 5.0f);

markPoint(point2, Vec3(0.0f, 1.0f, 0.0f), 5.0f);

markPoint(point3, Vec3(0.0f, 0.0f, 1.0f), 5.0f);

markPoint(point4, Vec3(1.0f, 1.0f, 0.0f), 5.0f);

// mengatur warna obyek menjadi berwarna putih

glColor3f(1.0f, 1.0f, 1.0f);

glBegin(GL\_LINES);

// membuat kurva spline cubic dari titik kontrol diatas

// drawSplineCubic(point1, point2, point3, point4, 30);

drawSplineBezier(point1, point2, point3, point4, 30);

// drawSplineCatmullRom(point1, point2, point3, point4, 30);

glEnd();

glPopMatrix();

glPopMatrix();

}

// taruh semua obyek yang akan digambar di fungsi display()

void display()

{

// bersihkan dan reset layar dan buffer

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

// posisikan kamera pandang

// dalam hal ini sumbu Y ada diatas dan posisi kamera pandang di (posX, posY, posZ)

gluLookAt(posX, posY, posZ, posX + rotX, posY + rotY, posZ + rotZ, 0.0f, 1.0f, 0.0f);

// panggil fungsi untuk menggambar obyek

drawObject();

// tampilkan obyek ke layar

// gunakan glFlush() bila memakai single buffer

// gunakan glutSwapBuffers() bila memakai double buffer

glutSwapBuffers();

}

// inisialisasikan variabel, pencahayaan, tekstur dan pengaturan kamera pandang di fungsi init()

void init(void)

{

// inisialisasi warna latar belakang layar dalam hal ini warna putih (1.0, 1.0, 1.0, 0.0)

glClearColor(0.0, 0.0, 0.0, 0.0);

glEnable(GL\_DEPTH\_TEST); // mengaktifkan depth buffer

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

//gluPerspective(45.0, 1.0, 1.0, 100.0); // set proyeksi ke perspektif

glOrtho((GLfloat)-SCREEN\_WIDTH / 2, (GLfloat)SCREEN\_WIDTH / 2,

(GLfloat)-SCREEN\_HEIGHT / 2, (GLfloat)SCREEN\_HEIGHT / 2, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// inisialisasi kamera pandang

gluLookAt(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

}

// fungsi ini digunakan bila layar akan diresize (default)

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

//gluPerspective(45, (GLfloat)w / (GLfloat)h, 1.0, 100.0);

glOrtho((GLfloat)-SCREEN\_WIDTH / 2, (GLfloat)SCREEN\_WIDTH / 2,

(GLfloat)-SCREEN\_HEIGHT / 2, (GLfloat)SCREEN\_HEIGHT / 2, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

}

// fungsi untuk mengatur masukan dari keyboard

// untuk arah kiri, kanan, atas, bawah, PgUp, dan PgDn

void keyboard(int key, int x, int y)

{

float fraction = 0.1f;

switch (key)

{

// masukkan perintah disini bila tombol kiri ditekan

case GLUT\_KEY\_LEFT:

// dalam hal ini perintah rotasi obyek ke kiri sebanyak 1 derajat

objectAngleY -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol kanan ditekan

case GLUT\_KEY\_RIGHT:

// dalam hal ini perintah rotasi obyek ke kanan sebanyak 1 derajat

objectAngleY += 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol atas ditekan

case GLUT\_KEY\_UP:

// dalam hal ini perintah rotasi obyek ke atas sebanyak 1 derajat

objectAngleX -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol bawah ditekan

case GLUT\_KEY\_DOWN:

// dalam hal ini perintah rotasi obyek ke bawah sebanyak 1 derajat

objectAngleX += 1.0f;

glutPostRedisplay(); // update obyek

break;

// zoom in

case GLUT\_KEY\_PAGE\_UP:

// masukkan perintah disini bila tombol PgUp ditekan

posX += rotX \* fraction;

posZ += rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

// zoom out

case GLUT\_KEY\_PAGE\_DOWN:

// masukkan perintah disini bila tombol PgDn ditekan

posX -= rotX \* fraction;

posZ -= rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

}

}

// timer untuk animasi (gunakan bila perlu)

void timer(int value)

{

glutPostRedisplay();

glutTimerFunc(55, timer, 0);

}

// program utama

int main(int argc, char\*\* argv)

{

// inisialisasi jendela OpenGL

// GLUT\_SINGLE berarti memakai single buffer

// GLUT\_DOUBLE berarti memakai double buffer

// GLUT\_RGB berarti mode tampilan yang dipakai RGB

// GLUT\_RGBA berarti mode tampilan yang dipakai RGBA

// GLUT\_DEPTH berarti memakai depth buffer

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA | GLUT\_DEPTH);

// set ukuran jendela tampilan

glutInitWindowSize(480, 480); // besarnya jendela dalam piksel dalam hal ini 300x300

glutInitWindowPosition(100, 100); // posisi jendela dilayar komputer dalam piksel

// judul jendela (wajib diubah dengan informasi NAMA / NIM - JUDUL PRAKTIKUM masing-masing)

glutCreateWindow("SIRLY ZIADATUL MUSTAFIDAH / 2200018345 - KURVA SPLINE");

// panggil fungsi init untuk inisialisasi awal

init();

// event handler untuk display, reshape dan keyboard

glutDisplayFunc(display); // display

glutReshapeFunc(reshape); // reshape

glutSpecialFunc(keyboard); // keyboard

//glutTimerFunc(0, timer, 0); // aktifkan timer bila perlu

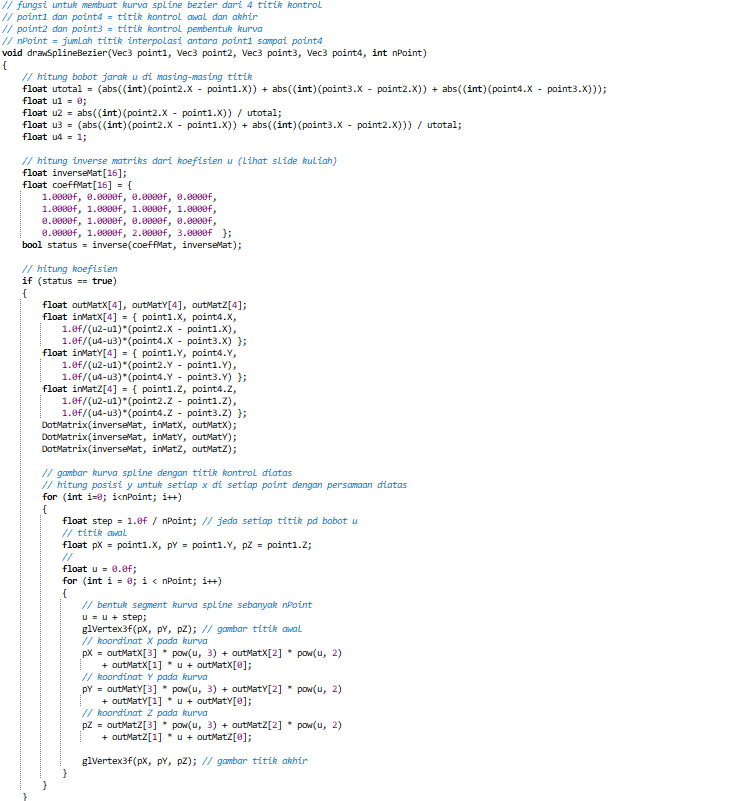
// looping

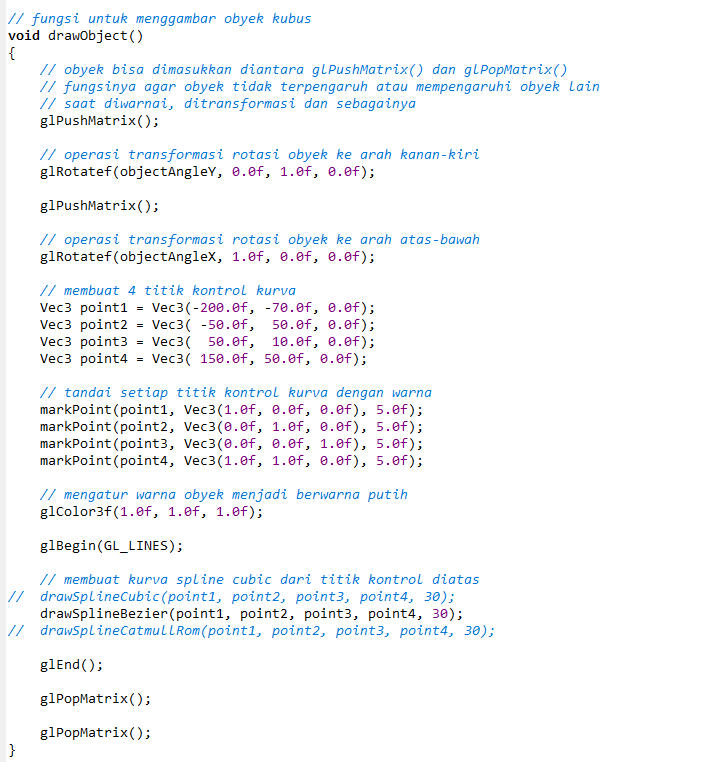
glutMainLoop();

return 0;

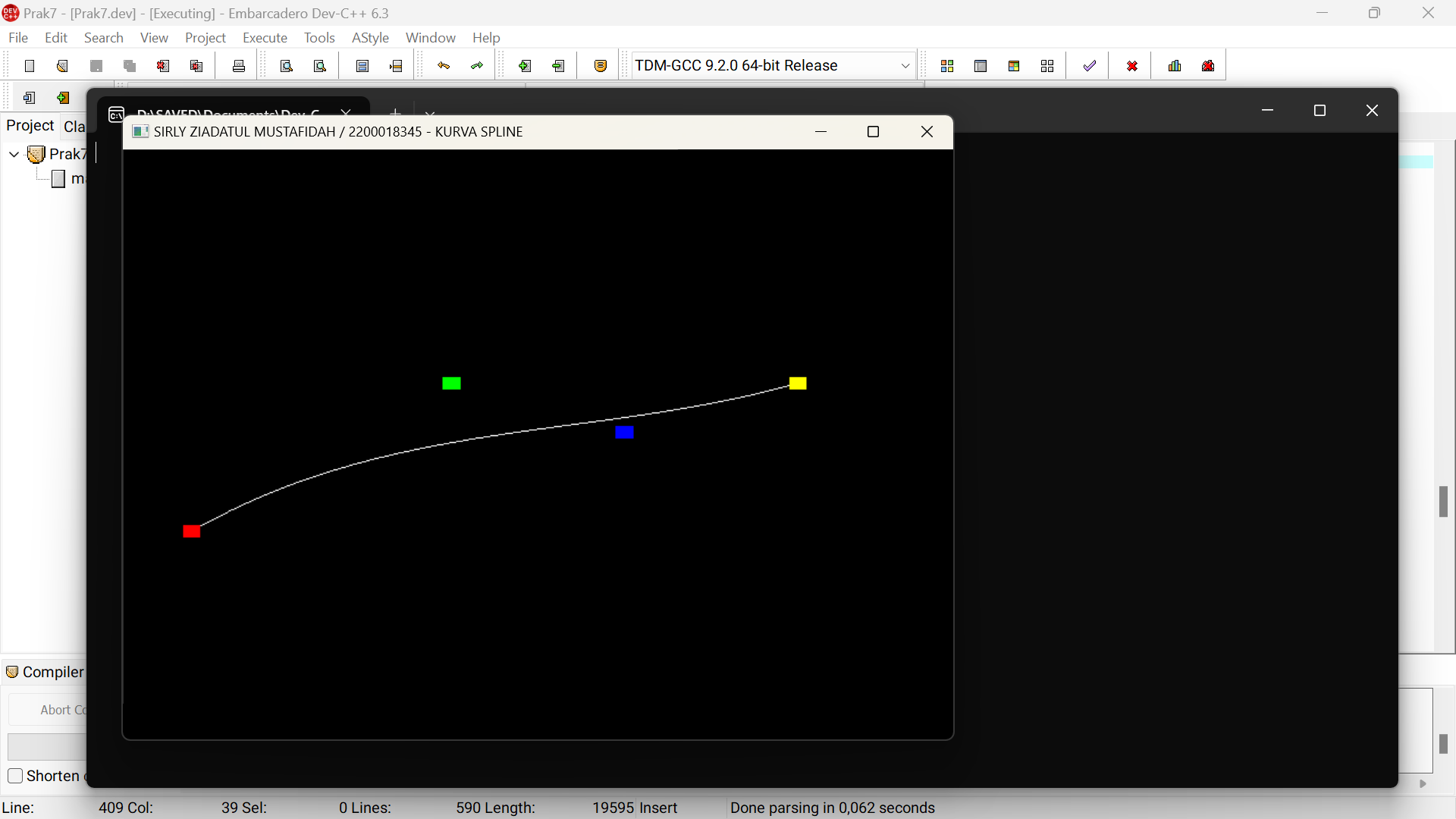
}

Penjelasan





Output



Gambar 7. 1

1. Posttest 7

Source Code

// kode dasar untuk semua praktikum grafika komputer

// adhi prahara. 2018

// deklarasikan semua header disini

#include <windows.h>

#include <math.h>

#include <GL/glut.h>

#define SCREEN\_WIDTH 480

#define SCREEN\_HEIGHT 480

// inisialisasi variabel untuk transformasi seperti translasi, rotasi atau scaling

float angle = 0.0f; // sudut transformasi kamera

float posX = 0.0f, rotX = 0.0f; // posisi kamera di sumbu X

float posY = 0.0f, rotY = 0.0f; // posisi kamera di sumbu Y

float posZ = 5.0f, rotZ = -1.0f; // posisi kamera di sumbu Z

float objectAngleX = 0.0f; // sudut tranformasi obyek di sumbu X

float objectAngleY = 0.0f; // sudut tranformasi obyek di sumbu Y

float objectAngleZ = 0.0f; // sudut tranformasi obyek di sumbu Z

// container untuk membuat tipe data 3D (X, Y, Z)

struct Vec3

{

float X; float Y; float Z;

Vec3(float x, float y, float z) { X = x; Y = y; Z = z; }

//

Vec3() { }

~Vec3() { }

};

// menggambar setiap titik kontrol kurva

void markPoint(Vec3 points, Vec3 colors, float width)

{

// tandai setiap titik dengan warna

glPushMatrix();

glColor3f(colors.X, colors.Y, colors.Z);

glBegin(GL\_QUADS);

glVertex3f(points.X - width, points.Y - width, points.Z);

glVertex3f(points.X + width, points.Y - width, points.Z);

glVertex3f(points.X + width, points.Y + width, points.Z);

glVertex3f(points.X - width, points.Y + width, points.Z);

glEnd();

glPopMatrix();

}

// fungsi untuk menghitung invers matriks ordo 4x4

bool inverse(float inMat[16], float outMat[16])

{

float inv[16], det;

int i;

inv[0] =

inMat[5] \* inMat[10] \* inMat[15] -

inMat[5] \* inMat[11] \* inMat[14] -

inMat[9] \* inMat[6] \* inMat[15] +

inMat[9] \* inMat[7] \* inMat[14] +

inMat[13] \* inMat[6] \* inMat[11] -

inMat[13] \* inMat[7] \* inMat[10];

inv[4] =

-inMat[4] \* inMat[10] \* inMat[15] +

inMat[4] \* inMat[11] \* inMat[14] +

inMat[8] \* inMat[6] \* inMat[15] -

inMat[8] \* inMat[7] \* inMat[14] -

inMat[12] \* inMat[6] \* inMat[11] +

inMat[12] \* inMat[7] \* inMat[10];

inv[8] =

inMat[4] \* inMat[9] \* inMat[15] -

inMat[4] \* inMat[11] \* inMat[13] -

inMat[8] \* inMat[5] \* inMat[15] +

inMat[8] \* inMat[7] \* inMat[13] +

inMat[12] \* inMat[5] \* inMat[11] -

inMat[12] \* inMat[7] \* inMat[9];

inv[12] =

-inMat[4] \* inMat[9] \* inMat[14] +

inMat[4] \* inMat[10] \* inMat[13] +

inMat[8] \* inMat[5] \* inMat[14] -

inMat[8] \* inMat[6] \* inMat[13] -

inMat[12] \* inMat[5] \* inMat[10] +

inMat[12] \* inMat[6] \* inMat[9];

inv[1] =

-inMat[1] \* inMat[10] \* inMat[15] +

inMat[1] \* inMat[11] \* inMat[14] +

inMat[9] \* inMat[2] \* inMat[15] -

inMat[9] \* inMat[3] \* inMat[14] -

inMat[13] \* inMat[2] \* inMat[11] +

inMat[13] \* inMat[3] \* inMat[10];

inv[5] =

inMat[0] \* inMat[10] \* inMat[15] -

inMat[0] \* inMat[11] \* inMat[14] -

inMat[8] \* inMat[2] \* inMat[15] +

inMat[8] \* inMat[3] \* inMat[14] +

inMat[12] \* inMat[2] \* inMat[11] -

inMat[12] \* inMat[3] \* inMat[10];

inv[9] =

-inMat[0] \* inMat[9] \* inMat[15] +

inMat[0] \* inMat[11] \* inMat[13] +

inMat[8] \* inMat[1] \* inMat[15] -

inMat[8] \* inMat[3] \* inMat[13] -

inMat[12] \* inMat[1] \* inMat[11] +

inMat[12] \* inMat[3] \* inMat[9];

inv[13] =

inMat[0] \* inMat[9] \* inMat[14] -

inMat[0] \* inMat[10] \* inMat[13] -

inMat[8] \* inMat[1] \* inMat[14] +

inMat[8] \* inMat[2] \* inMat[13] +

inMat[12] \* inMat[1] \* inMat[10] -

inMat[12] \* inMat[2] \* inMat[9];

inv[2] =

inMat[1] \* inMat[6] \* inMat[15] -

inMat[1] \* inMat[7] \* inMat[14] -

inMat[5] \* inMat[2] \* inMat[15] +

inMat[5] \* inMat[3] \* inMat[14] +

inMat[13] \* inMat[2] \* inMat[7] -

inMat[13] \* inMat[3] \* inMat[6];

inv[6] =

-inMat[0] \* inMat[6] \* inMat[15] +

inMat[0] \* inMat[7] \* inMat[14] +

inMat[4] \* inMat[2] \* inMat[15] -

inMat[4] \* inMat[3] \* inMat[14] -

inMat[12] \* inMat[2] \* inMat[7] +

inMat[12] \* inMat[3] \* inMat[6];

inv[10] =

inMat[0] \* inMat[5] \* inMat[15] -

inMat[0] \* inMat[7] \* inMat[13] -

inMat[4] \* inMat[1] \* inMat[15] +

inMat[4] \* inMat[3] \* inMat[13] +

inMat[12] \* inMat[1] \* inMat[7] -

inMat[12] \* inMat[3] \* inMat[5];

inv[14] =

-inMat[0] \* inMat[5] \* inMat[14] +

inMat[0] \* inMat[6] \* inMat[13] +

inMat[4] \* inMat[1] \* inMat[14] -

inMat[4] \* inMat[2] \* inMat[13] -

inMat[12] \* inMat[1] \* inMat[6] +

inMat[12] \* inMat[2] \* inMat[5];

inv[3] =

-inMat[1] \* inMat[6] \* inMat[11] +

inMat[1] \* inMat[7] \* inMat[10] +

inMat[5] \* inMat[2] \* inMat[11] -

inMat[5] \* inMat[3] \* inMat[10] -

inMat[9] \* inMat[2] \* inMat[7] +

inMat[9] \* inMat[3] \* inMat[6];

inv[7] =

inMat[0] \* inMat[6] \* inMat[11] -

inMat[0] \* inMat[7] \* inMat[10] -

inMat[4] \* inMat[2] \* inMat[11] +

inMat[4] \* inMat[3] \* inMat[10] +

inMat[8] \* inMat[2] \* inMat[7] -

inMat[8] \* inMat[3] \* inMat[6];

inv[11] =

-inMat[0] \* inMat[5] \* inMat[11] +

inMat[0] \* inMat[7] \* inMat[9] +

inMat[4] \* inMat[1] \* inMat[11] -

inMat[4] \* inMat[3] \* inMat[9] -

inMat[8] \* inMat[1] \* inMat[7] +

inMat[8] \* inMat[3] \* inMat[5];

inv[15] =

inMat[0] \* inMat[5] \* inMat[10] -

inMat[0] \* inMat[6] \* inMat[9] -

inMat[4] \* inMat[1] \* inMat[10] +

inMat[4] \* inMat[2] \* inMat[9] +

inMat[8] \* inMat[1] \* inMat[6] -

inMat[8] \* inMat[2] \* inMat[5];

det = inMat[0] \* inv[0] + inMat[1] \* inv[4] + inMat[2] \* inv[8] + inMat[3] \* inv[12];

if (det == 0)

return false;

det = 1.0 / det;

for (i = 0; i < 16; i++)

outMat[i] = inv[i] \* det;

return true;

}

// fungsi untuk perkalian matriks 4x4 dengan 4x1

void DotMatrix(float inMat1[16], float inMat2[4], float outMat[4])

{

outMat[0] = inMat1[0] \* inMat2[0] + inMat1[1] \* inMat2[1] +

inMat1[2] \* inMat2[2] + inMat1[3] \* inMat2[3];

outMat[1] = inMat1[4] \* inMat2[0] + inMat1[5] \* inMat2[1] +

inMat1[6] \* inMat2[2] + inMat1[7] \* inMat2[3];

outMat[2] = inMat1[8] \* inMat2[0] + inMat1[9] \* inMat2[1] +

inMat1[10] \* inMat2[2] + inMat1[11] \* inMat2[3];

outMat[3] = inMat1[12] \* inMat2[0] + inMat1[13] \* inMat2[1] +

inMat1[14] \* inMat2[2] + inMat1[15] \* inMat2[3];

}

// fungsi untuk membuat kurva spline cubic dari 4 titik kontrol

// point1 sampai point4 = titik kontrol

// nPoint = jumlah titik interpolasi antara point1 sampai point4

void drawSplineCubic(Vec3 point1, Vec3 point2, Vec3 point3, Vec3 point4, int nPoint)

{

// hitung bobot jarak u di masing-masing titik

float utotal = (abs((int)(point2.X - point1.X)) + abs((int)(point3.X - point2.X)) + abs((int)(point4.X - point3.X)));

float u1 = 0;

float u2 = abs((int)(point2.X - point1.X)) / utotal;

float u3 = (abs((int)(point2.X - point1.X)) + abs((int)(point3.X - point2.X))) / utotal;

float u4 = 1;

// hitung inverse matriks dari koefisien u (lihat slide kuliah)

float inverseMat[16];

float coeffMat[16] = {

1.00f, 0.00f, 0.00f, 0.00f,

1.00f, u2, pow(u2, 2), pow(u2, 3),

1.00f, u3, pow(u3, 2), pow(u3, 3),

1.00f, 1.00f, 1.00f, 1.00f };

bool status = inverse(coeffMat, inverseMat);

// hitung koefisien cubic au^3 + bu^2 + cu + d

if (status == true)

{

float outMatX[4], outMatY[4], outMatZ[4];

float inMatX[4] = { point1.X, point2.X, point3.X, point4.X };

float inMatY[4] = { point1.Y, point2.Y, point3.Y, point4.Y };

float inMatZ[4] = { point1.Z, point2.Z, point3.Z, point4.Z };

DotMatrix(inverseMat, inMatX, outMatX);

DotMatrix(inverseMat, inMatY, outMatY);

DotMatrix(inverseMat, inMatZ, outMatZ);

// gambar kurva cubic spline dengan titik kontrol diatas

// hitung posisi y untuk setiap x di setiap point dengan persamaan diatas

for (int i=0; i<nPoint; i++)

{

float step = 1.0f / nPoint; // jeda setiap titik pd bobot u

// titik awal

float pX = point1.X, pY = point1.Y, pZ = point1.Z;

//

float u = 0.0f;

for (int i = 0; i < nPoint; i++)

{

// bentuk segment kurva cubic spline sebanyak nPoint

u = u + step;

glVertex3f(pX, pY, pZ); // gambar titik awal

// koordinat X pada kurva

pX = outMatX[3] \* pow(u, 3) + outMatX[2] \* pow(u, 2)

+ outMatX[1] \* u + outMatX[0];

// koordinat Y pada kurva

pY = outMatY[3] \* pow(u, 3) + outMatY[2] \* pow(u, 2)

+ outMatY[1] \* u + outMatY[0];

// koordinat Z pada kurva

pZ = outMatZ[3] \* pow(u, 3) + outMatZ[2] \* pow(u, 2)

+ outMatZ[1] \* u + outMatZ[0];

glVertex3f(pX, pY, pZ); // gambar titik akhir

}

}

}

}

// fungsi untuk membuat kurva spline bezier dari 4 titik kontrol

// point1 dan point4 = titik kontrol awal dan akhir

// point2 dan point3 = titik kontrol pembentuk kurva

// nPoint = jumlah titik interpolasi antara point1 sampai point4

void drawSplineBezier(Vec3 point1, Vec3 point2, Vec3 point3, Vec3 point4, int nPoint)

{

// hitung bobot jarak u di masing-masing titik

float utotal = (abs((int)(point2.X - point1.X)) + abs((int)(point3.X - point2.X)) + abs((int)(point4.X - point3.X)));

float u1 = 0;

float u2 = abs((int)(point2.X - point1.X)) / utotal;

float u3 = (abs((int)(point2.X - point1.X)) + abs((int)(point3.X - point2.X))) / utotal;

float u4 = 1;

// hitung inverse matriks dari koefisien u (lihat slide kuliah)

float inverseMat[16];

float coeffMat[16] = {

1.0000f, 0.0000f, 0.0000f, 0.0000f,

1.0000f, 1.0000f, 1.0000f, 1.0000f,

0.0000f, 1.0000f, 0.0000f, 0.0000f,

0.0000f, 1.0000f, 2.0000f, 3.0000f };

bool status = inverse(coeffMat, inverseMat);

// hitung koefisien

if (status == true)

{

float outMatX[4], outMatY[4], outMatZ[4];

float inMatX[4] = { point1.X, point4.X,

1.0f/(u2-u1)\*(point2.X - point1.X),

1.0f/(u4-u3)\*(point4.X - point3.X) };

float inMatY[4] = { point1.Y, point4.Y,

1.0f/(u2-u1)\*(point2.Y - point1.Y),

1.0f/(u4-u3)\*(point4.Y - point3.Y) };

float inMatZ[4] = { point1.Z, point4.Z,

1.0f/(u2-u1)\*(point2.Z - point1.Z),

1.0f/(u4-u3)\*(point4.Z - point3.Z) };

DotMatrix(inverseMat, inMatX, outMatX);

DotMatrix(inverseMat, inMatY, outMatY);

DotMatrix(inverseMat, inMatZ, outMatZ);

// gambar kurva spline dengan titik kontrol diatas

// hitung posisi y untuk setiap x di setiap point dengan persamaan diatas

for (int i=0; i<nPoint; i++)

{

float step = 1.0f / nPoint; // jeda setiap titik pd bobot u

// titik awal

float pX = point1.X, pY = point1.Y, pZ = point1.Z;

//

float u = 0.0f;

for (int i = 0; i < nPoint; i++)

{

// bentuk segment kurva spline sebanyak nPoint

u = u + step;

glVertex3f(pX, pY, pZ); // gambar titik awal

// koordinat X pada kurva

pX = outMatX[3] \* pow(u, 3) + outMatX[2] \* pow(u, 2)

+ outMatX[1] \* u + outMatX[0];

// koordinat Y pada kurva

pY = outMatY[3] \* pow(u, 3) + outMatY[2] \* pow(u, 2)

+ outMatY[1] \* u + outMatY[0];

// koordinat Z pada kurva

pZ = outMatZ[3] \* pow(u, 3) + outMatZ[2] \* pow(u, 2)

+ outMatZ[1] \* u + outMatZ[0];

glVertex3f(pX, pY, pZ); // gambar titik akhir

}

}

}

}

// fungsi untuk membuat kurva spline catmull-rom dari 4 titik kontrol

// point1 dan point4 = titik kontrol awal dan akhir

// point2 dan point3 = titik kontrol pembentuk kurva

// nPoint = jumlah titik interpolasi antara point1 sampai point4

void drawSplineCatmullRom(Vec3 point1, Vec3 point2, Vec3 point3, Vec3 point4, int nPoint)

{

// hitung bobot jarak u di masing-masing titik

float utotal = (abs((int)(point2.X - point1.X)) + abs((int)(point3.X - point2.X)) + abs((int)(point4.X - point3.X)));

float u1 = 0;

float u2 = abs((int)(point2.X - point1.X)) / utotal;

float u3 = (abs((int)(point2.X - point1.X)) + abs((int)(point3.X - point2.X))) / utotal;

float u4 = 1;

// hitung inverse matriks dari koefisien u (lihat slide kuliah)

float inverseMat[16];

float coeffMat[16] = {

1.0000f, 0.0000f, 0.0000f, 0.0000f,

1.0000f, 1.0000f, 1.0000f, 1.0000f,

0.0000f, 1.0000f, 0.0000f, 0.0000f,

0.0000f, 1.0000f, 2.0000f, 3.0000f };

bool status = inverse(coeffMat, inverseMat);

// hitung koefisien

if (status == true)

{

float outMatX[4], outMatY[4], outMatZ[4];

float inMatX[4] = { point1.X, point4.X,

1.0f/(u3-u1)\*(point3.X - point1.X),

1.0f/(u4-u2)\*(point4.X - point2.X) };

float inMatY[4] = { point1.Y, point4.Y,

1.0f/(u3-u1)\*(point3.Y - point1.Y),

1.0f/(u4-u2)\*(point4.Y - point2.Y) };

float inMatZ[4] = { point1.Z, point4.Z,

1.0f/(u3-u1)\*(point3.Z - point1.Z),

1.0f/(u4-u2)\*(point4.Z - point2.Z) };

DotMatrix(inverseMat, inMatX, outMatX);

DotMatrix(inverseMat, inMatY, outMatY);

DotMatrix(inverseMat, inMatZ, outMatZ);

// gambar kurva spline dengan titik kontrol diatas

// hitung posisi y untuk setiap x di setiap point dengan persamaan diatas

for (int i=0; i<nPoint; i++)

{

float step = 1.0f / nPoint; // jeda setiap titik pd bobot u

// titik awal

float pX = point1.X, pY = point1.Y, pZ = point1.Z;

//

float u = 0.0f;

for (int i = 0; i < nPoint; i++)

{

// bentuk segment kurva spline sebanyak nPoint

u = u + step;

glVertex3f(pX, pY, pZ); // gambar titik awal

// koordinat X pada kurva

pX = outMatX[3] \* pow(u, 3) + outMatX[2] \* pow(u, 2)

+ outMatX[1] \* u + outMatX[0];

// koordinat Y pada kurva

pY = outMatY[3] \* pow(u, 3) + outMatY[2] \* pow(u, 2)

+ outMatY[1] \* u + outMatY[0];

// koordinat Z pada kurva

pZ = outMatZ[3] \* pow(u, 3) + outMatZ[2] \* pow(u, 2)

+ outMatZ[1] \* u + outMatZ[0];

glVertex3f(pX, pY, pZ); // gambar titik akhir

}

}

}

}

// fungsi untuk menggambar obyek kubus

void drawObject()

{

// obyek bisa dimasukkan diantara glPushMatrix() dan glPopMatrix()

// fungsinya agar obyek tidak terpengaruh atau mempengaruhi obyek lain

// saat diwarnai, ditransformasi dan sebagainya

glPushMatrix();

// operasi transformasi rotasi obyek ke arah kanan-kiri

glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);

glPushMatrix();

// operasi transformasi rotasi obyek ke arah atas-bawah

glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

// membuat 4 titik kontrol kurva

Vec3 point1 = Vec3(100.0f, -90.0f, 0.0f);

Vec3 point2 = Vec3( -80.0f, 50.0f, 0.0f);

Vec3 point3 = Vec3( 50.0f, -20.0f, 0.0f);

Vec3 point4 = Vec3(-100.0f, -80.0f, 0.0f);

// tandai setiap titik kontrol kurva dengan warna

markPoint(point1, Vec3(0.0f, 1.0f, 0.0f), 5.0f);

markPoint(point2, Vec3(1.0f, 0.0f, 0.0f), 5.0f);

markPoint(point3, Vec3(1.0f, 1.0f, 0.0f), 5.0f);

markPoint(point4, Vec3(0.0f, 0.0f, 1.0f), 5.0f);

// mengatur warna obyek menjadi berwarna putih

glColor3f(0.3f, 0.3f, 0.0f);

glBegin(GL\_LINES);

// membuat kurva spline cubic dari titik kontrol diatas

drawSplineCubic(point1, point2, point3, point4, 30);

// drawSplineBezier(point1, point2, point3, point4, 30);

// drawSplineCatmullRom(point1, point2, point3, point4, 30);

glEnd();

glPopMatrix();

glPopMatrix();

}

// taruh semua obyek yang akan digambar di fungsi display()

void display()

{

// bersihkan dan reset layar dan buffer

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

// posisikan kamera pandang

// dalam hal ini sumbu Y ada diatas dan posisi kamera pandang di (posX, posY, posZ)

gluLookAt(posX, posY, posZ, posX + rotX, posY + rotY, posZ + rotZ, 0.0f, 1.0f, 0.0f);

// panggil fungsi untuk menggambar obyek

drawObject();

// tampilkan obyek ke layar

// gunakan glFlush() bila memakai single buffer

// gunakan glutSwapBuffers() bila memakai double buffer

glutSwapBuffers();

}

// inisialisasikan variabel, pencahayaan, tekstur dan pengaturan kamera pandang di fungsi init()

void init(void)

{

// inisialisasi warna latar belakang layar dalam hal ini warna putih (1.0, 1.0, 1.0, 0.0)

glClearColor(0.0, 0.0, 0.0, 0.0);

glEnable(GL\_DEPTH\_TEST); // mengaktifkan depth buffer

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

//gluPerspective(45.0, 1.0, 1.0, 100.0); // set proyeksi ke perspektif

glOrtho((GLfloat)-SCREEN\_WIDTH / 2, (GLfloat)SCREEN\_WIDTH / 2,

(GLfloat)-SCREEN\_HEIGHT / 2, (GLfloat)SCREEN\_HEIGHT / 2, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// inisialisasi kamera pandang

gluLookAt(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

}

// fungsi ini digunakan bila layar akan diresize (default)

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

//gluPerspective(45, (GLfloat)w / (GLfloat)h, 1.0, 100.0);

glOrtho((GLfloat)-SCREEN\_WIDTH / 2, (GLfloat)SCREEN\_WIDTH / 2,

(GLfloat)-SCREEN\_HEIGHT / 2, (GLfloat)SCREEN\_HEIGHT / 2, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

}

// fungsi untuk mengatur masukan dari keyboard

// untuk arah kiri, kanan, atas, bawah, PgUp, dan PgDn

void keyboard(int key, int x, int y)

{

float fraction = 0.1f;

switch (key)

{

// masukkan perintah disini bila tombol kiri ditekan

case GLUT\_KEY\_LEFT:

// dalam hal ini perintah rotasi obyek ke kiri sebanyak 1 derajat

objectAngleY -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol kanan ditekan

case GLUT\_KEY\_RIGHT:

// dalam hal ini perintah rotasi obyek ke kanan sebanyak 1 derajat

objectAngleY += 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol atas ditekan

case GLUT\_KEY\_UP:

// dalam hal ini perintah rotasi obyek ke atas sebanyak 1 derajat

objectAngleX -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol bawah ditekan

case GLUT\_KEY\_DOWN:

// dalam hal ini perintah rotasi obyek ke bawah sebanyak 1 derajat

objectAngleX += 1.0f;

glutPostRedisplay(); // update obyek

break;

// zoom in

case GLUT\_KEY\_PAGE\_UP:

// masukkan perintah disini bila tombol PgUp ditekan

posX += rotX \* fraction;

posZ += rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

// zoom out

case GLUT\_KEY\_PAGE\_DOWN:

// masukkan perintah disini bila tombol PgDn ditekan

posX -= rotX \* fraction;

posZ -= rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

}

}

// timer untuk animasi (gunakan bila perlu)

void timer(int value)

{

glutPostRedisplay();

glutTimerFunc(55, timer, 0);

}

// program utama

int main(int argc, char\*\* argv)

{

// inisialisasi jendela OpenGL

// GLUT\_SINGLE berarti memakai single buffer

// GLUT\_DOUBLE berarti memakai double buffer

// GLUT\_RGB berarti mode tampilan yang dipakai RGB

// GLUT\_RGBA berarti mode tampilan yang dipakai RGBA

// GLUT\_DEPTH berarti memakai depth buffer

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA | GLUT\_DEPTH);

// set ukuran jendela tampilan

glutInitWindowSize(480, 480); // besarnya jendela dalam piksel dalam hal ini 300x300

glutInitWindowPosition(100, 100); // posisi jendela dilayar komputer dalam piksel

// judul jendela (wajib diubah dengan informasi NAMA / NIM - JUDUL PRAKTIKUM masing-masing)

glutCreateWindow("SIRLY ZIADATUL MUSTAFIDAH / 2200018345 - KURVA SPLINE");

// panggil fungsi init untuk inisialisasi awal

init();

// event handler untuk display, reshape dan keyboard

glutDisplayFunc(display); // display

glutReshapeFunc(reshape); // reshape

glutSpecialFunc(keyboard); // keyboard

//glutTimerFunc(0, timer, 0); // aktifkan timer bila perlu

// looping

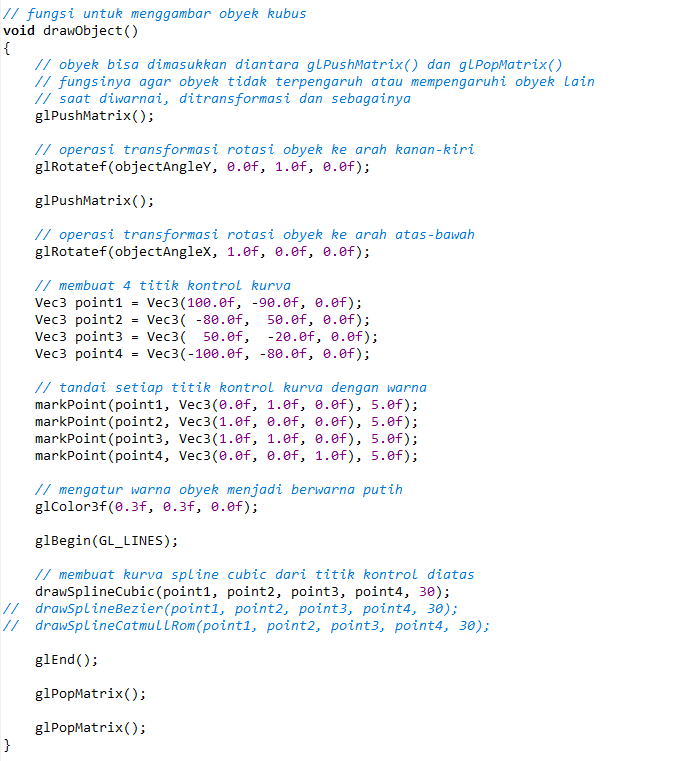
glutMainLoop();

return 0;

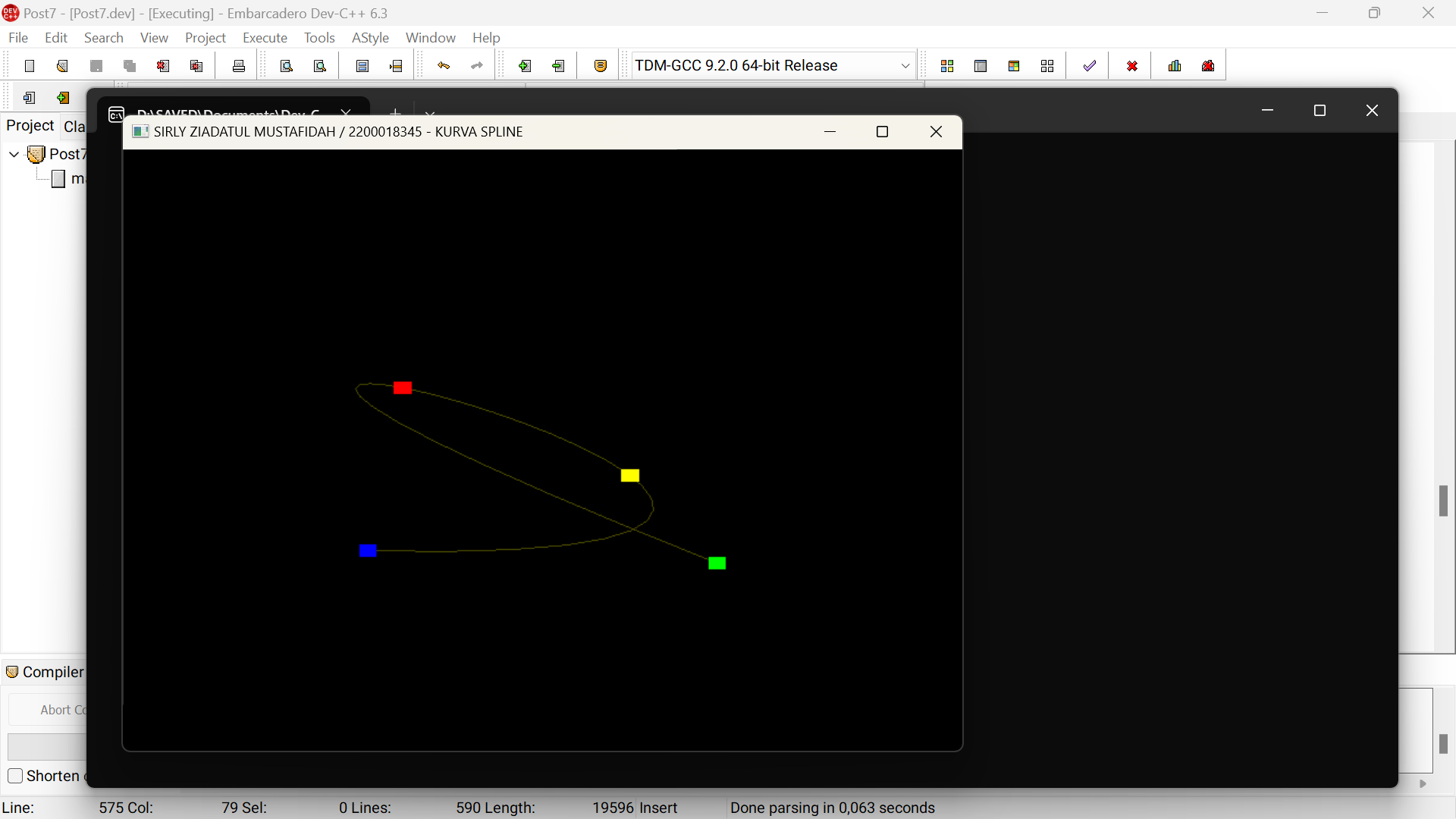
}

Penjelasan





Output



Gambar 7. 2

# BAB VIII. TEKNIK REPRESENTASI PERMUKAAN

1. Pretest 8
2. Sebutkan teknik representasi permukaan yang anda ketahui!

* Mesh polygon
* NURBS
* Subdivision surfaces
* Triangulating polygon

1. Jelaskan setiap teknik representaasi permukaan yang anda sebutkan di soal nomor 1!

* Mesh polygon adalah teknik yang permukaan obyek 3D direpresentasikan menggunakan sekumpulan poligon
* NURBS adalah teknik yang menggunakan kurva dan permukaan matematos yang fleksibel dan presisi
* Subdivisi surfaces adalah teknik yang menggunakan mesh polygon dasar yang kemudian dihaluskan melalui iterasi
* Triangulating polygon adalah teknik yang membagi permukaan atau mesh polygon menjadi beberapa mesh segitiga

1. Langkah Praktikum 8

Source Code

// praktikum 09 grafika komputer

// adhi prahara. 2018

// deklarasikan semua header disini

#include <windows.h>

#include "GL/glut.h"

// inisialisasi variabel untuk transformasi seperti translasi, rotasi atau scaling

float angle = 0.0f; // sudut transformasi kamera

float posX = 0.0f, rotX = 0.0f; // posisi kamera di sumbu X

float posY = 0.0f, rotY = 0.0f; // posisi kamera di sumbu Y

float posZ = 5.0f, rotZ = -1.0f; // posisi kamera di sumbu Z

float objectAngleX = 0.0f; // sudut tranformasi obyek di sumbu X

float objectAngleY = 0.0f; // sudut tranformasi obyek di sumbu Y

float objectAngleZ = 0.0f; // sudut tranformasi obyek di sumbu Z

// container untuk membuat tipe data 3D (X, Y, Z)

struct Vec3

{

float X; float Y; float Z;

Vec3(float x, float y, float z) { X = x; Y = y; Z = z; }

//

Vec3() { }

~Vec3() { }

};

// kurva

Vec3 controlPoint[4][4];

bool showPoints = false;

GLfloat mat\_diffuse[] = { 0.7, 0.7, 0.7, 1.0 };

GLfloat mat\_specular[] = { 1.0, 1.0, 1.0, 1.0 };

GLfloat mat\_shininess[] = { 100.0 };

GLUnurbsObj \*theNurb;

// fungsi untuk menggambar obyek

void drawObject()

{

glPushMatrix();

// operasi transformasi rotasi obyek ke arah kanan-kiri

glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);

glPushMatrix();

// operasi transformasi rotasi obyek ke arah atas-bawah

glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

glScalef(0.5, 0.5, 0.5);

// set warna obyek ke warna hijau (0.0f, 1.0f, 0.0f)

glColor3f(0.0f, 1.0f, 0.0f);

//

GLfloat knots[8] = { 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0 };

gluBeginSurface(theNurb);

gluNurbsSurface(theNurb, 8, knots, 8, knots, 4 \* 3, 3, &controlPoint[0][0].X, 4, 4, GL\_MAP2\_VERTEX\_3);

gluEndSurface(theNurb);

if (showPoints)

{

glPointSize(5.0);

glDisable(GL\_LIGHTING);

glColor3f(1.0, 1.0, 0.0);

glBegin(GL\_POINTS);

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 4; j++)

{

glVertex3f(controlPoint[i][j].X, controlPoint[i][j].Y,

controlPoint[i][j].Z);

}

}

glEnd();

glEnable(GL\_LIGHTING);

}

glPopMatrix();

glPopMatrix();

}

// taruh semua obyek yang akan digambar di fungsi display()

void display()

{

// bersihkan dan reset layar dan buffer

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

// posisikan kamera pandang

// dalam hal ini sumbu Y ada diatas dan posisi kamera pandang di (posX, posY, posZ)

gluLookAt(posX, posY, posZ, posX + rotX, posY + rotY, posZ + rotZ, 0.0f, 1.0f, 0.0f);

// panggil fungsi untuk menggambar obyek

drawObject();

// tampilkan obyek ke layar

// gunakan glFlush() bila memakai single buffer

// gunakan glutSwapBuffers() bila memakai double buffer

glutSwapBuffers();

}

// inisialisasikan variabel, pencahayaan, tekstur dan pengaturan kamera pandang di fungsi init()

// inisialisasi

void init(void)

{

// inisialisasi warna latar belakang

glClearColor(1.0, 1.0, 1.0, 0.0);

glEnable(GL\_DEPTH\_TEST); // mengaktifkan depth buffer

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45.0, 1.0, 1.0, 100.0);// set proyeksi ke perspektif

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// inisialisasi kamera pandang

gluLookAt(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

int u, v;

for (u = 0; u < 4; u++)

{

for (v = 0; v < 4; v++)

{

controlPoint[u][v].X = 2.0\*((GLfloat)u - 1.5);

controlPoint[u][v].Y = 2.0\*((GLfloat)v - 1.5);

if ((u == 1 || u == 2) || (v == 1 || v == 2))

controlPoint[u][v].Z = 3.0;

else

controlPoint[u][v].Z = -3.0;

}

}

glClearColor(0.0, 0.0, 0.0, 0.0);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, mat\_diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat\_specular);

glMaterialfv(GL\_FRONT, GL\_SHININESS, mat\_shininess);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glEnable(GL\_DEPTH\_TEST);

glEnable(GL\_AUTO\_NORMAL);

glEnable(GL\_NORMALIZE);

theNurb = gluNewNurbsRenderer();

gluNurbsProperty(theNurb, GLU\_SAMPLING\_TOLERANCE, 25.0);

gluNurbsProperty(theNurb, GLU\_DISPLAY\_MODE, GLU\_FILL);

}

// fungsi ini digunakan bila layar akan diresize (default)

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45, (GLfloat)w / (GLfloat)h, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

}

// fungsi untuk mengatur masukan dari keyboard

// untuk arah kiri, kanan, atas, bawah, PgUp, dan PgDn

void keyboard(int key, int x, int y)

{

float fraction = 0.1f;

switch (key)

{

// masukkan perintah disini bila tombol kiri ditekan

case GLUT\_KEY\_LEFT:

// dalam hal ini perintah rotasi obyek ke kiri sebanyak 1 derajat

objectAngleY -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol kanan ditekan

case GLUT\_KEY\_RIGHT:

// dalam hal ini perintah rotasi obyek ke kanan sebanyak 1 derajat

objectAngleY += 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol atas ditekan

case GLUT\_KEY\_UP:

// dalam hal ini perintah rotasi obyek ke atas sebanyak 1 derajat

objectAngleX -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol bawah ditekan

case GLUT\_KEY\_DOWN:

// dalam hal ini perintah rotasi obyek ke bawah sebanyak 1 derajat

objectAngleX += 1.0f;

glutPostRedisplay(); // update obyek

break;

// zoom in

case GLUT\_KEY\_PAGE\_UP:

// masukkan perintah disini bila tombol PgUp ditekan

posX += rotX \* fraction;

posZ += rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

// zoom out

case GLUT\_KEY\_PAGE\_DOWN:

// masukkan perintah disini bila tombol PgDn ditekan

posX -= rotX \* fraction;

posZ -= rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

// tampilkan point

case GLUT\_KEY\_F1:

showPoints = !showPoints;

break;

}

}

// timer untuk animasi (gunakan bila perlu)

void timer(int value)

{

glutPostRedisplay();

glutTimerFunc(55, timer, 0);

}

// program utama

int main(int argc, char\*\* argv)

{

// inisialisasi jendela OpenGL

// GLUT\_SINGLE berarti memakai single buffer

// GLUT\_DOUBLE berarti memakai double buffer

// GLUT\_RGB berarti mode tampilan yang dipakai RGB

// GLUT\_RGBA berarti mode tampilan yang dipakai RGBA

// GLUT\_DEPTH berarti memakai depth buffer

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA | GLUT\_DEPTH);

// set ukuran jendela tampilan

glutInitWindowSize(480, 480); // besarnya jendela dalam piksel dalam hal ini 300x300

glutInitWindowPosition(100, 100); // posisi jendela dilayar komputer dalam piksel

// judul jendela (wajib diubah dengan informasi NAMA / NIM - JUDUL PRAKTIKUM masing-masing)

glutCreateWindow("SIRLY ZIADATUL MUSTAFIDAH / 2200018345 - TEKNIK REPRESENTASI PERMUKAAN");

// panggil fungsi init untuk inisialisasi awal

init();

// event handler untuk display, reshape dan keyboard

glutDisplayFunc(display); // display

glutReshapeFunc(reshape); // reshape

glutSpecialFunc(keyboard); // keyboard

//glutTimerFunc(0, timer, 0); // aktifkan timer bila perlu

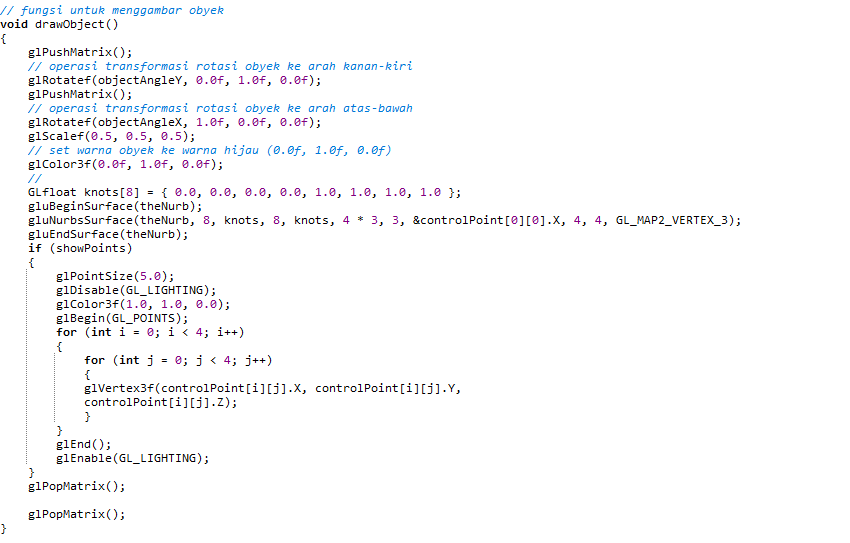
// looping

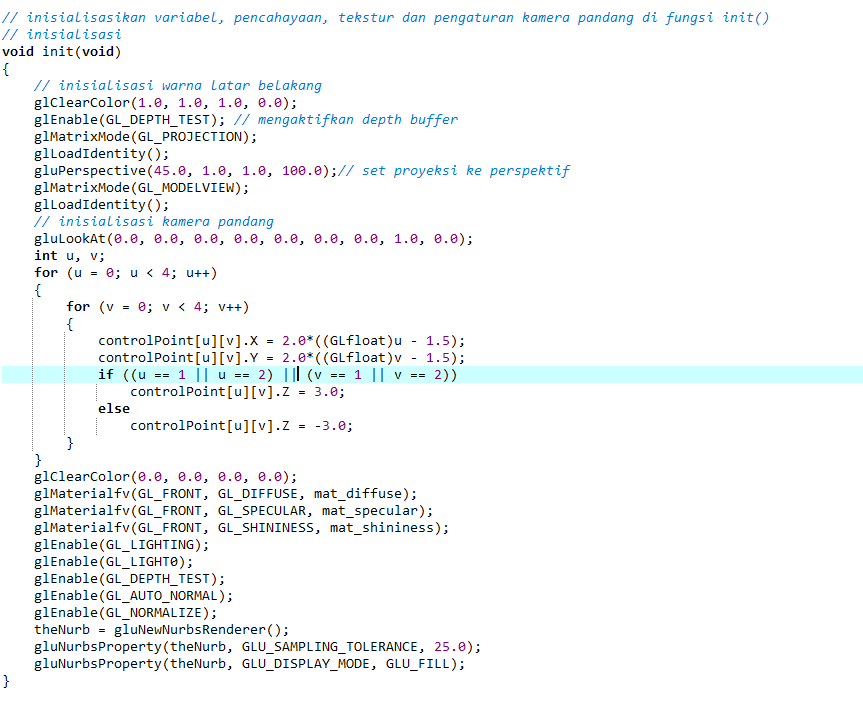
glutMainLoop();

return 0;

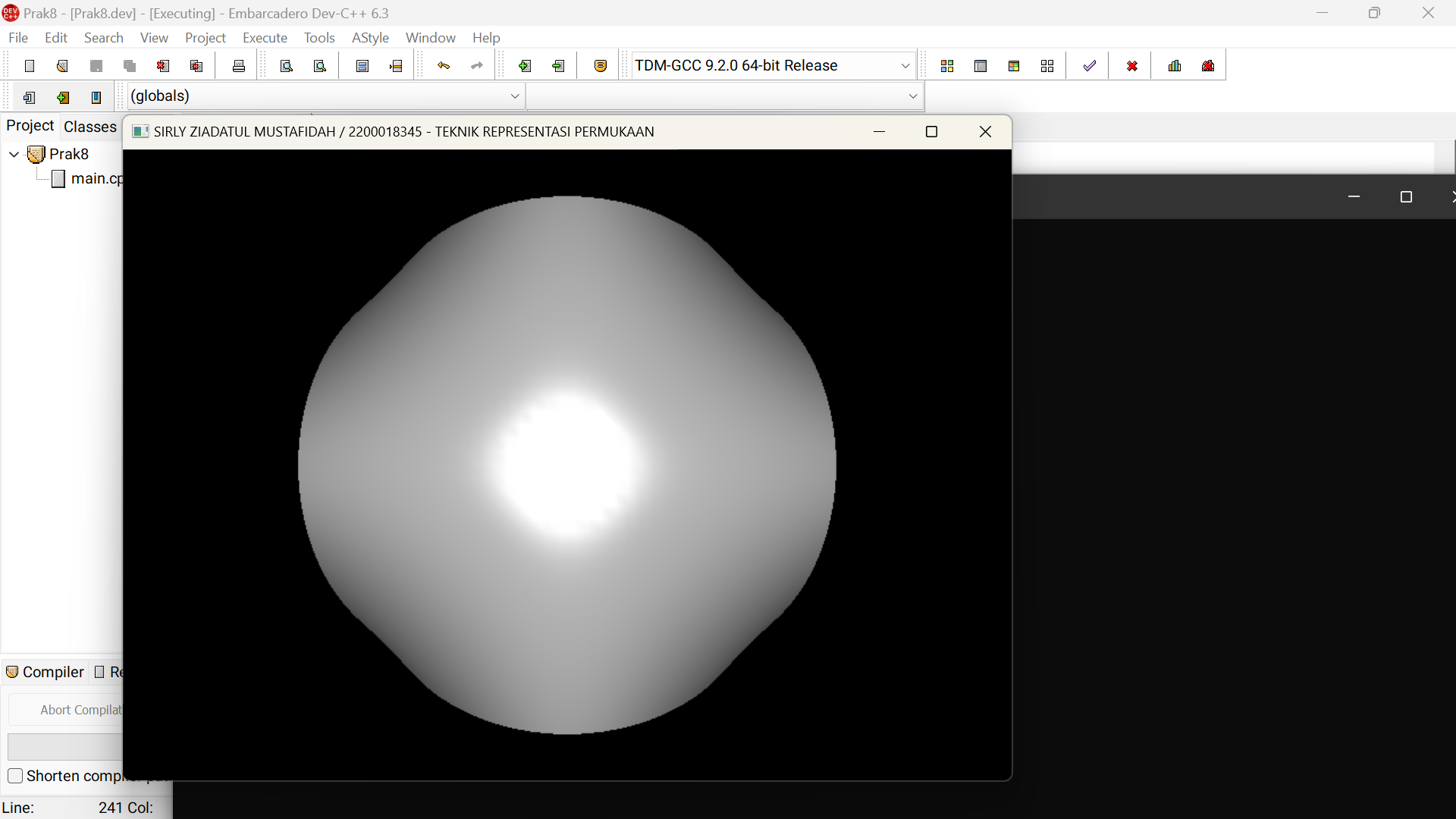
}

Penjelasan

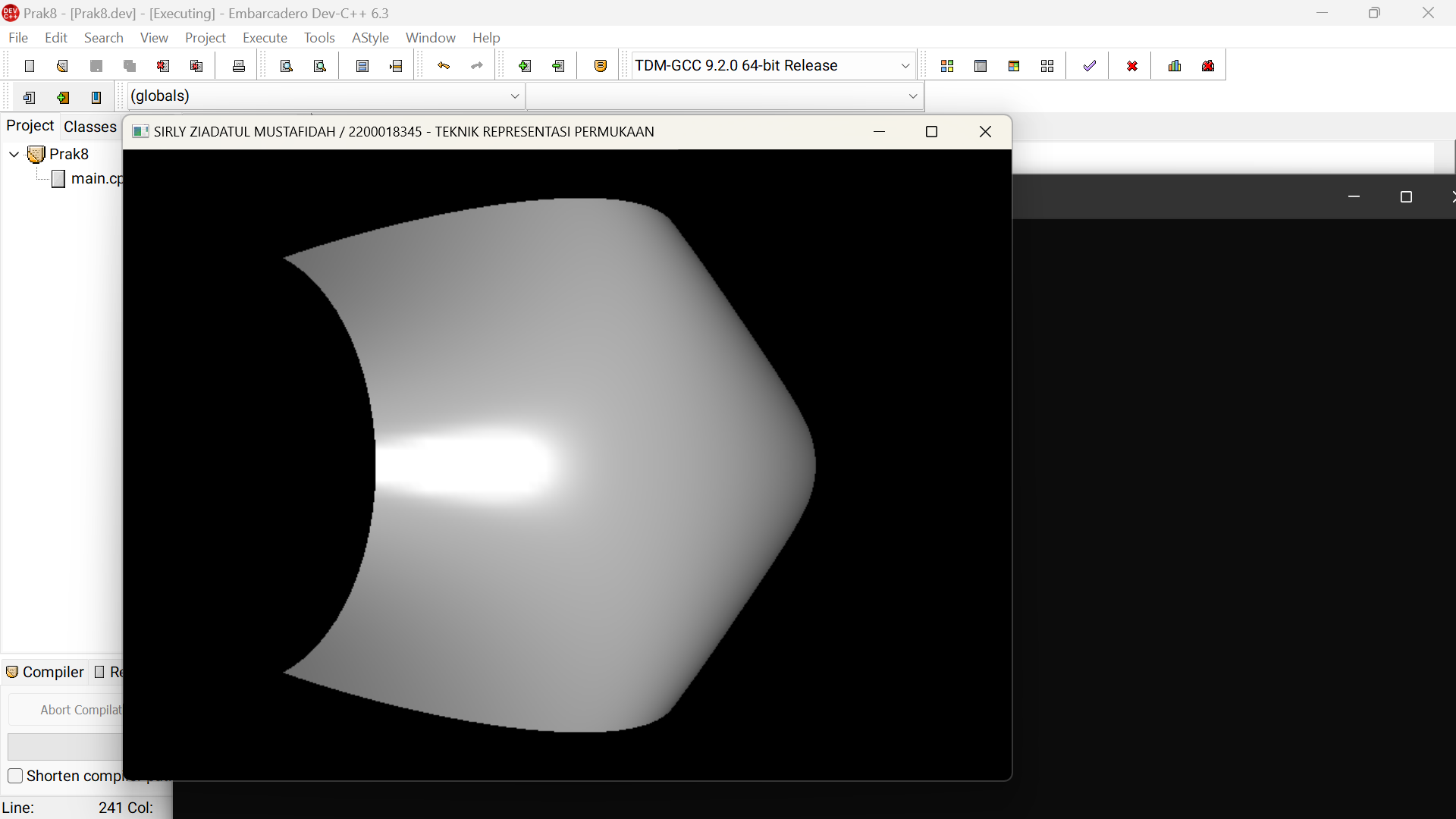




Output



Gambar 8. 1



Gambar 8. 2

1. Posttest 8

Source Code

// praktikum 09 grafika komputer

// adhi prahara. 2018

// deklarasikan semua header disini

#include <windows.h>

#include "GL/glut.h"

// inisialisasi variabel untuk transformasi seperti translasi, rotasi atau scaling

float angle = 0.0f; // sudut transformasi kamera

float posX = 0.0f, rotX = 0.0f; // posisi kamera di sumbu X

float posY = 0.0f, rotY = 0.0f; // posisi kamera di sumbu Y

float posZ = 5.0f, rotZ = -1.0f; // posisi kamera di sumbu Z

float objectAngleX = 0.0f; // sudut tranformasi obyek di sumbu X

float objectAngleY = 0.0f; // sudut tranformasi obyek di sumbu Y

float objectAngleZ = 0.0f; // sudut tranformasi obyek di sumbu Z

// container untuk membuat tipe data 3D (X, Y, Z)

struct Vec3

{

float X; float Y; float Z;

Vec3(float x, float y, float z) { X = x; Y = y; Z = z; }

//

Vec3() { }

~Vec3() { }

};

// kurva

Vec3 controlPoint[4][4];

bool showPoints = false;

GLfloat mat\_diffuse[] = { 0.7, 0.7, 0.7, 1.0 };

GLfloat mat\_specular[] = { 1.0, 1.0, 1.0, 1.0 };

GLfloat mat\_shininess[] = { 100.0 };

GLUnurbsObj \*theNurb;

// fungsi untuk menggambar obyek

void drawObject()

{

glPushMatrix();

// operasi transformasi rotasi obyek ke arah kanan-kiri

glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);

glPushMatrix();

// operasi transformasi rotasi obyek ke arah atas-bawah

glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

glScalef(0.5, 0.5, 0.5);

// set warna obyek ke warna hijau (0.0f, 1.0f, 0.0f)

glColor3f(0.0f, 1.0f, 0.0f);

//

GLfloat knots[8] = { 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0 };

gluBeginSurface(theNurb);

gluNurbsSurface(theNurb, 8, knots, 8, knots, 4 \* 3, 3, &controlPoint[0][0].X, 4, 4, GL\_MAP2\_VERTEX\_3);

gluEndSurface(theNurb);

if (showPoints)

{

glPointSize(5.0);

glDisable(GL\_LIGHTING);

glColor3f(1.0, 1.0, 0.0);

glBegin(GL\_POINTS);

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 4; j++)

{

glVertex3f(controlPoint[i][j].X, controlPoint[i][j].Y,

controlPoint[i][j].Z);

}

}

glEnd();

glEnable(GL\_LIGHTING);

}

glPopMatrix();

glPopMatrix();

}

// taruh semua obyek yang akan digambar di fungsi display()

void display()

{

// bersihkan dan reset layar dan buffer

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

// posisikan kamera pandang

// dalam hal ini sumbu Y ada diatas dan posisi kamera pandang di (posX, posY, posZ)

gluLookAt(posX, posY, posZ, posX + rotX, posY + rotY, posZ + rotZ, 0.0f, 1.0f, 0.0f);

// panggil fungsi untuk menggambar obyek

drawObject();

// tampilkan obyek ke layar

// gunakan glFlush() bila memakai single buffer

// gunakan glutSwapBuffers() bila memakai double buffer

glutSwapBuffers();

}

// inisialisasikan variabel, pencahayaan, tekstur dan pengaturan kamera pandang di fungsi init()

// inisialisasi

void init(void)

{

// inisialisasi warna latar belakang

glClearColor(1.0, 1.0, 1.0, 0.0);

glEnable(GL\_DEPTH\_TEST); // mengaktifkan depth buffer

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45.0, 1.0, 1.0, 100.0);// set proyeksi ke perspektif

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// inisialisasi kamera pandang

gluLookAt(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

int u, v;

for (u = 0; u < 4; u++)

{

for (v = 0; v < 4; v++)

{

controlPoint[u][v].X = 2.0\*((GLfloat)u - 1.5);

controlPoint[u][v].Y = 2.0\*((GLfloat)v - 1.5);

if ((u == 1 || u == 2)) // || (v == 1 || v == 2)

controlPoint[u][v].Z = 3.0;

else

controlPoint[u][v].Z = -3.0;

}

}

glClearColor(0.0, 0.0, 0.0, 0.0);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, mat\_diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat\_specular);

glMaterialfv(GL\_FRONT, GL\_SHININESS, mat\_shininess);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glEnable(GL\_DEPTH\_TEST);

glEnable(GL\_AUTO\_NORMAL);

glEnable(GL\_NORMALIZE);

theNurb = gluNewNurbsRenderer();

gluNurbsProperty(theNurb, GLU\_SAMPLING\_TOLERANCE, 25.0);

gluNurbsProperty(theNurb, GLU\_DISPLAY\_MODE, GLU\_FILL);

}

// fungsi ini digunakan bila layar akan diresize (default)

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45, (GLfloat)w / (GLfloat)h, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

}

// fungsi untuk mengatur masukan dari keyboard

// untuk arah kiri, kanan, atas, bawah, PgUp, dan PgDn

void keyboard(int key, int x, int y)

{

float fraction = 0.1f;

switch (key)

{

// masukkan perintah disini bila tombol kiri ditekan

case GLUT\_KEY\_LEFT:

// dalam hal ini perintah rotasi obyek ke kiri sebanyak 1 derajat

objectAngleY -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol kanan ditekan

case GLUT\_KEY\_RIGHT:

// dalam hal ini perintah rotasi obyek ke kanan sebanyak 1 derajat

objectAngleY += 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol atas ditekan

case GLUT\_KEY\_UP:

// dalam hal ini perintah rotasi obyek ke atas sebanyak 1 derajat

objectAngleX -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol bawah ditekan

case GLUT\_KEY\_DOWN:

// dalam hal ini perintah rotasi obyek ke bawah sebanyak 1 derajat

objectAngleX += 1.0f;

glutPostRedisplay(); // update obyek

break;

// zoom in

case GLUT\_KEY\_PAGE\_UP:

// masukkan perintah disini bila tombol PgUp ditekan

posX += rotX \* fraction;

posZ += rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

// zoom out

case GLUT\_KEY\_PAGE\_DOWN:

// masukkan perintah disini bila tombol PgDn ditekan

posX -= rotX \* fraction;

posZ -= rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

// tampilkan point

case GLUT\_KEY\_F1:

showPoints = !showPoints;

break;

}

}

// timer untuk animasi (gunakan bila perlu)

void timer(int value)

{

glutPostRedisplay();

glutTimerFunc(55, timer, 0);

}

// program utama

int main(int argc, char\*\* argv)

{

// inisialisasi jendela OpenGL

// GLUT\_SINGLE berarti memakai single buffer

// GLUT\_DOUBLE berarti memakai double buffer

// GLUT\_RGB berarti mode tampilan yang dipakai RGB

// GLUT\_RGBA berarti mode tampilan yang dipakai RGBA

// GLUT\_DEPTH berarti memakai depth buffer

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA | GLUT\_DEPTH);

// set ukuran jendela tampilan

glutInitWindowSize(480, 480); // besarnya jendela dalam piksel dalam hal ini 300x300

glutInitWindowPosition(100, 100); // posisi jendela dilayar komputer dalam piksel

// judul jendela (wajib diubah dengan informasi NAMA / NIM - JUDUL PRAKTIKUM masing-masing)

glutCreateWindow("SIRLY ZIADATUL MUSTAFIDAH / 2200018345 - TEKNIK REPRESENTASI PERMUKAAN");

// panggil fungsi init untuk inisialisasi awal

init();

// event handler untuk display, reshape dan keyboard

glutDisplayFunc(display); // display

glutReshapeFunc(reshape); // reshape

glutSpecialFunc(keyboard); // keyboard

//glutTimerFunc(0, timer, 0); // aktifkan timer bila perlu

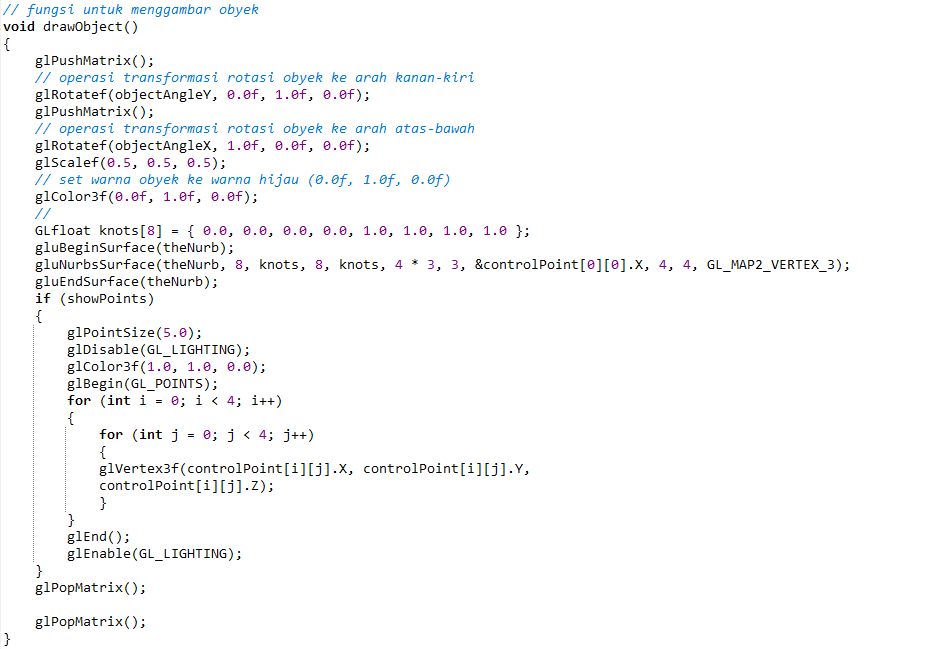
// looping

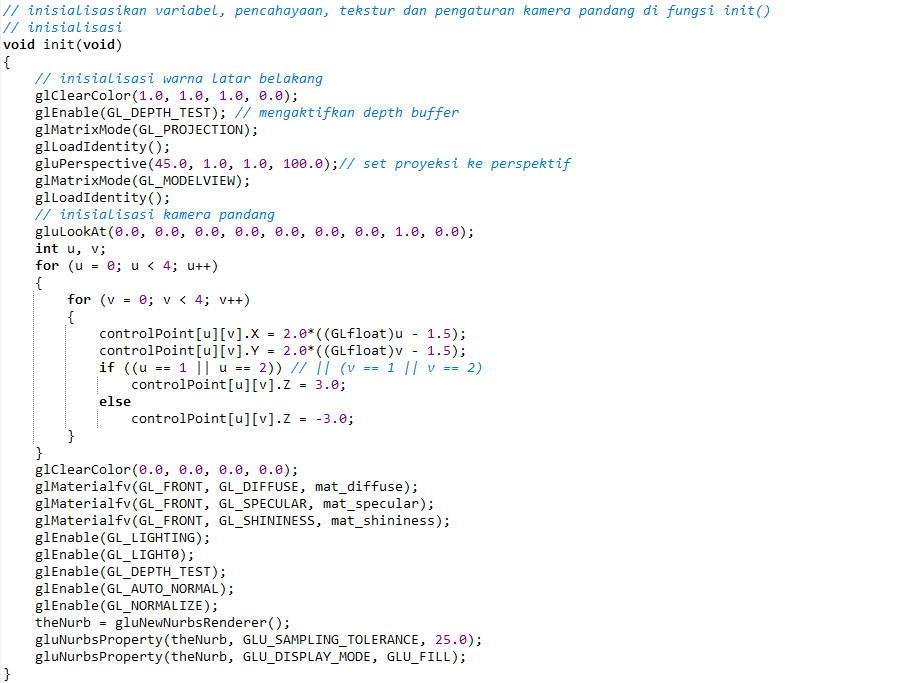
glutMainLoop();

return 0;

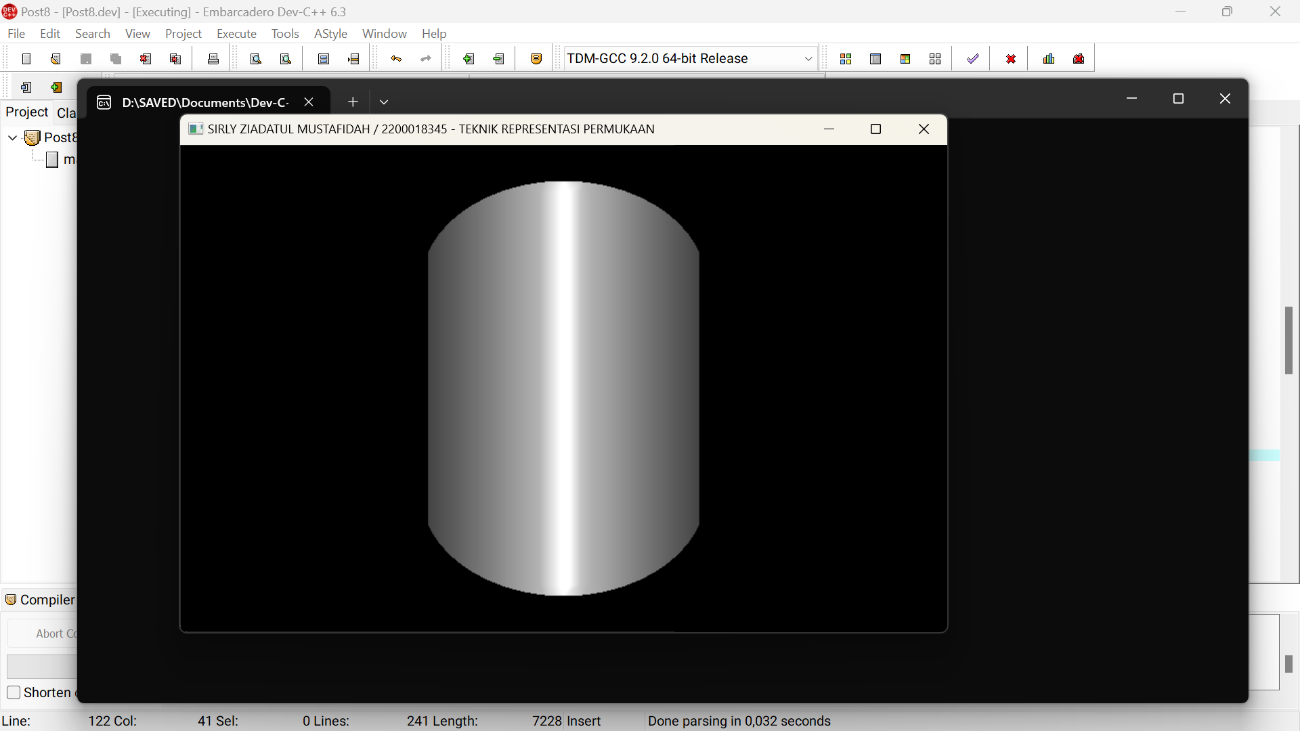
}

Penjelasan





Output



Gambar 8. 3

# BAB IX. TEKNIK PEMODELAN OBYEK 3D

1. Pretest 9
2. Sebutkan teknik pemodelan 3D yang anda ketahui!

* Sweep representation
* Surface of revolution
* Boolean modeling

1. Jelaskan setiap teknik pemodelan 3D yang anda sebutkan di soal nomor 1!

* Sweep representation adalah metode yang digunakan untuk membuat obyek 3D yang simetris
* Surface of revolution adalah metode yang memodelkan obyek dengan merotasikan garis atau kurva terhadap suatu sumbu
* Boolean modeling adalah metode yang membuat model dari kombinasi bentuk-bentuk dasar, biasanya digunakan pada Coustructive Solid Geometry (CSG)

1. Langkah Praktikum 9

Source Code

// deklarasikan semua header disini

#include <windows.h>

#include <GL/glut.h>

#include "COLOR.h"

#include "CUBE\_GRID.h"

#include "METABALL.h"

#include "VECTOR3D.h"

#include "TIMER.h"

// inisialisasi variabel untuk transformasi seperti translasi, rotasi atau scaling

float angle = 0.0f; // sudut transformasi kamera

float posX = 0.0f, rotX = 0.0f; // posisi kamera di sumbu X

float posY = 0.0f, rotY = 0.0f; // posisi kamera di sumbu Y

float posZ = 5.0f, rotZ = -1.0f; // posisi kamera di sumbu Z

float objectAngleX = 0.0f; // sudut tranformasi obyek di sumbu X

float objectAngleY = 0.0f; // sudut tranformasi obyek di sumbu Y

float objectAngleZ = 0.0f; // sudut tranformasi obyek di sumbu Z

const int minGridSize = 10;

int gridSize = 40;

float threshold = 1.0f;

const int numDiffuseColors = 10;

COLOR diffuseColors[numDiffuseColors];

int currentDiffuseColor = 0;

COLOR backgroundColor(0.773f, 0.773f, 0.773f, 1.0f);

CUBE\_GRID cubeGrid;

const int numMetaballs = 10;

METABALL metaballs[numMetaballs];

TIMER timers;

//set up lighting

float shininess = 32.0f;

float ambient[] = { 0.0f, 0.0f, 0.2f, 1.0f };

float position[] = { -1.0f, 1.0f, 1.0f, 0.0f };

float specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };

// fungsi ini digunakan untuk menggambar obyek

void drawObject()

{

glPushMatrix();

// glRotatef(objectAngle, objectRotation.X, objectRotation.Y,

// objectRotation.Z);

glRotatef(objectAngleY, 0.0, 1.0f, 0.0f);

// dinormalisasi dulu

glEnable(GL\_NORMALIZE);

glEnable(GL\_CULL\_FACE);

glLightfv(GL\_LIGHT1, GL\_AMBIENT, ambient);

glLightfv(GL\_LIGHT1, GL\_DIFFUSE, diffuseColors[currentDiffuseColor]);

glLightfv(GL\_LIGHT1, GL\_POSITION, position);

glLightfv(GL\_LIGHT1, GL\_SPECULAR, specular);

glEnable(GL\_LIGHT1);

// set pencahayaan

glMaterialfv(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE, white);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, white);

glMaterialfv(GL\_FRONT, GL\_SHININESS, &shininess);

// update posisi metaball

float c = 2.0f\*(float)cos(timers.GetTime() / 600);

metaballs[0].position.x = -4.0f\*(float)cos(timers.GetTime()/700) - c;

metaballs[0].position.y = 4.0f\*(float)sin(timers.GetTime()/600) - c;

metaballs[1].position.x = 5.0f\*(float)sin(timers.GetTime()/400) + c;

metaballs[1].position.y = 5.0f\*(float)cos(timers.GetTime()/400) - c;

metaballs[2].position.x = -5.0f\*(float)cos(timers.GetTime()/400) -

0.2f\*(float)sin(timers.GetTime() / 600);

metaballs[2].position.y = 5.0f\*(float)sin(timers.GetTime()/500) -

0.2f\*(float)sin(timers.GetTime() / 400);

// bersihkan layar

for (int i = 0; i<cubeGrid.numVertices; i++)

{

cubeGrid.vertices[i].value = 0.0f;

cubeGrid.vertices[i].normal.LoadZero();

}

// hitung bidang skalar disetiap titik

VECTOR3D ballToPoint;

float squaredRadius;

VECTOR3D ballPosition;

float normalScale;

for (int i = 0; i<numMetaballs; i++)

{

squaredRadius = metaballs[i].squaredRadius;

ballPosition = metaballs[i].position;

for (int j = 0; j<cubeGrid.numVertices; j++)

{

ballToPoint.x = cubeGrid.vertices[j].position.x -

ballPosition.x;

ballToPoint.y = cubeGrid.vertices[j].position.y -

ballPosition.y;

ballToPoint.z = cubeGrid.vertices[j].position.z -

ballPosition.z;

// hitung jarak bola ke titik

float squaredDistance = ballToPoint.x\*ballToPoint.x +

ballToPoint.y\*ballToPoint.y + ballToPoint.z\*ballToPoint.z;

if (squaredDistance == 0.0f)

squaredDistance = 0.0001f;

// value = r^2/d^2

cubeGrid.vertices[j].value += squaredRadius / squaredDistance;

// normal = (r^2 \* v)/d^4

normalScale = squaredRadius /(squaredDistance\*squaredDistance);

cubeGrid.vertices[j].normal.x += ballToPoint.x\*normalScale;

cubeGrid.vertices[j].normal.y += ballToPoint.y\*normalScale;

cubeGrid.vertices[j].normal.z += ballToPoint.z\*normalScale;

}

}

glPopMatrix();

}

/// taruh semua fungsi obyek yang akan digambar di fungsi display()

void display()

{

// bersihkan dan reset layar dan buffer

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

gluLookAt(posX, posY, posZ, posX + rotX, posY + rotY, posZ + rotZ, 0.0f, 1.0f, 0.0f);

glShadeModel(GL\_SMOOTH);

// panggil fungsi untuk menggambar obyek

drawObject();

glEnable(GL\_LIGHTING);

glTranslatef(0.0f, 0.0f, -30.0f);

glRotatef((float)timers.GetTime() / 30, 1.0f, 0.0f, 1.0f);

cubeGrid.DrawSurface(threshold);

glDisable(GL\_LIGHTING);

cubeGrid.DrawSurface(threshold);

glutSwapBuffers();

}

// inisialisasikan variabel, pencahayaan, tekstur,

// pengaturan pandangan kamera dan sebagainya di fungsi init()

void init(void)

{

// inisialisasi warna latar belakang layar

// dalam hal ini warna putih warna putih (1.0, 1.0, 1.0, 0.0)

glClearColor(1.0, 1.0, 1.0, 0.0);

// mengaktifkan depth buffer

glEnable(GL\_DEPTH\_TEST);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

// set proyeksi ke proyeksi perspektif

gluPerspective(45.0, 1.0, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// inisialisasi kamera pandang

// kamera berada di posisi (0.0f, 0.0f, 0.0f)

gluLookAt(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

//set up grid

if (!cubeGrid.CreateMemory())

return;

if (!cubeGrid.Init(gridSize))

return;

//set up metaballs

for (int i = 0; i<numMetaballs; i++)

metaballs[i].Init(VECTOR3D(0.0f, 0.0f, 0.0f), 5.0f + float(i));

//Set Up Colors

diffuseColors[0].Set(0.129f, 0.126f, 0.142f, 1.0f);

diffuseColors[1].Set(0.047f, 0.839f, 0.271f, 1.0f);

diffuseColors[2].Set(0.976f, 0.213f, 0.847f, 1.0f);

timers.Reset();

}

// fungsi ini digunakan bila layar akan diresize (default)

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45, (GLfloat)w / (GLfloat)h, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

}

// fungsi untuk mengatur masukan dari keyboard

// untuk arah kiri, kanan, atas, bawah, PgUp, dan PgDn

void keyboard(int key, int x, int y)

{

float fraction = 0.1f;

switch (key)

{

// masukkan perintah disini bila tombol kiri ditekan

case GLUT\_KEY\_LEFT:

// dalam hal ini perintah rotasi obyek ke kiri sebanyak 1 derajat

objectAngleY -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol kanan ditekan

case GLUT\_KEY\_RIGHT:

// dalam hal ini perintah rotasi obyek ke kanan sebanyak 1 derajat

objectAngleY += 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol atas ditekan

case GLUT\_KEY\_UP:

// dalam hal ini perintah rotasi obyek ke atas sebanyak 1 derajat

objectAngleX -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol bawah ditekan

case GLUT\_KEY\_DOWN:

// dalam hal ini perintah rotasi obyek ke bawah sebanyak 1 derajat

objectAngleX += 1.0f;

glutPostRedisplay(); // update obyek

break;

// zoom in

case GLUT\_KEY\_PAGE\_UP:

// masukkan perintah disini bila tombol PgUp ditekan

posX += rotX \* fraction;

posZ += rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

// zoom out

case GLUT\_KEY\_PAGE\_DOWN:

// masukkan perintah disini bila tombol PgDn ditekan

posX -= rotX \* fraction;

posZ -= rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

}

}

// timer untuk animasi (gunakan bila perlu)

void timer(int value)

{

glutPostRedisplay();

glutTimerFunc(55, timer, 0);

}

// program utama

int main(int argc, char\*\* argv)

{

// inisialisasi jendela OpenGL

// GLUT\_SINGLE berarti memakai single buffer

// GLUT\_DOUBLE berarti memakai double buffer

// GLUT\_RGB berarti mode tampilan yang dipakai RGB

// GLUT\_RGBA berarti mode tampilan yang dipakai RGBA

// GLUT\_DEPTH berarti memakai depth buffer

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA | GLUT\_DEPTH);

// set ukuran jendela tampilan

glutInitWindowSize(480, 480); // besarnya jendela dalam piksel dalam hal ini 300x300

glutInitWindowPosition(100, 100); // posisi jendela dilayar komputer dalam piksel

// judul jendela (wajib diubah dengan informasi NAMA / NIM - JUDUL PRAKTIKUM masing-masing)

glutCreateWindow("SIRLY ZIADATUL MUSTAFIDAH / 2200018345 - TEKNIK PEMODELAN OBYEK 3D");

// panggil fungsi init untuk inisialisasi awal

init();

// event handler untuk display, reshape dan keyboard

glutDisplayFunc(display); // display

glutReshapeFunc(reshape); // reshape

glutSpecialFunc(keyboard); // keyboard

//glutTimerFunc(0, timer, 0); // aktifkan timer bila perlu

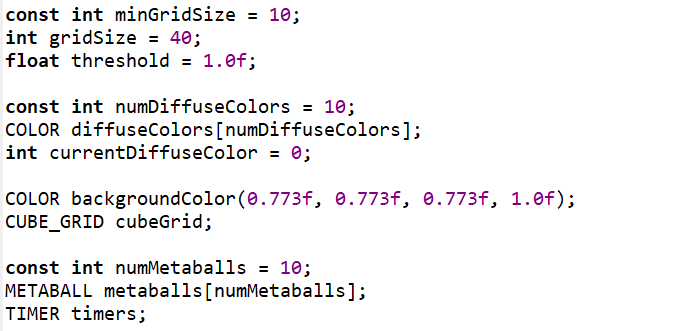
// looping

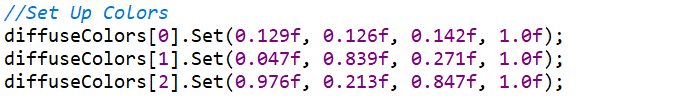
glutMainLoop();

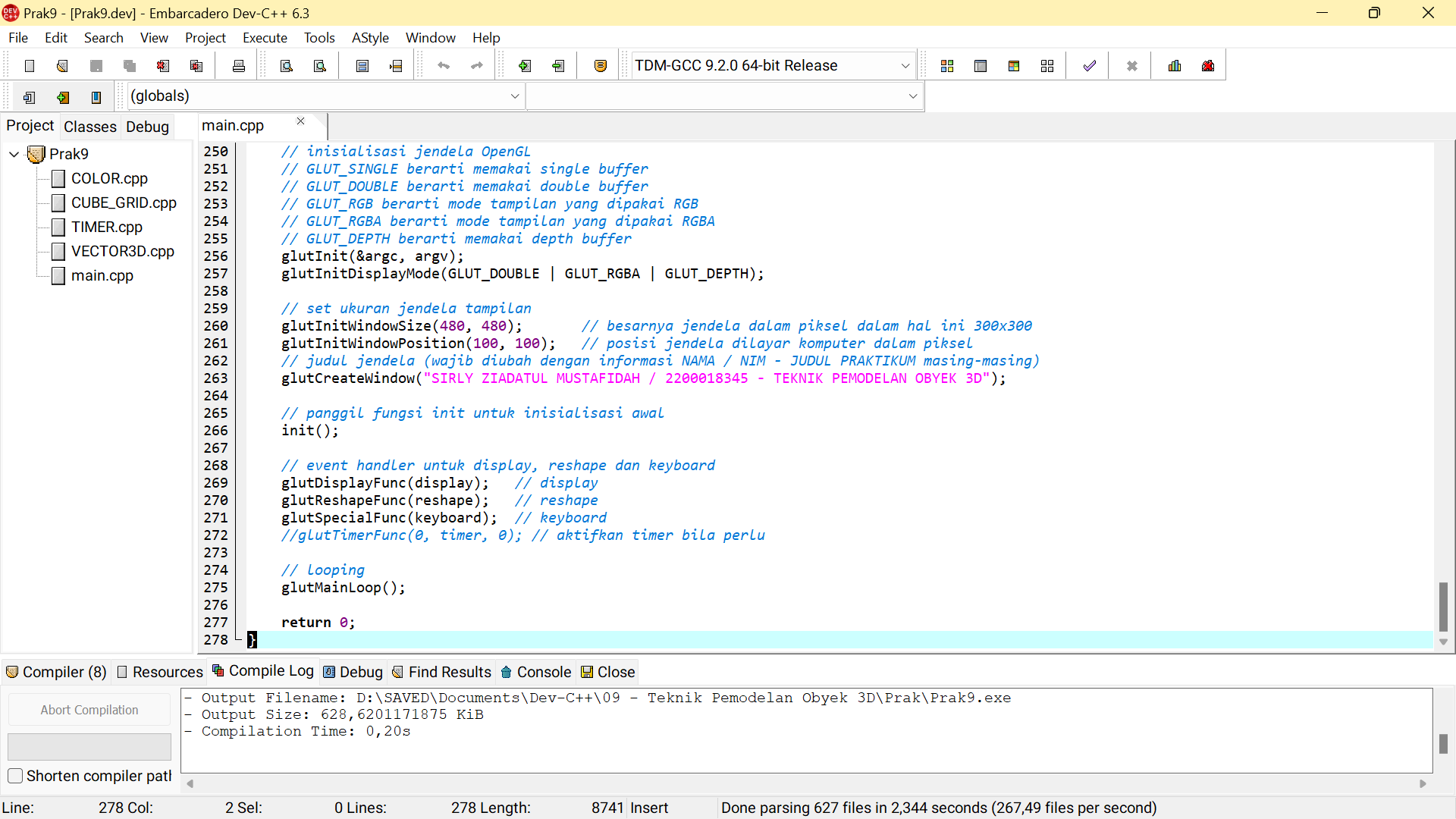
return 0;

}

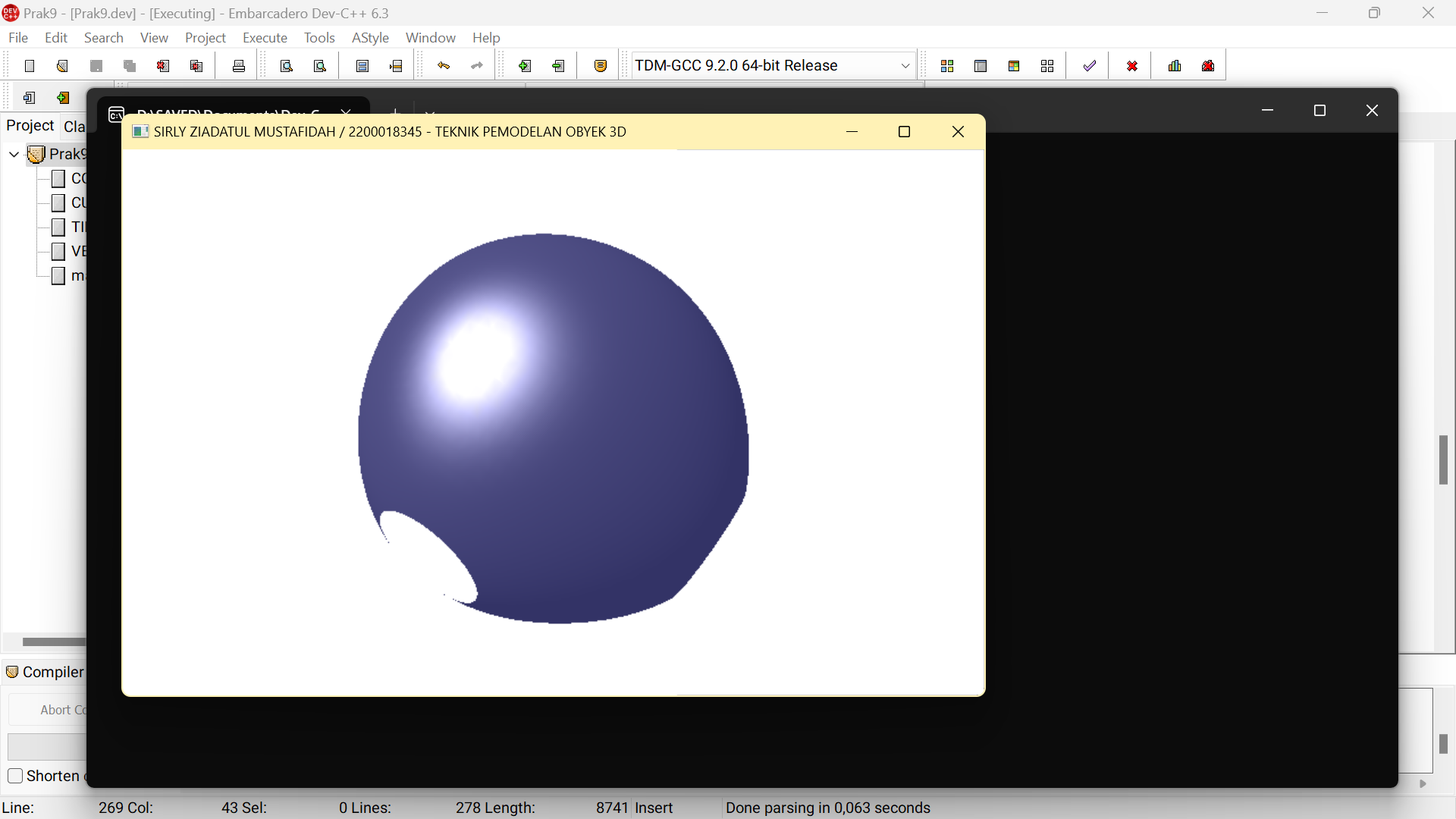
Penjelasan







Output



Gambar 9. 1

1. Posttest 9

Source Code

// deklarasikan semua header disini

#include <windows.h>

#include <GL/glut.h>

#include "COLOR.h"

#include "CUBE\_GRID.h"

#include "METABALL.h"

#include "VECTOR3D.h"

#include "TIMER.h"

// inisialisasi variabel untuk transformasi seperti translasi, rotasi atau scaling

float angle = 0.0f; // sudut transformasi kamera

float posX = 0.0f, rotX = 0.0f; // posisi kamera di sumbu X

float posY = 0.0f, rotY = 0.0f; // posisi kamera di sumbu Y

float posZ = 5.0f, rotZ = -1.0f; // posisi kamera di sumbu Z

float objectAngleX = 0.0f; // sudut tranformasi obyek di sumbu X

float objectAngleY = 0.0f; // sudut tranformasi obyek di sumbu Y

float objectAngleZ = 0.0f; // sudut tranformasi obyek di sumbu Z

const int minGridSize = 10;

int gridSize = 40;

float threshold = 6.0f;

const int numDiffuseColors = 5;

COLOR diffuseColors[numDiffuseColors];

int currentDiffuseColor = 0;

COLOR backgroundColor(0.773f, 0.773f, 0.773f, 1.0f);

CUBE\_GRID cubeGrid;

const int numMetaballs = 5;

METABALL metaballs[numMetaballs];

TIMER timers;

//set up lighting

float shininess = 32.0f;

float ambient[] = { 0.0f, 0.0f, 0.2f, 1.0f };

float position[] = { -1.0f, 1.0f, 1.0f, 0.0f };

float specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };

// fungsi ini digunakan untuk menggambar obyek

void drawObject()

{

glPushMatrix();

// glRotatef(objectAngle, objectRotation.X, objectRotation.Y,

// objectRotation.Z);

glRotatef(objectAngleY, 0.0, 1.0f, 0.0f);

// dinormalisasi dulu

glEnable(GL\_NORMALIZE);

glEnable(GL\_CULL\_FACE);

glLightfv(GL\_LIGHT1, GL\_AMBIENT, ambient);

glLightfv(GL\_LIGHT1, GL\_DIFFUSE, diffuseColors[currentDiffuseColor]);

glLightfv(GL\_LIGHT1, GL\_POSITION, position);

glLightfv(GL\_LIGHT1, GL\_SPECULAR, specular);

glEnable(GL\_LIGHT1);

// set pencahayaan

glMaterialfv(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE, white);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, white);

glMaterialfv(GL\_FRONT, GL\_SHININESS, &shininess);

// update posisi metaball

float c = 2.0f\*(float)cos(timers.GetTime() / 600);

metaballs[0].position.x = -4.0f\*(float)cos(timers.GetTime()/700) - c;

metaballs[0].position.y = 4.0f\*(float)sin(timers.GetTime()/600) - c;

metaballs[1].position.x = 5.0f\*(float)sin(timers.GetTime()/400) + c;

metaballs[1].position.y = 5.0f\*(float)cos(timers.GetTime()/400) - c;

metaballs[2].position.x = -5.0f\*(float)cos(timers.GetTime()/400) -

0.2f\*(float)sin(timers.GetTime() / 600);

metaballs[2].position.y = 5.0f\*(float)sin(timers.GetTime()/500) -

0.2f\*(float)sin(timers.GetTime() / 400);

// bersihkan layar

for (int i = 0; i<cubeGrid.numVertices; i++)

{

cubeGrid.vertices[i].value = 0.0f;

cubeGrid.vertices[i].normal.LoadZero();

}

// hitung bidang skalar disetiap titik

VECTOR3D ballToPoint;

float squaredRadius;

VECTOR3D ballPosition;

float normalScale;

for (int i = 0; i<numMetaballs; i++)

{

squaredRadius = metaballs[i].squaredRadius;

ballPosition = metaballs[i].position;

for (int j = 0; j<cubeGrid.numVertices; j++)

{

ballToPoint.x = cubeGrid.vertices[j].position.x -

ballPosition.x;

ballToPoint.y = cubeGrid.vertices[j].position.y -

ballPosition.y;

ballToPoint.z = cubeGrid.vertices[j].position.z -

ballPosition.z;

// hitung jarak bola ke titik

float squaredDistance = ballToPoint.x\*ballToPoint.x +

ballToPoint.y\*ballToPoint.y + ballToPoint.z\*ballToPoint.z;

if (squaredDistance == 0.0f)

squaredDistance = 0.0001f;

// value = r^2/d^2

cubeGrid.vertices[j].value += squaredRadius / squaredDistance;

// normal = (r^2 \* v)/d^4

normalScale = squaredRadius /(squaredDistance\*squaredDistance);

cubeGrid.vertices[j].normal.x += ballToPoint.x\*normalScale;

cubeGrid.vertices[j].normal.y += ballToPoint.y\*normalScale;

cubeGrid.vertices[j].normal.z += ballToPoint.z\*normalScale;

}

}

glPopMatrix();

}

/// taruh semua fungsi obyek yang akan digambar di fungsi display()

void display()

{

// bersihkan dan reset layar dan buffer

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

gluLookAt(posX, posY, posZ, posX + rotX, posY + rotY, posZ + rotZ, 0.0f, 1.0f, 0.0f);

glShadeModel(GL\_SMOOTH);

// panggil fungsi untuk menggambar obyek

drawObject();

glEnable(GL\_LIGHTING);

glTranslatef(0.0f, 0.0f, -30.0f);

glRotatef((float)timers.GetTime() / 30, 1.0f, 0.0f, 1.0f);

cubeGrid.DrawSurface(threshold);

glDisable(GL\_LIGHTING);

cubeGrid.DrawSurface(threshold);

glutSwapBuffers();

}

// inisialisasikan variabel, pencahayaan, tekstur,

// pengaturan pandangan kamera dan sebagainya di fungsi init()

void init(void)

{

// inisialisasi warna latar belakang layar

// dalam hal ini warna putih warna putih (1.0, 1.0, 1.0, 0.0)

glClearColor(1.0, 1.0, 1.0, 0.0);

// mengaktifkan depth buffer

glEnable(GL\_DEPTH\_TEST);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

// set proyeksi ke proyeksi perspektif

gluPerspective(45.0, 1.0, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// inisialisasi kamera pandang

// kamera berada di posisi (0.0f, 0.0f, 0.0f)

gluLookAt(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

//set up grid

if (!cubeGrid.CreateMemory())

return;

if (!cubeGrid.Init(gridSize))

return;

//set up metaballs

for (int i = 0; i<numMetaballs; i++)

metaballs[i].Init(VECTOR3D(0.0f, 0.0f, 0.0f), 5.0f + float(i));

//Set Up Colors

diffuseColors[0].Set(0.129f, 0.126f, 0.142f, 1.0f);

diffuseColors[1].Set(0.047f, 0.839f, 0.271f, 1.0f);

diffuseColors[2].Set(0.976f, 0.213f, 0.847f, 1.0f);

timers.Reset();

}

// fungsi ini digunakan bila layar akan diresize (default)

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45, (GLfloat)w / (GLfloat)h, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

}

// fungsi untuk mengatur masukan dari keyboard

// untuk arah kiri, kanan, atas, bawah, PgUp, dan PgDn

void keyboard(int key, int x, int y)

{

float fraction = 0.1f;

switch (key)

{

// masukkan perintah disini bila tombol kiri ditekan

case GLUT\_KEY\_LEFT:

// dalam hal ini perintah rotasi obyek ke kiri sebanyak 1 derajat

objectAngleY -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol kanan ditekan

case GLUT\_KEY\_RIGHT:

// dalam hal ini perintah rotasi obyek ke kanan sebanyak 1 derajat

objectAngleY += 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol atas ditekan

case GLUT\_KEY\_UP:

// dalam hal ini perintah rotasi obyek ke atas sebanyak 1 derajat

objectAngleX -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol bawah ditekan

case GLUT\_KEY\_DOWN:

// dalam hal ini perintah rotasi obyek ke bawah sebanyak 1 derajat

objectAngleX += 1.0f;

glutPostRedisplay(); // update obyek

break;

// zoom in

case GLUT\_KEY\_PAGE\_UP:

// masukkan perintah disini bila tombol PgUp ditekan

posX += rotX \* fraction;

posZ += rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

// zoom out

case GLUT\_KEY\_PAGE\_DOWN:

// masukkan perintah disini bila tombol PgDn ditekan

posX -= rotX \* fraction;

posZ -= rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

}

}

// timer untuk animasi (gunakan bila perlu)

void timer(int value)

{

glutPostRedisplay();

glutTimerFunc(55, timer, 0);

}

// program utama

int main(int argc, char\*\* argv)

{

// inisialisasi jendela OpenGL

// GLUT\_SINGLE berarti memakai single buffer

// GLUT\_DOUBLE berarti memakai double buffer

// GLUT\_RGB berarti mode tampilan yang dipakai RGB

// GLUT\_RGBA berarti mode tampilan yang dipakai RGBA

// GLUT\_DEPTH berarti memakai depth buffer

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA | GLUT\_DEPTH);

// set ukuran jendela tampilan

glutInitWindowSize(480, 480); // besarnya jendela dalam piksel dalam hal ini 300x300

glutInitWindowPosition(100, 100); // posisi jendela dilayar komputer dalam piksel

// judul jendela (wajib diubah dengan informasi NAMA / NIM - JUDUL PRAKTIKUM masing-masing)

glutCreateWindow("SIRLY ZIADATUL MUSTAFIDAH / 2200018345 - TEKNIK PEMODELAN OBYEK 3D");

// panggil fungsi init untuk inisialisasi awal

init();

// event handler untuk display, reshape dan keyboard

glutDisplayFunc(display); // display

glutReshapeFunc(reshape); // reshape

glutSpecialFunc(keyboard); // keyboard

//glutTimerFunc(0, timer, 0); // aktifkan timer bila perlu

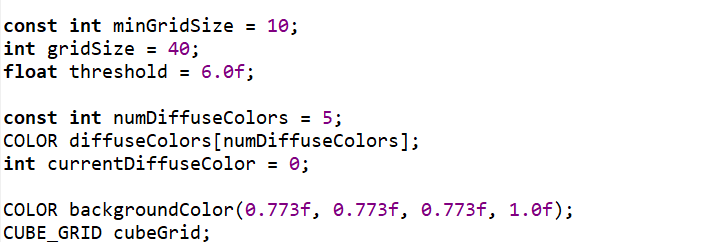
// looping

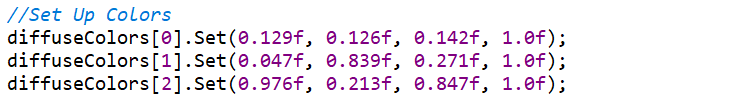
glutMainLoop();

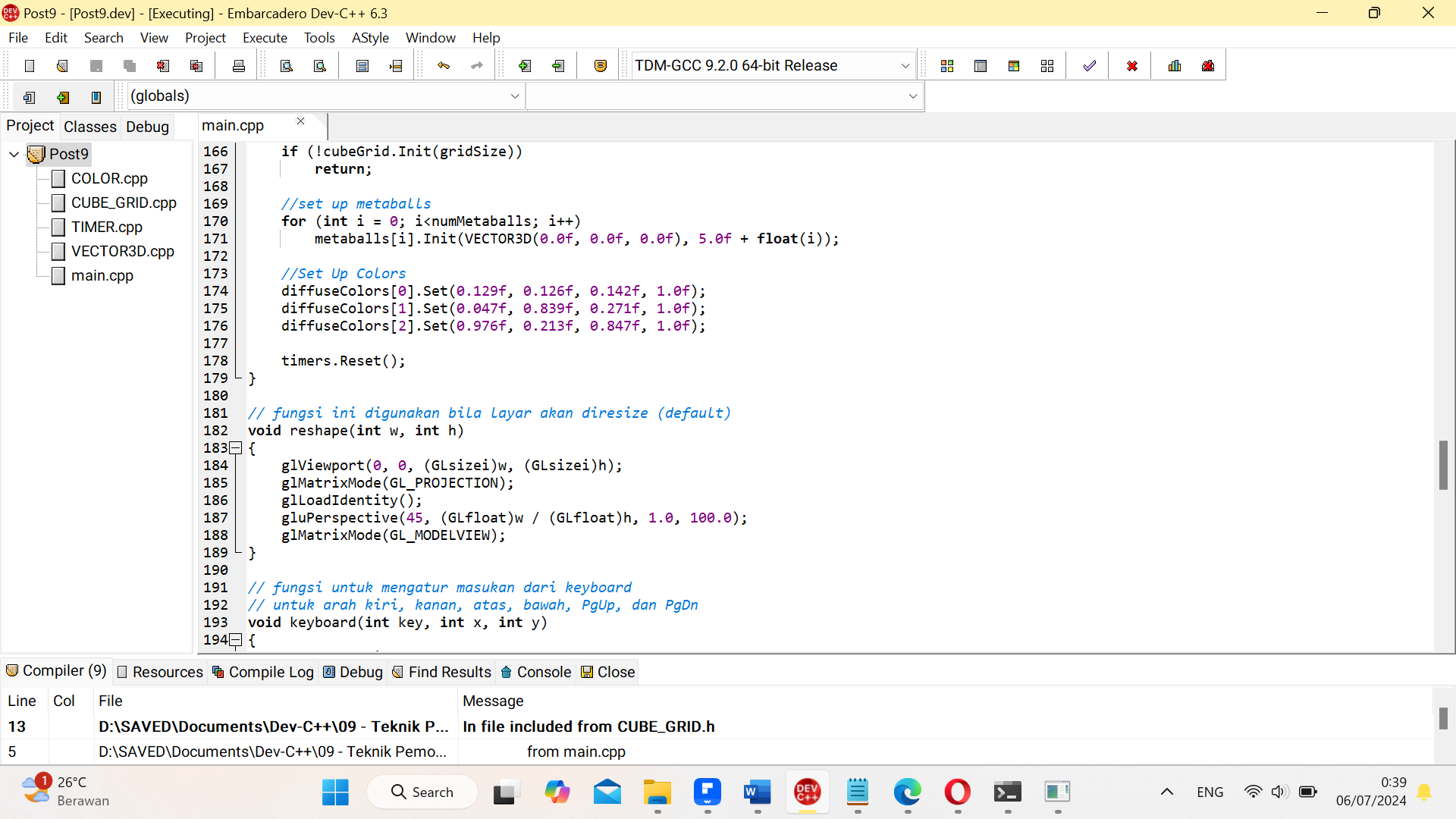
return 0;

}

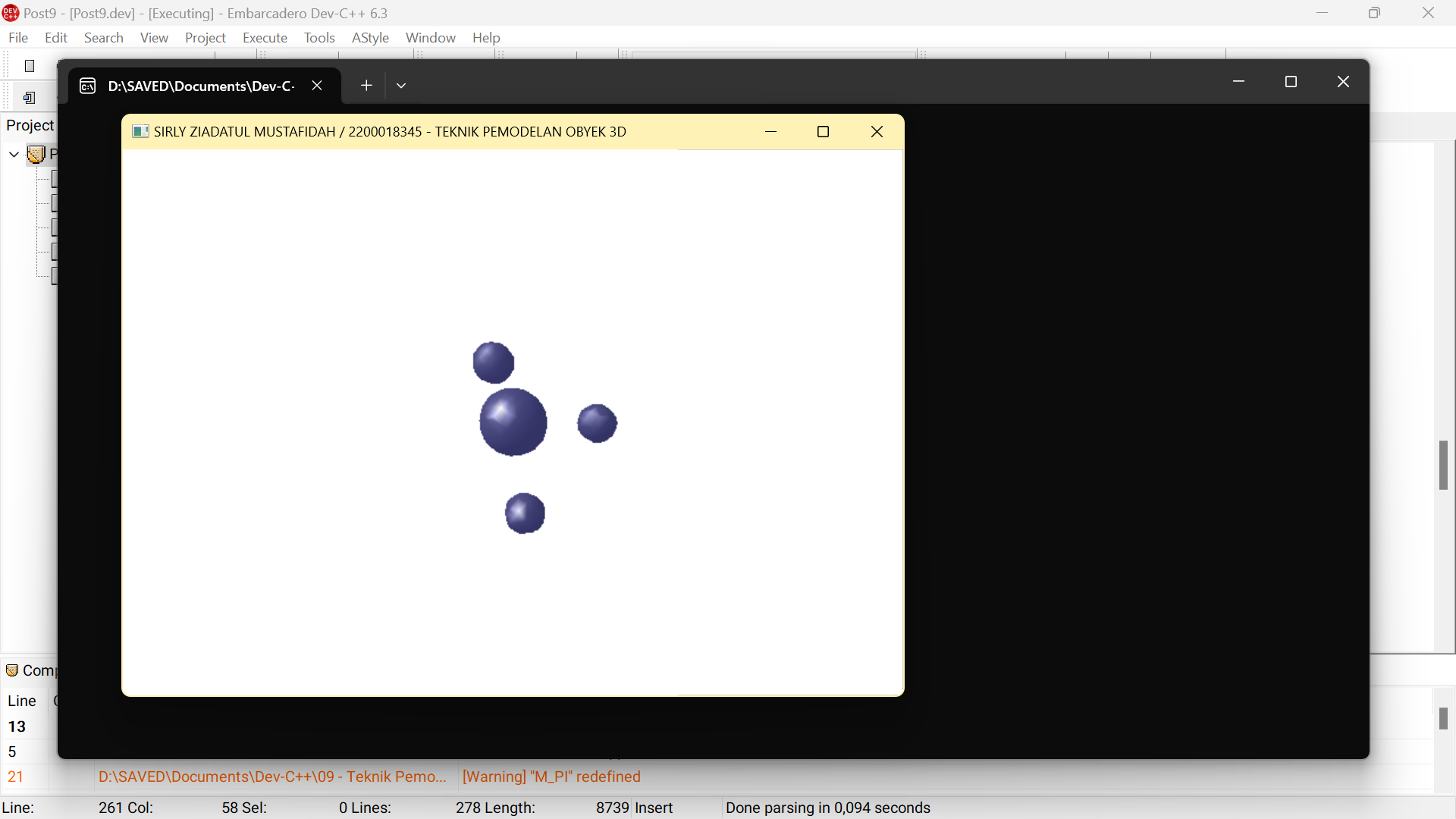
Penjelasan







Output



Gambar 9. 2

# BAB X. TEKNIK SUBDIVISI

1. Pretest 10
2. Bagaimana cara menerapkan subdivisi 2x pada permukaan segitiga?

* Bagi setiap sisi segitiga menjadi dua bagian yang sama panjang dengan membuat titik tengah di setiap sisi
* Buat titik baru setiap sudut segititga asli dengan menghubungkan titik-titik tengah pada sisi-sisinya
* Buat 4 segitiga baru dengan menghubungkan titik-titk baru yang telah dibuat
* Ulangi langkah tersebut

1. Langkah Praktikum 10

Source Code

// deklarasikan semua header disini

#include <windows.h>

#include <GL/glut.h>

//langkah 1

#include <math.h>

#define vX 0.525731112119133696

#define vZ 0.850650808352039932

// inisialisasi variabel untuk transformasi seperti translasi, rotasi atau scaling

float angle = 0.0f; // sudut transformasi kamera

float posX = 0.0f, rotX = 0.0f; // posisi kamera di sumbu X

float posY = 0.0f, rotY = 0.0f; // posisi kamera di sumbu Y

float posZ = 5.0f, rotZ = -1.0f; // posisi kamera di sumbu Z

float objectAngleX = 0.0f; // sudut tranformasi obyek di sumbu X

float objectAngleY = 0.0f; // sudut tranformasi obyek di sumbu Y

float objectAngleZ = 0.0f; // sudut tranformasi obyek di sumbu Z

// container untuk membuat tipe data 3D (X, Y, Z)

struct Vec3 {

float X; float Y; float Z;

Vec3(float x, float y, float z) { X = x; Y = y; Z = z; }

Vec3() { }

~Vec3() { }

};

//langkah 1

// vertex data array

static GLfloat vdata[12][3] =

{

{ -vX, 0.0, vZ },{ vX, 0.0, vZ },{ -vX, 0.0, -vZ },{ vX, 0.0, -vZ },

{ 0.0, vZ, vX },{ 0.0, vZ, -vX },{ 0.0, -vZ, vX },{ 0.0, -vZ, -vX },

{ vZ, vX, 0.0 },{ -vZ, vX, 0.0 },{ vZ, -vX, 0.0 },{ -vZ, -vX, 0.0 }

};

// titik-titik segitiga

static int tindices[20][3] = {

{ 1,4,0 },{ 4,9,0 },{ 4,5,9 },{ 8,5,4 },{ 1,8,4 },

{ 1,10,8 },{ 10,3,8 },{ 8,3,5 },{ 3,2,5 },{ 3,7,2 },

{ 3,10,7 },{ 10,6,7 },{ 6,11,7 },{ 6,0,11 },{ 6,1,0 },

{ 10,1,6 },{ 11,0,9 },{ 2,11,9 },{ 5,2,9 },{ 11,2,7 }

};

GLfloat mat\_specular[] = { 0.0, 0.0, 0.0, 1.0 };

GLfloat mat\_diffuse[] = { 0.8, 0.6, 0.4, 1.0 };

GLfloat mat\_ambient[] = { 0.8, 0.6, 0.4, 1.0 };

GLfloat mat\_shininess = 100.0;

GLfloat light\_ambient[] = { 0.2, 0.2, 0.2, 1.0 };

GLfloat light\_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };

GLfloat light\_specular[] = { 0.0, 0.0, 0.0, 1.0 };

GLfloat light\_position1[] = { 1.5, 1.0, -2.0, 0.0 };

GLfloat light\_position2[] = { 1.5, 1.0, 2.0, 0.0 };

int flat = 1; // 0 = smooth shading, 1 = flat shading

int subdiv = 4;

//langkah 2

// fungsi untuk melakukan normalisasi koordinat posisi

Vec3 normalize(Vec3 value)

{

Vec3 result;

float lengths = sqrt((value.X \* value.X) + (value.Y \* value.Y) + (value.Z \* value.Z));

result.X = value.X / lengths;

result.Y = value.Y / lengths;

result.Z = value.Z / lengths;

return result;

}

// fungsi untuk melakukan operasi perkalian cross

Vec3 cross(Vec3 value1, Vec3 value2)

{

Vec3 result;

result.X = value1.Y \* value2.Z - value2.Y \* value1.Z;

result.Y = value1.Z \* value2.X - value2.Z \* value1.X;

result.Z = value1.X \* value2.Y - value2.X \* value1.Y;

return result;

}

// fungsi untuk menghitung normal

void normface(Vec3 v1, Vec3 v2, Vec3 v3)

{

Vec3 d1, d2;

d1.X = v1.X - v2.X; d1.Y = v1.Y - v2.Y; d1.Z = v1.Z - v2.Z;

d2.X = v2.X - v3.X; d2.Y = v2.Y - v3.Y; d2.Z = v2.Z - v3.Z;

Vec3 tn = cross(d1, d2);

tn = normalize(tn);

glNormal3f(tn.X, tn.Y, tn.Z);

}

// menggambar polygon segitiga dengan sisi normal

void drawTriangleFlat(Vec3 v1, Vec3 v2, Vec3 v3)

{

glBegin(GL\_TRIANGLES);

normface(v1, v2, v3);

glVertex3f(v1.X, v1.Y, v1.Z);

glVertex3f(v2.X, v2.Y, v2.Z);

glVertex3f(v3.X, v3.Y, v3.Z);

glEnd();

}

// menggambar polygon segitiga smooth dengan normal

void drawTriangleSmooth(Vec3 v1, Vec3 v2, Vec3 v3)

{

glBegin(GL\_TRIANGLES);

glNormal3f(v1.X, v1.Y, v1.Z);

glVertex3f(v1.X, v1.Y, v1.Z);

glNormal3f(v2.X, v2.Y, v2.Z);

glVertex3f(v2.X, v2.Y, v2.Z);

glNormal3f(v3.X, v3.Y, v3.Z);

glVertex3f(v3.X, v3.Y, v3.Z);

glEnd();

}

// subdivisi permukaan secara rekursif

// gambar hasil subdivisi segitiganya

void subdivide(Vec3 &v1, Vec3 &v2, Vec3 &v3, int depth)

{

Vec3 v12, v23, v31;

if (depth == 0)

{

if (flat == 1)

drawTriangleFlat(v1, v2, v3);

else

drawTriangleSmooth(v1, v2, v3);

return;

}

// hitung titik tengah polygon segitiga

v12.X = (v1.X + v2.X) / 2.0;

v12.Y = (v1.Y + v2.Y) / 2.0;

v12.Z = (v1.Z + v2.Z) / 2.0;

v23.X = (v2.X + v3.X) / 2.0;

v23.Y = (v2.Y + v3.Y) / 2.0;

v23.Z = (v2.Z + v3.Z) / 2.0;

v31.X = (v3.X + v1.X) / 2.0;

v31.Y = (v3.Y + v1.Y) / 2.0;

v31.Z = (v3.Z + v1.Z) / 2.0;

// extrude titik tengahnya

v12 = normalize(v12);

v23 = normalize(v23);

v31 = normalize(v31);

// subdivisi polygon segitiga secara rekursif

subdivide(v1, v12, v31, depth - 1);

subdivide(v2, v23, v12, depth - 1);

subdivide(v3, v31, v23, depth - 1);

subdivide(v12, v23, v31, depth - 1);

}

//langkah 3

// fungsi untuk menggambar obyek

void drawObject()

{

glPushMatrix();

glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position1);

glLightfv(GL\_LIGHT1, GL\_POSITION, light\_position2);

// operasi transformasi rotasi obyek ke arah kanan-kiri

glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);

glPushMatrix();

// operasi transformasi rotasi obyek ke arah atas-bawah

glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

// buat daftar vertex

for (int i = 0; i < 20; i++)

{

Vec3 vdata1 = Vec3(

vdata[tindices[i][0]][0],

vdata[tindices[i][0]][1],

vdata[tindices[i][0]][2]);

Vec3 vdata2 = Vec3(

vdata[tindices[i][1]][0],

vdata[tindices[i][1]][1],

vdata[tindices[i][1]][2]);

Vec3 vdata3 = Vec3(

vdata[tindices[i][2]][0],

vdata[tindices[i][2]][1],

vdata[tindices[i][2]][2]);

subdivide(vdata1, vdata2, vdata3, subdiv);

vdata[tindices[i][0]][0] = vdata1.X;

vdata[tindices[i][0]][1] = vdata1.Y;

vdata[tindices[i][0]][2] = vdata1.Z;

vdata[tindices[i][1]][0] = vdata2.X;

vdata[tindices[i][1]][1] = vdata2.Y;

vdata[tindices[i][1]][2] = vdata2.Z;

vdata[tindices[i][2]][0] = vdata3.X;

vdata[tindices[i][2]][1] = vdata3.Y;

vdata[tindices[i][2]][2] = vdata3.Z;

}

glPopMatrix();

glPopMatrix();

}

// taruh semua obyek yang akan digambar di fungsi display()

void display()

{

// bersihkan dan reset layar dan buffer

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

// posisikan kamera pandang

// dalam hal ini sumbu Y ada diatas dan posisi kamera pandang di (posX, posY, posZ)

gluLookAt(posX, posY, posZ, posX + rotX, posY + rotY, posZ + rotZ, 0.0f, 1.0f, 0.0f);

// panggil fungsi untuk menggambar obyek

drawObject();

// tampilkan obyek ke layar

// gunakan glFlush() bila memakai single buffer

// gunakan glutSwapBuffers() bila memakai double buffer

glutSwapBuffers();

}

//langkah 4

// inisialisasi

void init(void) {

// inisialisasi warna latar belakang

glClearColor(0.1, 0.1, 0.2, 0.0);

glEnable(GL\_DEPTH\_TEST); // mengaktifkan depth buffer

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45.0, 1.0, 1.0, 100.0);// set proyeksi ke perspektif

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// inisialisasi kamera pandang

gluLookAt(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

// inisialisasi pencahayaan

glLightfv(GL\_LIGHT0, GL\_AMBIENT, light\_ambient);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light\_diffuse);

glLightfv(GL\_LIGHT0, GL\_SPECULAR, light\_specular);

glLightfv(GL\_LIGHT1, GL\_AMBIENT, light\_ambient);

glLightfv(GL\_LIGHT1, GL\_DIFFUSE, light\_diffuse);

glLightfv(GL\_LIGHT1, GL\_SPECULAR, light\_specular);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat\_specular);

glMaterialfv(GL\_FRONT, GL\_AMBIENT, mat\_ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, mat\_diffuse);

glMaterialf(GL\_LIGHT0, GL\_SHININESS, mat\_shininess);

glMaterialf(GL\_LIGHT1, GL\_SHININESS, mat\_shininess);

glShadeModel(GL\_SMOOTH); // aktifkan smooth shading

glEnable(GL\_LIGHTING); // aktifkan pencahayaan

glEnable(GL\_LIGHT0); // aktifkan sumber cahaya 0

glEnable(GL\_LIGHT1); // aktifkan sumber cahaya 1

}

// fungsi ini digunakan bila layar akan diresize (default)

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45, (GLfloat)w / (GLfloat)h, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

}

//langkah 5

// fungsi untuk mengatur masukan dari keyboard

// untuk arah kiri, kanan, atas, bawah, PgUp, dan PgDn

void keyboard(int key, int x, int y)

{

float fraction = 0.1f;

switch (key)

{

// masukkan perintah disini bila tombol kiri ditekan

case GLUT\_KEY\_LEFT:

// dalam hal ini perintah rotasi obyek ke kiri sebanyak 1 derajat

objectAngleY -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol kanan ditekan

case GLUT\_KEY\_RIGHT:

// dalam hal ini perintah rotasi obyek ke kanan sebanyak 1 derajat

objectAngleY += 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol atas ditekan

case GLUT\_KEY\_UP:

// dalam hal ini perintah rotasi obyek ke atas sebanyak 1 derajat

objectAngleX -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol bawah ditekan

case GLUT\_KEY\_DOWN:

// dalam hal ini perintah rotasi obyek ke bawah sebanyak 1 derajat

objectAngleX += 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol PgUp ditekan

case GLUT\_KEY\_PAGE\_UP:

posX += rotX \* fraction;

posZ += rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol PgDn ditekan

case GLUT\_KEY\_PAGE\_DOWN:

posX -= rotX \* fraction;

posZ -= rotZ \* fraction;

glutPostRedisplay(); // update obyek

case GLUT\_KEY\_F1:

subdiv++; // lakukan subdivisi

glutPostRedisplay(); // update obyek

break;

case GLUT\_KEY\_F2:

subdiv--; // lakukan subdivisi

glutPostRedisplay(); // update obyek

break;

}

if (subdiv<0) subdiv = 0;

}

// timer untuk animasi (gunakan bila perlu)

void timer(int value)

{

glutPostRedisplay();

glutTimerFunc(55, timer, 0);

}

// program utama

int main(int argc, char\*\* argv)

{

// inisialisasi jendela OpenGL

// GLUT\_SINGLE berarti memakai single buffer

// GLUT\_DOUBLE berarti memakai double buffer

// GLUT\_RGB berarti mode tampilan yang dipakai RGB

// GLUT\_RGBA berarti mode tampilan yang dipakai RGBA

// GLUT\_DEPTH berarti memakai depth buffer

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA | GLUT\_DEPTH);

// set ukuran jendela tampilan

glutInitWindowSize(480, 480); // besarnya jendela dalam piksel dalam hal ini 300x300

glutInitWindowPosition(100, 100); // posisi jendela dilayar komputer dalam piksel

// judul jendela (wajib diubah dengan informasi NAMA / NIM - JUDUL PRAKTIKUM masing-masing)

glutCreateWindow("SIRLY ZIADATUL MUSTAFIDAH / 2200018345 - TEKNIK SUBDIVISI");

// panggil fungsi init untuk inisialisasi awal

init();

// event handler untuk display, reshape dan keyboard

glutDisplayFunc(display); // display

glutReshapeFunc(reshape); // reshape

glutSpecialFunc(keyboard); // keyboard

//glutTimerFunc(0, timer, 0); // aktifkan timer bila perlu

// looping

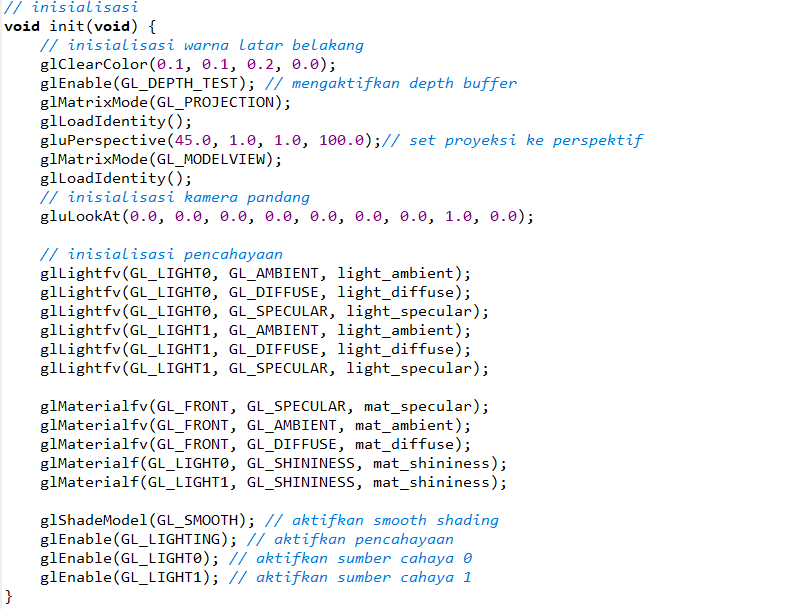
glutMainLoop();

return 0;

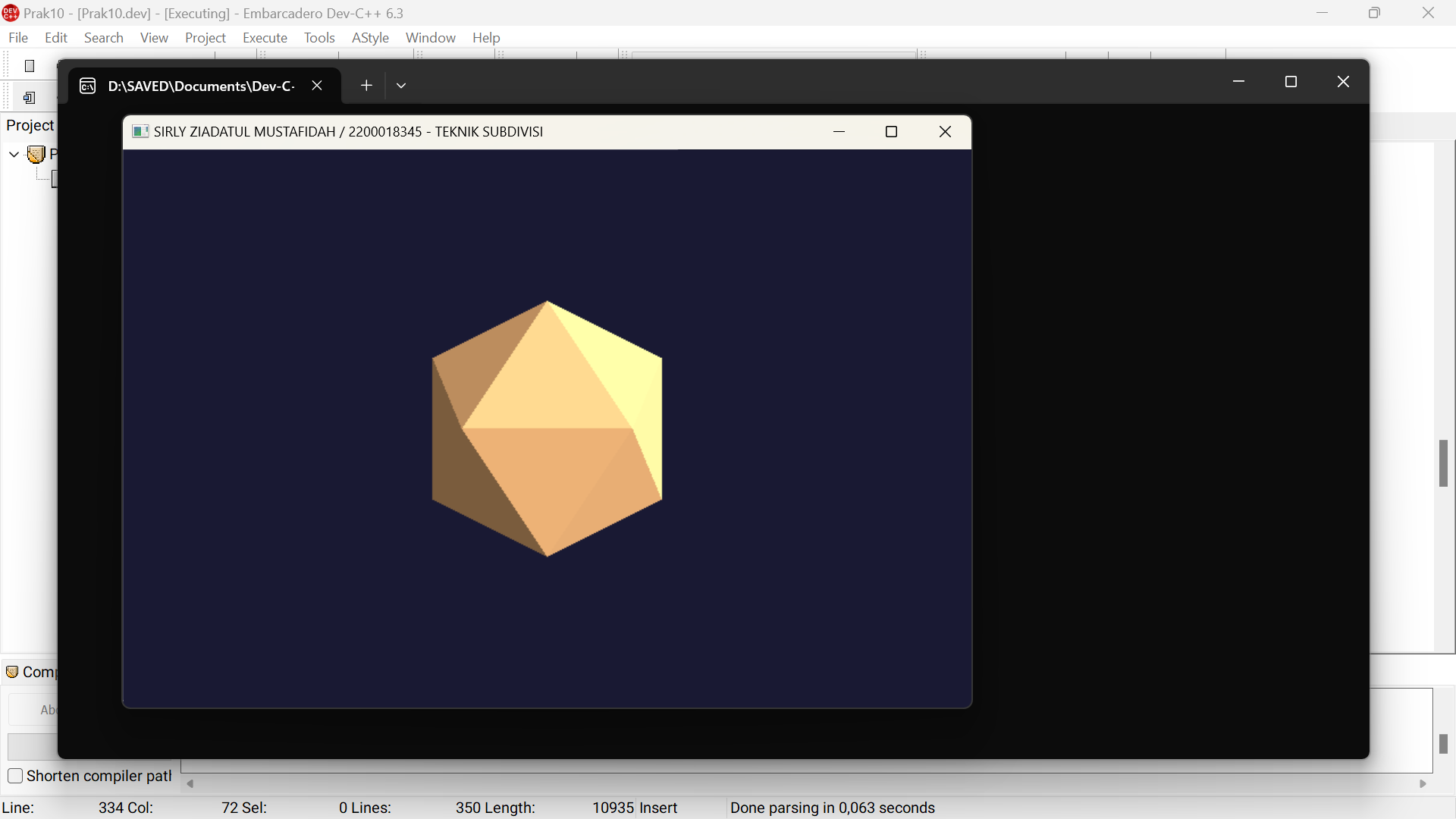
}

Penjelasan

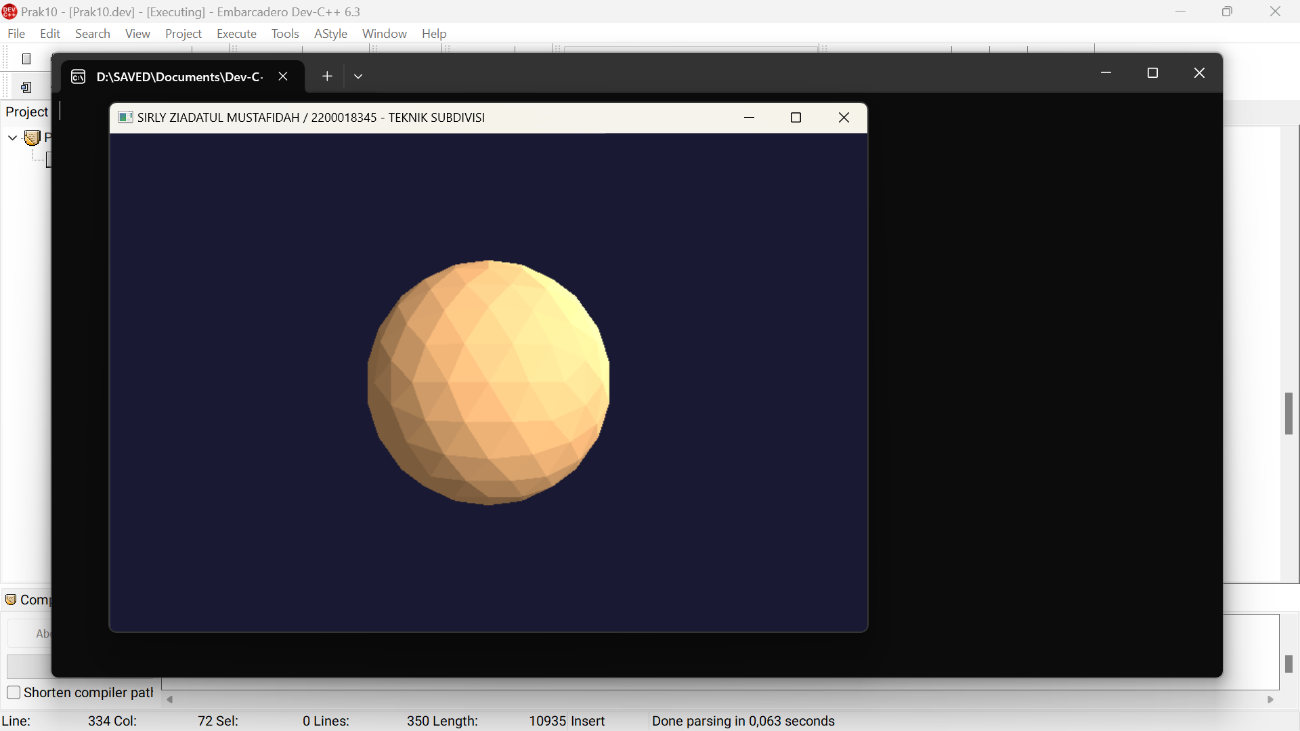




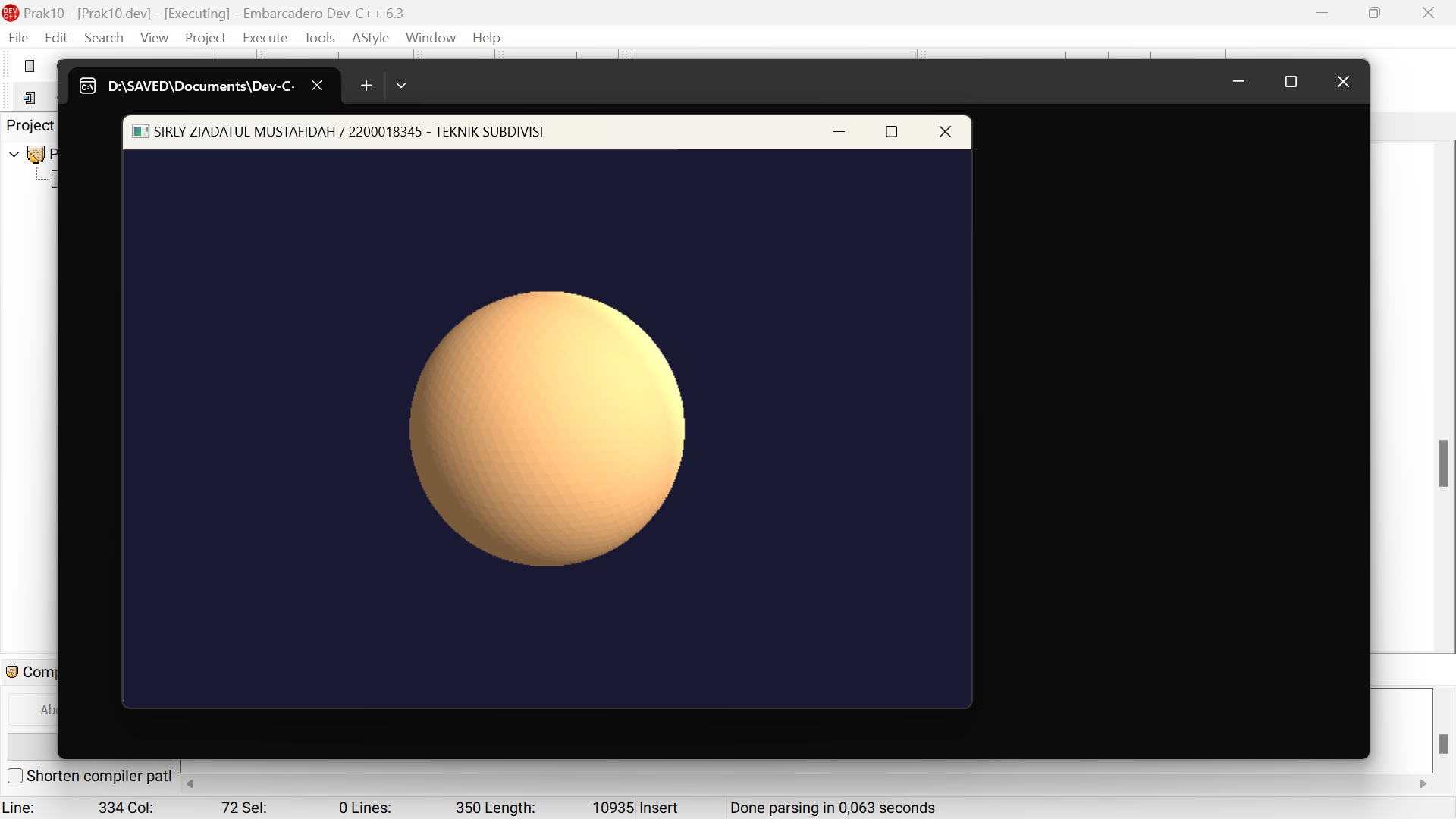
Output



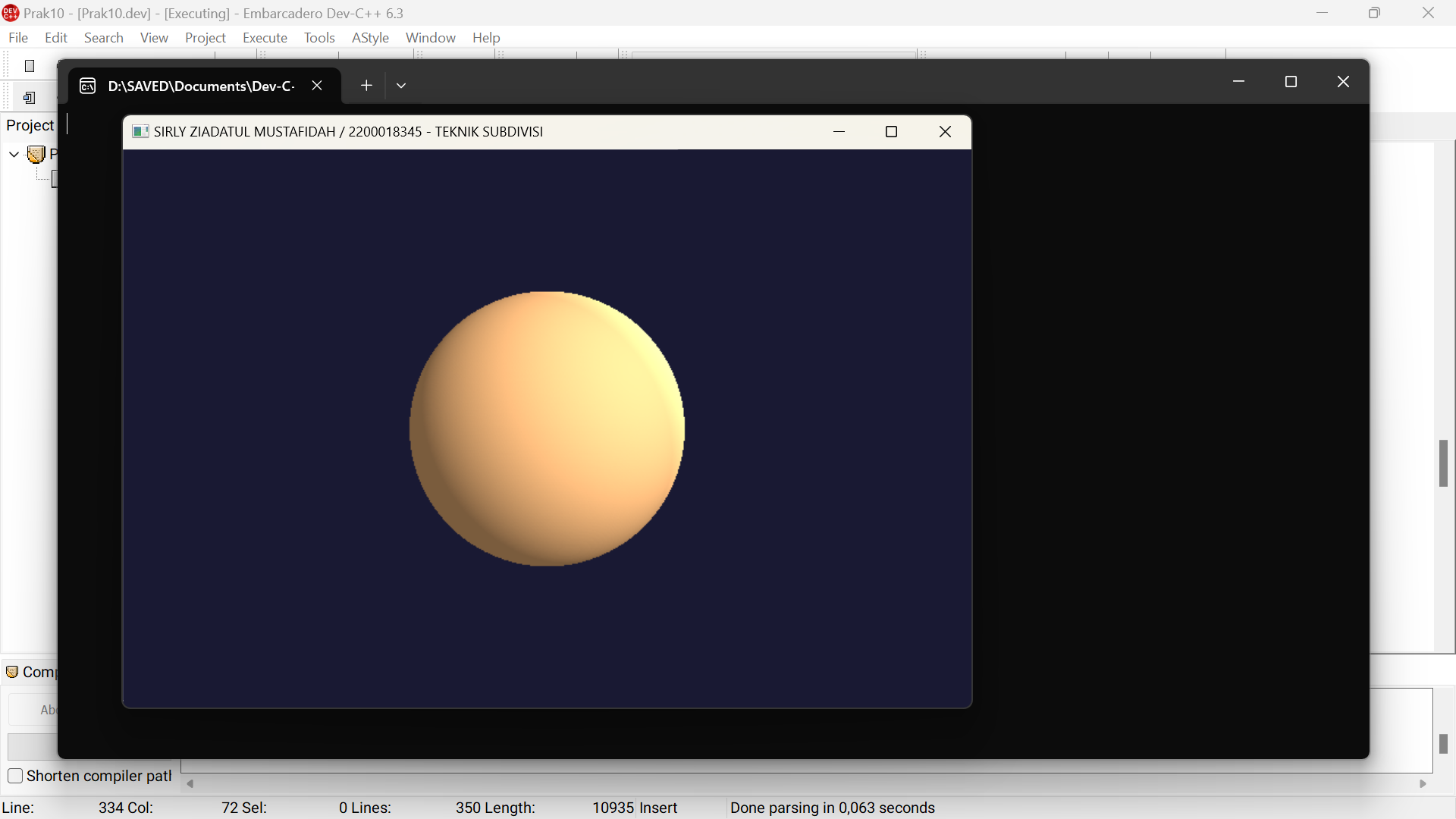
Gambar 10. 1



Gambar 10. 2



Gambar 10. 3



Gambar 10. 4

1. Posttest 10

Source Code

// deklarasikan semua header disini

#include <windows.h>

#include <GL/glut.h>

//langkah 1

#include <math.h>

#define vX 0.525731112119133696

#define vZ 0.850650808352039932

// inisialisasi variabel untuk transformasi seperti translasi, rotasi atau scaling

float angle = 0.0f; // sudut transformasi kamera

float posX = 0.0f, rotX = 0.0f; // posisi kamera di sumbu X

float posY = 0.0f, rotY = 0.0f; // posisi kamera di sumbu Y

float posZ = 5.0f, rotZ = -1.0f; // posisi kamera di sumbu Z

float objectAngleX = 0.0f; // sudut tranformasi obyek di sumbu X

float objectAngleY = 0.0f; // sudut tranformasi obyek di sumbu Y

float objectAngleZ = 0.0f; // sudut tranformasi obyek di sumbu Z

// container untuk membuat tipe data 3D (X, Y, Z)

struct Vec3 {

float X; float Y; float Z;

Vec3(float x, float y, float z) { X = x; Y = y; Z = z; } //

Vec3() { }

~Vec3() { }

};

//langkah 1

// vertex data array

static GLfloat vdata[12][3] =

{

{ -vX, 0.0, vZ },{ vX, 0.0, vZ },{ -vX, 0.0, -vZ },{ vX, 0.0, -vZ },

{ 0.0, vZ, vX },{ 0.0, vZ, -vX },{ 0.0, -vZ, vX },{ 0.0, -vZ, -vX },

{ vZ, vX, 0.0 },{ -vZ, vX, 0.0 },{ vZ, -vX, 0.0 },{ -vZ, -vX, 0.0 }

};

// titik-titik segitiga

static int tindices[20][3] = {

{ 1,4,0 },{ 4,9,0 },{ 4,5,9 },{ 8,5,4 },{ 1,8,4 },

{ 1,10,8 },{ 10,3,8 },{ 8,3,5 },{ 3,2,5 },{ 3,7,2 },

{ 3,10,7 },{ 10,6,7 },{ 6,11,7 },{ 6,0,11 },{ 6,1,0 },

{ 10,1,6 },{ 11,0,9 },{ 2,11,9 },{ 5,2,9 },{ 11,2,7 }

};

GLfloat mat\_specular[] = { 0.0, 0.0, 0.0, 1.0 };

GLfloat mat\_diffuse[] = { 0.8, 0.6, 0.4, 1.0 };

GLfloat mat\_ambient[] = { 0.8, 0.6, 0.4, 1.0 };

GLfloat mat\_shininess = 100.0;

GLfloat light\_ambient[] = { 0.2, 0.2, 0.2, 1.0 };

GLfloat light\_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };

GLfloat light\_specular[] = { 0.0, 0.0, 0.0, 1.0 };

GLfloat light\_position1[] = { 1.5, 1.0, -2.0, 0.0 };

GLfloat light\_position2[] = { 1.5, 1.0, 2.0, 0.0 };

int flat = 1; // 0 = smooth shading, 1 = flat shading

int subdiv = 4;

//langkah 2

// fungsi untuk melakukan normalisasi koordinat posisi

Vec3 normalize(Vec3 value)

{

Vec3 result;

float lengths = sqrt((value.X \* value.X) + (value.Y \* value.Y) + (value.Z \* value.Z));

result.X = value.X / lengths;

result.Y = value.Y / lengths;

result.Z = value.Z / lengths;

return result;

}

// fungsi untuk melakukan operasi perkalian cross

Vec3 cross(Vec3 value1, Vec3 value2)

{

Vec3 result;

result.X = value1.Y \* value2.Z - value2.Y \* value1.Z;

result.Y = value1.Z \* value2.X - value2.Z \* value1.X;

result.Z = value1.X \* value2.Y - value2.X \* value1.Y;

return result;

}

// fungsi untuk menghitung normal

void normface(Vec3 v1, Vec3 v2, Vec3 v3)

{

Vec3 d1, d2;

d1.X = v1.X - v2.X; d1.Y = v1.Y - v2.Y; d1.Z = v1.Z - v2.Z;

d2.X = v2.X - v3.X; d2.Y = v2.Y - v3.Y; d2.Z = v2.Z - v3.Z;

Vec3 tn = cross(d1, d2);

tn = normalize(tn);

glNormal3f(tn.X, tn.Y, tn.Z);

}

// menggambar polygon segitiga dengan sisi normal

void drawTriangleFlat(Vec3 v1, Vec3 v2, Vec3 v3)

{

glBegin(GL\_TRIANGLES);

normface(v1, v2, v3);

glVertex3f(v1.X, v1.Y, v1.Z);

glVertex3f(v2.X, v2.Y, v2.Z);

glVertex3f(v3.X, v3.Y, v3.Z);

glEnd();

}

// menggambar polygon segitiga smooth dengan normal

void drawTriangleSmooth(Vec3 v1, Vec3 v2, Vec3 v3)

{

glBegin(GL\_TRIANGLES);

glNormal3f(v1.X, v1.Y, v1.Z);

glVertex3f(v1.X, v1.Y, v1.Z);

glNormal3f(v2.X, v2.Y, v2.Z);

glVertex3f(v2.X, v2.Y, v2.Z);

glNormal3f(v3.X, v3.Y, v3.Z);

glVertex3f(v3.X, v3.Y, v3.Z);

glEnd();

}

// subdivisi permukaan secara rekursif

// gambar hasil subdivisi segitiganya

void subdivide(Vec3 &v1, Vec3 &v2, Vec3 &v3, int depth)

{

Vec3 v12, v23, v31;

if (depth == 0)

{

if (flat == 1)

drawTriangleFlat(v1, v2, v3);

else

drawTriangleSmooth(v1, v2, v3);

return;

}

// hitung titik tengah polygon segitiga

v12.X = (v1.X + v2.X) / 2.0;

v12.Y = (v1.Y + v2.Y) / 2.0;

v12.Z = (v1.Z + v2.Z) / 2.0;

v23.X = (v2.X + v3.X) / 2.0;

v23.Y = (v2.Y + v3.Y) / 2.0;

v23.Z = (v2.Z + v3.Z) / 2.0;

v31.X = (v3.X + v1.X) / 2.0;

v31.Y = (v3.Y + v1.Y) / 2.0;

v31.Z = (v3.Z + v1.Z) / 2.0;

// extrude titik tengahnya

v12 = normalize(v12);

v23 = normalize(v23);

v31 = normalize(v31);

// subdivisi polygon segitiga secara rekursif

subdivide(v1, v12, v31, depth - 1);

subdivide(v2, v23, v12, depth - 1);

subdivide(v3, v31, v23, depth - 1);

subdivide(v12, v23, v31, depth - 1);

}

//langkah 3

// fungsi untuk menggambar obyek kubus

void drawObject()

{

glPushMatrix();

glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position1);

glLightfv(GL\_LIGHT1, GL\_POSITION, light\_position2);

// operasi transformasi rotasi obyek ke arah kanan-kiri

glRotatef(objectAngleY, 0.0f, 1.0f, 0.0f);

glPushMatrix();

// operasi transformasi rotasi obyek ke arah atas-bawah

glRotatef(objectAngleX, 1.0f, 0.0f, 0.0f);

glutSolidTorus(0.3f, 0.5f, subdiv, subdiv);

glPopMatrix();

glPopMatrix();

}

// taruh semua obyek yang akan digambar di fungsi display()

void display()

{

// bersihkan dan reset layar dan buffer

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

// posisikan kamera pandang

// dalam hal ini sumbu Y ada diatas dan posisi kamera pandang di (posX, posY, posZ)

gluLookAt(posX, posY, posZ, posX + rotX, posY + rotY, posZ + rotZ, 0.0f, 1.0f, 0.0f);

// panggil fungsi untuk menggambar obyek

drawObject();

// tampilkan obyek ke layar

// gunakan glFlush() bila memakai single buffer

// gunakan glutSwapBuffers() bila memakai double buffer

glutSwapBuffers();

}

//langkah 4

// inisialisasi

void init(void) {

// inisialisasi warna latar belakang

glClearColor(0.1, 0.1, 0.2, 0.0);

glEnable(GL\_DEPTH\_TEST); // mengaktifkan depth buffer

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45.0, 1.0, 1.0, 100.0);// set proyeksi ke perspektif

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// inisialisasi kamera pandang

gluLookAt(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

// inisialisasi pencahayaan

glLightfv(GL\_LIGHT0, GL\_AMBIENT, light\_ambient);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light\_diffuse);

glLightfv(GL\_LIGHT0, GL\_SPECULAR, light\_specular);

glLightfv(GL\_LIGHT1, GL\_AMBIENT, light\_ambient);

glLightfv(GL\_LIGHT1, GL\_DIFFUSE, light\_diffuse);

glLightfv(GL\_LIGHT1, GL\_SPECULAR, light\_specular);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat\_specular);

glMaterialfv(GL\_FRONT, GL\_AMBIENT, mat\_ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, mat\_diffuse);

glMaterialf(GL\_LIGHT0, GL\_SHININESS, mat\_shininess);

glMaterialf(GL\_LIGHT1, GL\_SHININESS, mat\_shininess);

glShadeModel(GL\_SMOOTH); // aktifkan smooth shading

glEnable(GL\_LIGHTING); // aktifkan pencahayaan

glEnable(GL\_LIGHT0); // aktifkan sumber cahaya 0

glEnable(GL\_LIGHT1); // aktifkan sumber cahaya 1

}

// fungsi ini digunakan bila layar akan diresize (default)

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45, (GLfloat)w / (GLfloat)h, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

}

//langkah 5

// fungsi untuk mengatur masukan dari keyboard

// untuk arah kiri, kanan, atas, bawah, PgUp, dan PgDn

void keyboard(int key, int x, int y)

{

float fraction = 0.1f;

switch (key)

{

// masukkan perintah disini bila tombol kiri ditekan

case GLUT\_KEY\_LEFT:

// dalam hal ini perintah rotasi obyek ke kiri sebanyak 1 derajat

objectAngleY -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol kanan ditekan

case GLUT\_KEY\_RIGHT:

// dalam hal ini perintah rotasi obyek ke kanan sebanyak 1 derajat

objectAngleY += 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol atas ditekan

case GLUT\_KEY\_UP:

// dalam hal ini perintah rotasi obyek ke atas sebanyak 1 derajat

objectAngleX -= 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol bawah ditekan

case GLUT\_KEY\_DOWN:

// dalam hal ini perintah rotasi obyek ke bawah sebanyak 1 derajat

objectAngleX += 1.0f;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol PgUp ditekan

case GLUT\_KEY\_PAGE\_UP:

posX += rotX \* fraction;

posZ += rotZ \* fraction;

glutPostRedisplay(); // update obyek

break;

// masukkan perintah disini bila tombol PgDn ditekan

case GLUT\_KEY\_PAGE\_DOWN:

posX -= rotX \* fraction;

posZ -= rotZ \* fraction;

glutPostRedisplay(); // update obyek

case GLUT\_KEY\_F1:

subdiv++; // lakukan subdivisi

glutPostRedisplay(); // update obyek

break;

case GLUT\_KEY\_F2:

subdiv--; // lakukan subdivisi

glutPostRedisplay(); // update obyek

break;

}

if (subdiv<0) subdiv = 0;

}

// timer untuk animasi (gunakan bila perlu)

void timer(int value)

{

glutPostRedisplay();

glutTimerFunc(55, timer, 0);

}

// program utama

int main(int argc, char\*\* argv)

{

// inisialisasi jendela OpenGL

// GLUT\_SINGLE berarti memakai single buffer

// GLUT\_DOUBLE berarti memakai double buffer

// GLUT\_RGB berarti mode tampilan yang dipakai RGB

// GLUT\_RGBA berarti mode tampilan yang dipakai RGBA

// GLUT\_DEPTH berarti memakai depth buffer

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA | GLUT\_DEPTH);

// set ukuran jendela tampilan

glutInitWindowSize(480, 480); // besarnya jendela dalam piksel dalam hal ini 300x300

glutInitWindowPosition(100, 100); // posisi jendela dilayar komputer dalam piksel

// judul jendela (wajib diubah dengan informasi NAMA / NIM - JUDUL PRAKTIKUM masing-masing)

glutCreateWindow("SIRLY ZIADATUL MUSTAFIDAH / 2200018345 - TEKNIK SUBDIVISI");

// panggil fungsi init untuk inisialisasi awal

init();

// event handler untuk display, reshape dan keyboard

glutDisplayFunc(display); // display

glutReshapeFunc(reshape); // reshape

glutSpecialFunc(keyboard); // keyboard

//glutTimerFunc(0, timer, 0); // aktifkan timer bila perlu

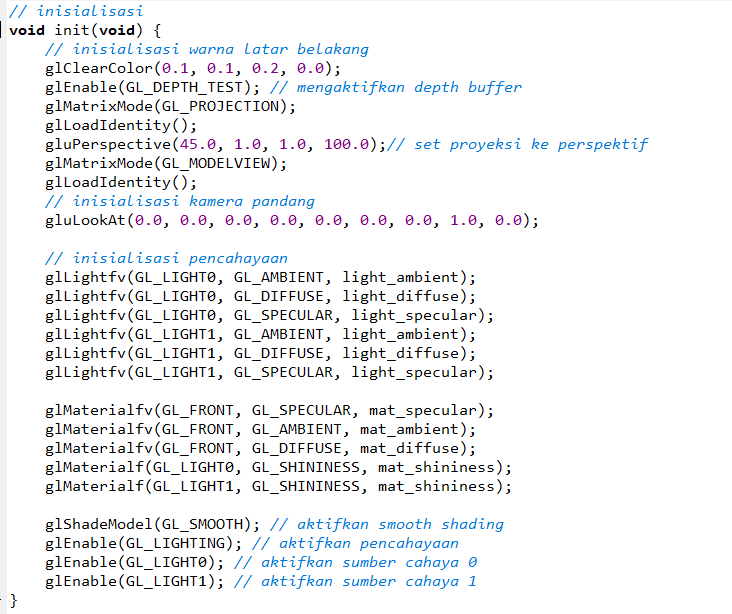
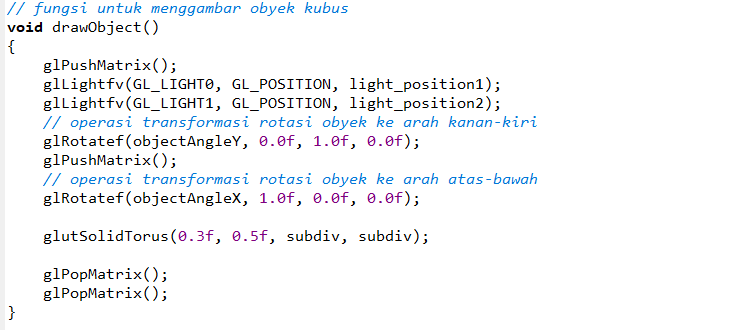
// looping

glutMainLoop();

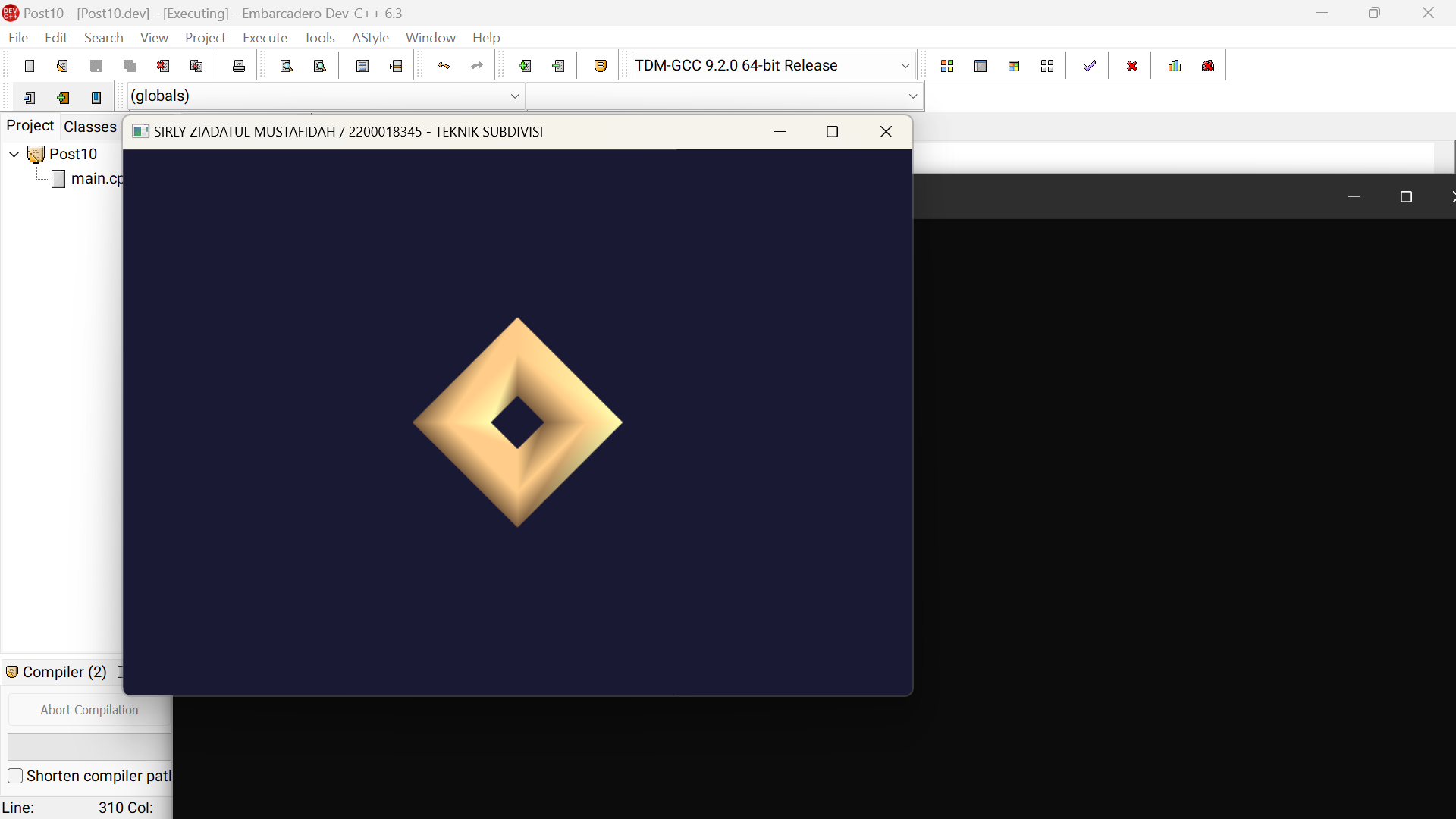
return 0;

}

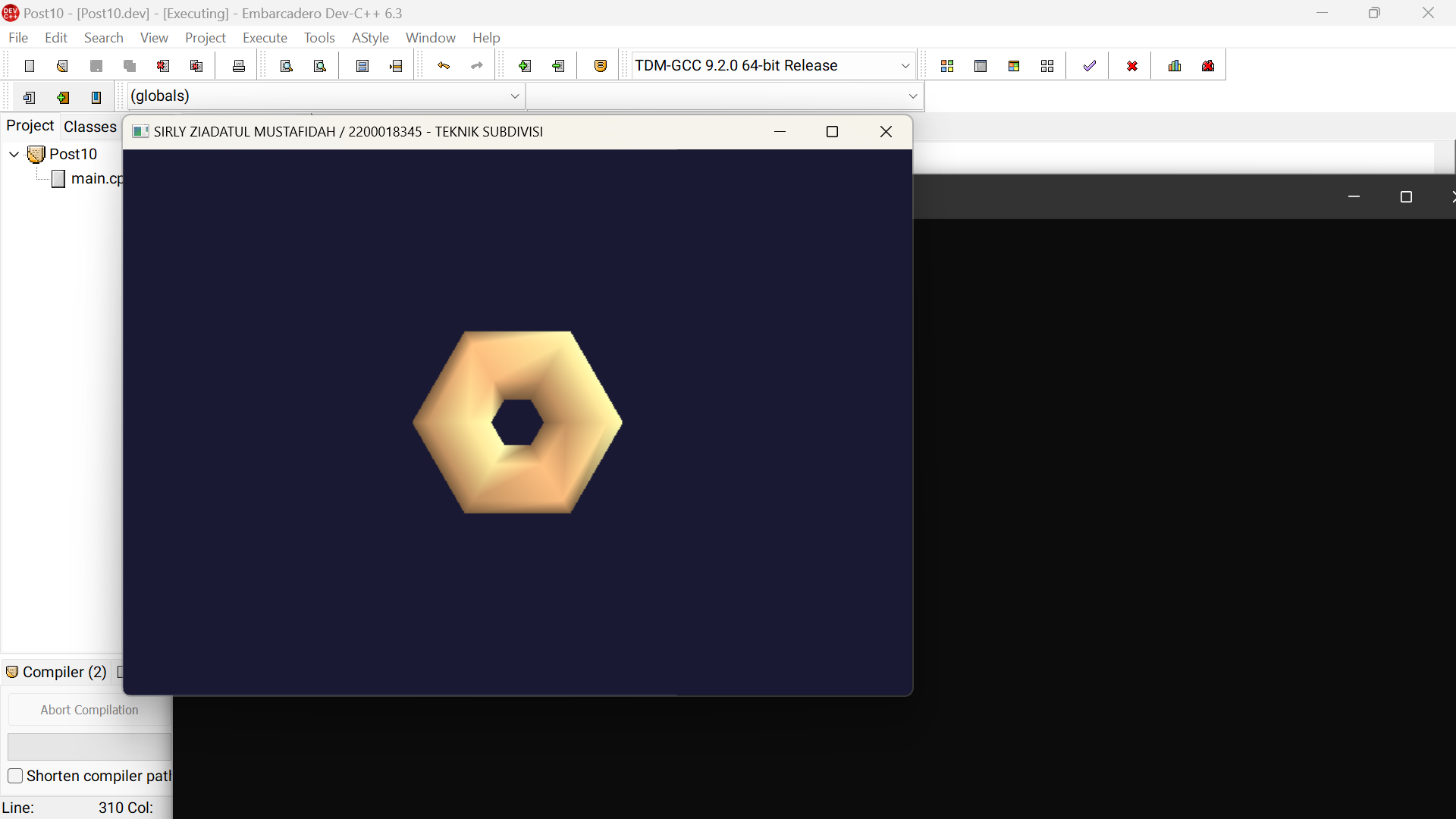
Penjelasan



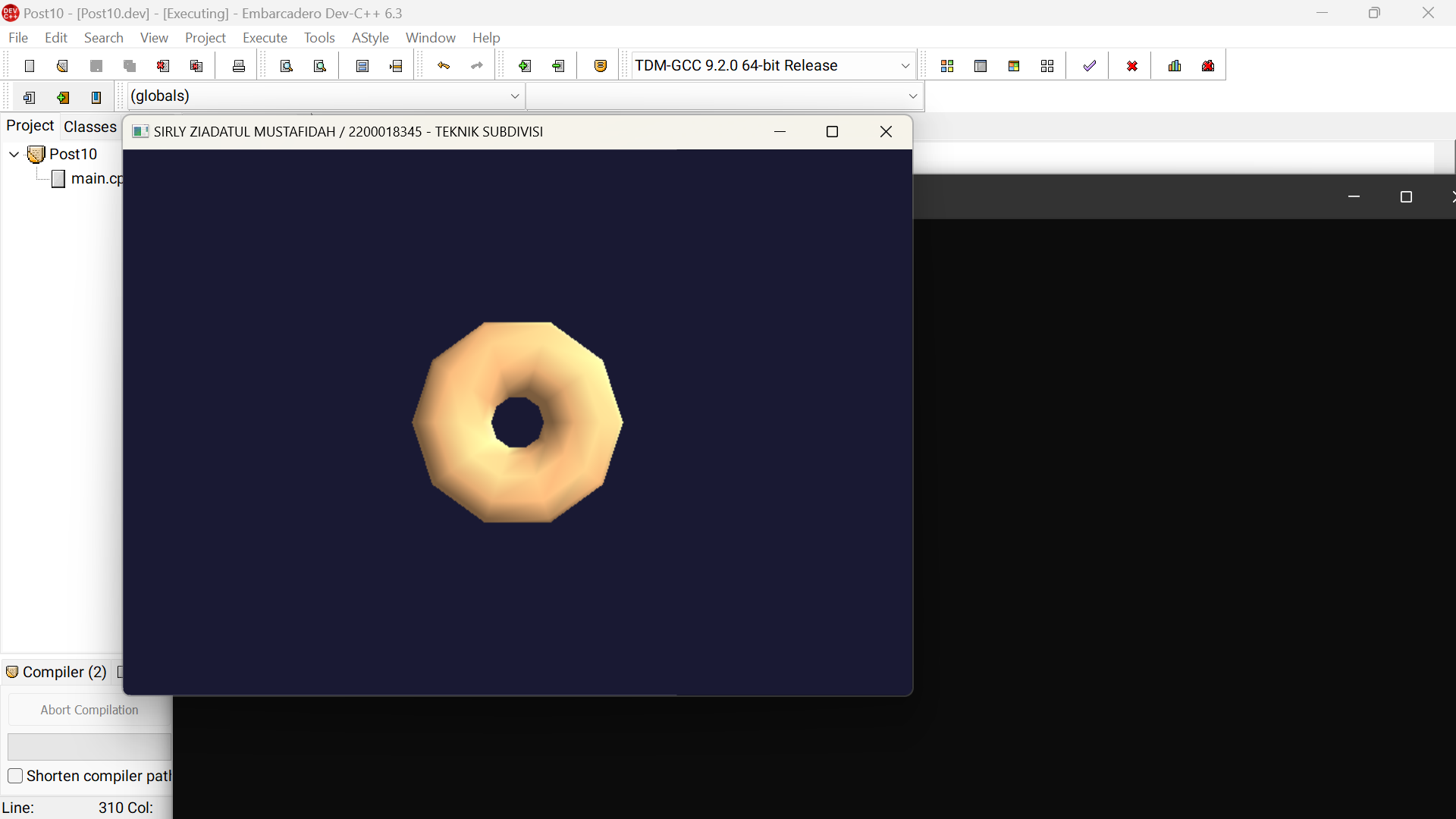
Output



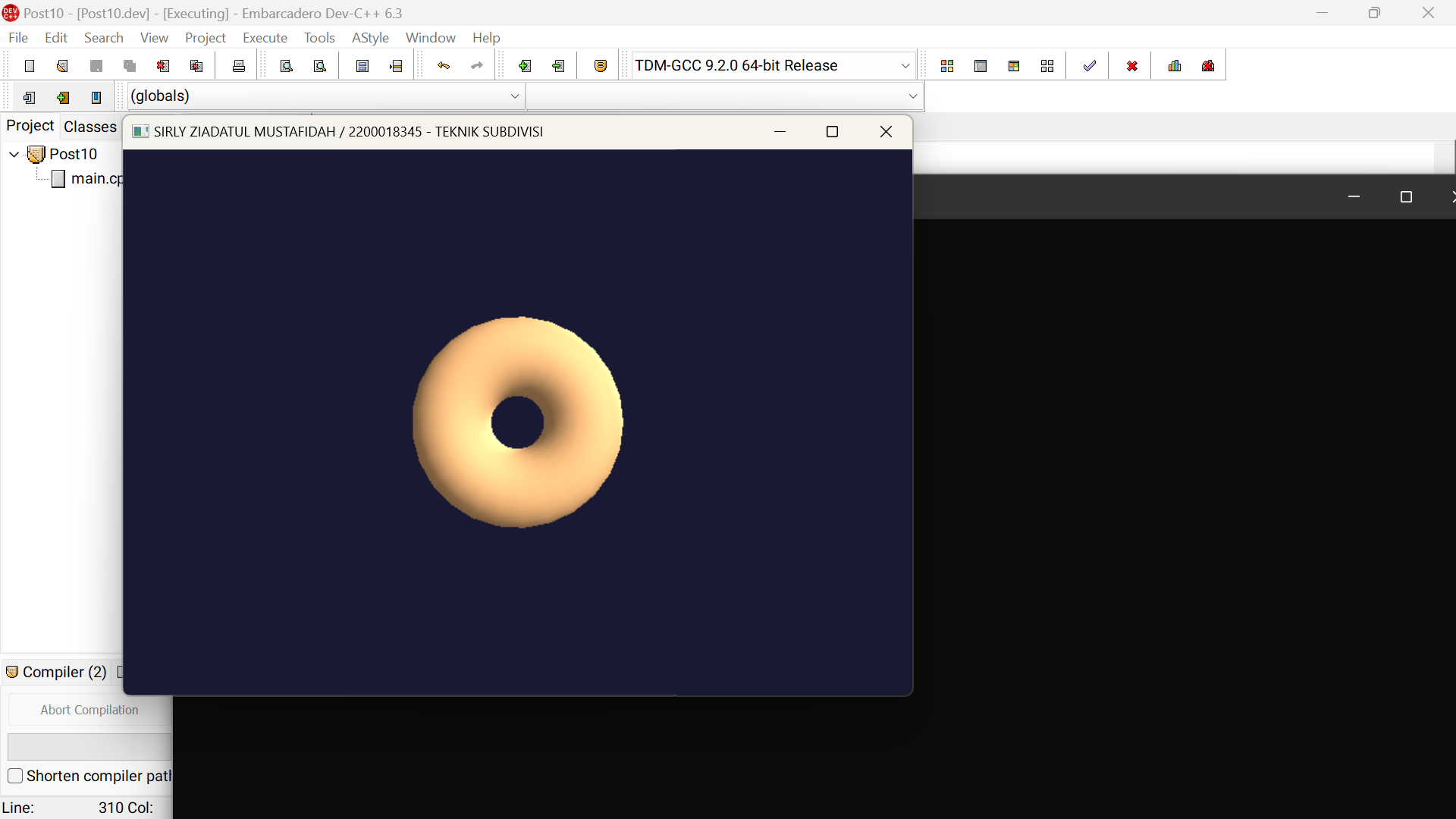
Gambar 10. 5



Gambar 10. 6



Gambar 10. 7



Gambar 10. 8

# DAFTAR PUSTAKA

Adnani, M., & Falani, A. Z. (2021). Implementasi Open Gl Untuk Pembuatan Objek 3d. *Zetroem Vol 03. No 01*, 2-5.

Akbar, A. M. (2014, 04 15). *Pengantar Pemrograman OpenGL GLUT C++ di Linux*. Diambil kembali dari Wordpress: https://malsasa.wordpress.com/2014/04/15/pengantar-pemrograman-opengl-glut-c-di-linux/

Gordong, V. S., & Clevenger, J. (2024). *Computer Graphics Programming in OpenGL with C++.* Bostom: David Pallai.

Kosarevsky, S., & Latypov, V. (2021). *3D Graphics Rendering Cookbook.* Birmingham: Packt Publishing Ltd.

Lehn, K., Gotzes, M., & Klawonn, F. (2023). *Introduction to Computer Graphics Third Edition.* Germany: Springer Nature Switzerland AG.

Mukundan, R. (2022). *3D Mesh Processing and Character Animation With Examples Using OpenGL, OpenMesh and Assimp.* New Zealand: Switzerland AG.

Zailani, A. U., Sibarani, H. S., & Perdananto, A. (2021). *KOMPUTER GRAFIK.* Tangerang Selatan: Unpam Press.