LAPORAN PRATIKUM GRAFIK KOMPUTER

Diajukan untuk memenuhi Tugas mata kuliah Pratikum Grafik Komputer

IMPLEMENTASI OPENGL UNTUK PEMBUATAN VISUALISASI 3D PYRAMID OF GIZA

Dosen Pengampu : Sri Rahayu, M.Kom

Instruktur Pratikum : Arul Budi Kalimat, S.Kom

Disusun oleh

Kelompok : 9

Kamila Lutfi Zakiah
NIM 2306108

Fadilah Nur Fatimah
NIM 2306109

Muhamad Imam Asidiq

NIM 2306101

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN ILMU KOMPUTER

INSTITUT TEKNOLOGI GARUT

2024

KATA PENGANTAR

Puji dan syukur kehadirat Allah SWT atas segala rahmat dan karunia-Nya sehingga kami dapat menyelesaikan Laporan Praktikum Jaringan Komputer ini. Laporan ini dibuat sebagai salah satu tugas dari mata kuliah Jaringan Komputer, dengan tujuan untuk memberikan pemahaman yang lebih baik tentang “Implementasi Opengl Untuk Pembuatan Visualisasi 3d Pyramid Of Giza”.

Kami mengucapkan terima kasih kepada dosen pengampu Sri Rahayu, M.Kom, instruktur praktikum Arul Budi Kalimat, S.Kom, serta semua pihak yang telah memberikan dukungan dalam penyusunan laporan ini.

Kami menyadari bahwa laporan ini masih memiliki kekurangan, untuk itu kami mengharapkan kritik dan saran yang membangun demi perbaikan di masa yang akan datang.

Garut, 10 Januari 2025

Penulis

DAFTAR ISI

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan zaman yang disertai oleh perkembangan teknologi komputer grafika dan pencitraan tiga dimensi saat ini telah mengalami kemajuan yang sangat pesat dengan tingkat kualitas dan pencapain yang cukup signifikan dalam dekade terakhir. Grafika komputer merupakan teknik dalam ilmu komputer dan matematika untuk merepresentasikan dan memanipulasikan gambar menggunakan komputer.

Grafika komputer(Computer Graphic) dapat diartikan sebagai seperangkat alat yang terdiri dari hardware dan software untuk membuat gambar, grafik atau citra realistik untuk seni, game komputer, foto dan film animasi. Sistem grafika komputer dapat dijalankan dengan komputer pribadi (Personal Computer) atau workstation. Grafika komputer semakin lama semakin pesat perkembangannya sehingga definisi dari grafika komputer dapat diartikan sebagai suatu studi tantang bagaimana menggambar (membuta grafik) dengan menggunakan komputer dan manipulasinya (merubah sedikit/transformasi/ animasi) [1].

Salah satu aplikasi yang komplit dari grafika komputer adalah untuk visualisasi data dalam bentuk grafika dua dimensi (2D), atau tiga dimensi (3D), yang merupakan sebuah objek yang memiliki Panjang, lebar, dan tinggi. Pada objek ini memiliki titik koordinat sumbu x sebagai bagian datar, titik koordinat sumbu y sebagai sebagai sumbu tegak, dan titik koordinat sumbu z sebagai sumbu lurus, agar hasil menjadi maksimal maka konfigurasi sudut x,y,z minimal 0 dan maksimal sebesar resolusi yang digunakan[2].

Diperlukan sebuah perangkat lunak yang mampu menampilkan visual dalam format tiga dimensi (3D) secara real-time, dengan memasukkan data atau parameter yang dihasilkan oleh perangkat eksternal. Perangkat lunak ini harus dapat menampilkan hasilnya langsung pada layar monitor tanpa melalui proses rendering ke dalam file terlebih dahulu.

Untuk mewujudkannya, OpenGL (Open Graphic Library) dapat digunakan. Open Graphics Library (OpenGL) merupakan perangkat lunak yang digunakan untuk mengembangkan aplikasi grafis 2D dan 3D. Sejak diperkenalkan pada tahun 1992 oleh Silicon Graphics Inc. (SGI), OpenGL telah digunakan secara luas di berbagai industri yang bergerak di bidang grafis, seperti CAD, Virtual Reality, Visualisasi Ilmiah, Visualisasi Informasi, dan Simulasi Penerbangan. Perangkat lunak ini mendukung format lintas-bahasa dan lintas-platform API, serta menyediakan lebih dari 250 antarmuka (interface) yang dapat digunakan untuk membangun bentuk tiga dimensi yang kompleks dari bentuk-bentuk sederhana. Sebagai

API perangkat lunak, OpenGL dirancang untuk efisiensi perangkat keras di berbagai platform. Selain itu, OpenGL Utility Library (GLU) menyediakan banyak fitur pemodelan, seperti permukaan quadric dan NURBS curves [3].

Bahasa pemrograman yang sering digunakan untuk mengembangkan aplikasi berbasis OpenGL adalah C++. Bahasa C++ diciptakan oleh Bjarne Stroustrup di AT&T Bell Laboratories awal tahun 1980-an berdasarkan C ANSI(American National Standard Institute). Pertama kali, prototype C++ muncul sebagai C yang dipercanggih dengan fasilitas kelas. Bahasa tersebut disebut C dengan kelas (C wih class). Selama tahun 1983-1984, C dengan kelas disempurnakan dengan menambah fasilitas pembeban lebihan operator dan fungsi yang kemudian melahirkan apa yang disebut C++ [4].

Ada beberapa perangkat lunak yang dapat digunakan untuk membuat program dengan bahasa C dan C++, antara lain:

1. Turbo C++
2. Borland C++
3. Dev-C++
4. GCC (GNU *Compiler Collection*)

Dalam pengimplementasian projek ini kami menggunakan aplikasi Dev C++, Salah satu kelebihan dari perangkat lunak ini adalah sifatnya yang *open source*, sehingga memungkinkan siapa saja untuk meningkatkan (meng-upgrade) aplikasi ini. Selain itu, perangkat lunak ini juga bersifat *freeware* (gratis) [4].

1.2 Rumusan Masalah

Penggunaan teknologi grafis, seperti OpenGL, telah menjadi bagian penting dalam pengembangan aplikasi 2D dan 3D. Untuk memanfaatkan teknologi ini secara efektif, diperlukan pemahaman tentang konsep dasar, konfigurasi, dan cara kerja OpenGL, terutama ketika diintegrasikan dengan perangkat lunak seperti Dev C++. Oleh karena itu, untuk mencapai tujuan tersebut, terdapat beberapa pertanyaan utama yang menjadi dasar rumusan masalah, yaitu:

1. Apa yang dimkasud dengan OpenGL ?
2. Bagaimana cara mengkonfigurasi OpenGL pada Dev C++
3. Bagaimana cara kerja dari OpenGL ?
4. Bagaimana membuat ***Pyramid Of Giza*** dalam OpenGL

1.3 Tujuan

Tujuan dari laporan praktikum :

1. Mengethaui apa itu OpenGL
2. Mengetahui cara mengkonfigurasi OpenGL pada Dev C++
3. Mengetahui cara kerja dari OpenGL
4. Mengetahui cara pembuatan ***Pyramid Of Giza*** dalam OpenGL

BAB II
TINJAUAN PUSTAKA

2.1 OpenGL

OpenGL (*Open Graphic Library*) merupakan *library* yang terdiri dari berbagai macam fungsi dan biasanya digunakan untuk menggambar beberapa objek 2Ddan 3D. Library-libraryini mendefinisikan sebuah cross-bahasa, cross-platform API (antarmuka pemrograman aplikasi). OpenGL merupakan library yang digunakan untuk melakukan pemrograman grafik; Graphic Programming [5].

OpenGL adalah sebuah library grafis yang bersifat open-source, multi-platform, dan mendukung berbagai bahasa pemrograman. OpenGL dapat dijalankan di berbagai sistem operasi, seperti Windows, Unix, SGI, Linux, FreeBSD, dan lainnya. Library dasar OpenGL adalah GLUT (OpenGL Utility Toolkit), yang menyediakan berbagai fungsi untuk mempermudah pengembangan aplikasi grafis. GLUT memiliki fasilitas dasar yang dapat dikembangkan lebih lanjut sesuai kebutuhan. Untuk sistem operasi Windows, library ini terdiri dari 3 files yaitu:

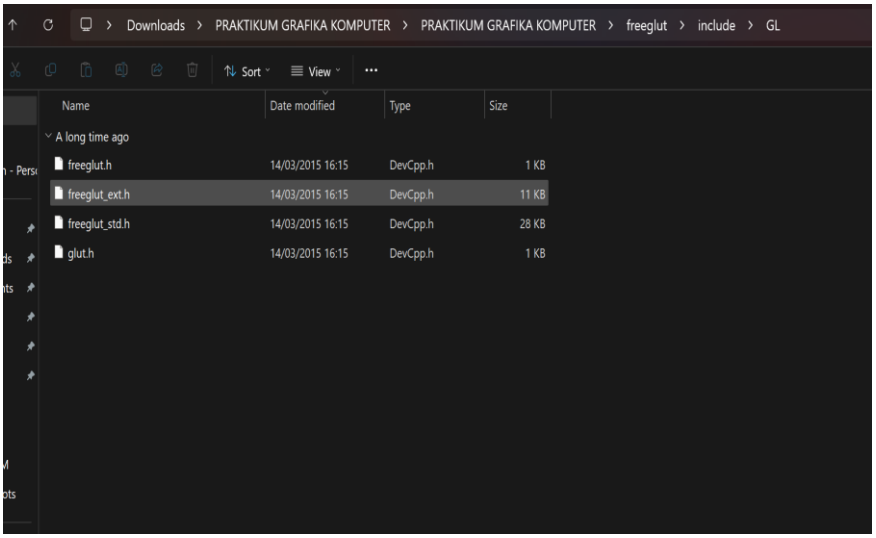
1. glut.h
2. glut32.lib
3. glut32.dll

Saat ini, OpenGL didukung oleh hampir semua bahasa pemrograman [6]

2.2 Konfigurasi OpenGL pada Dev C++

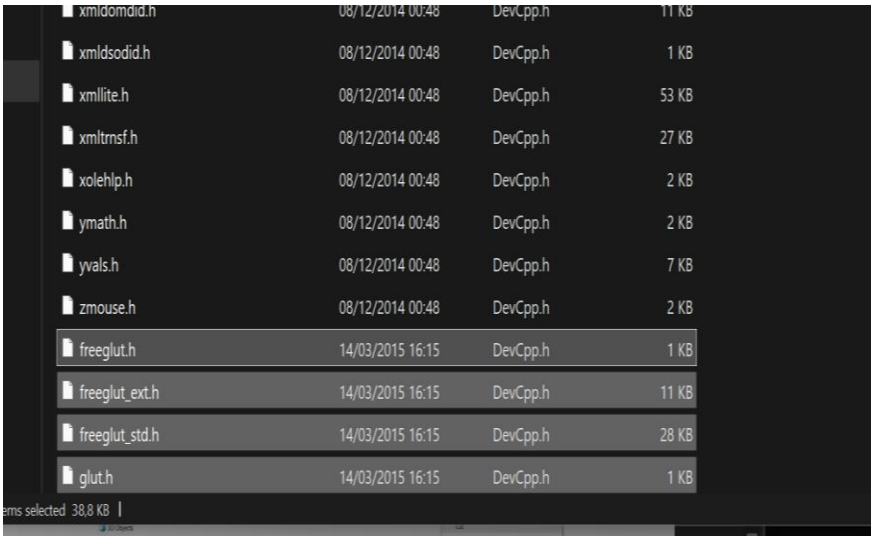
langkah-langkah untuk konfigurasi OpenGL pada Dev C++.

1. Buka File Freeglut yang bisa di akses di link: https://s.id/l5RKg
2. Buka folder freeglut tersebut
3. Copy semua file yang ada di folder freeglut\include\GL



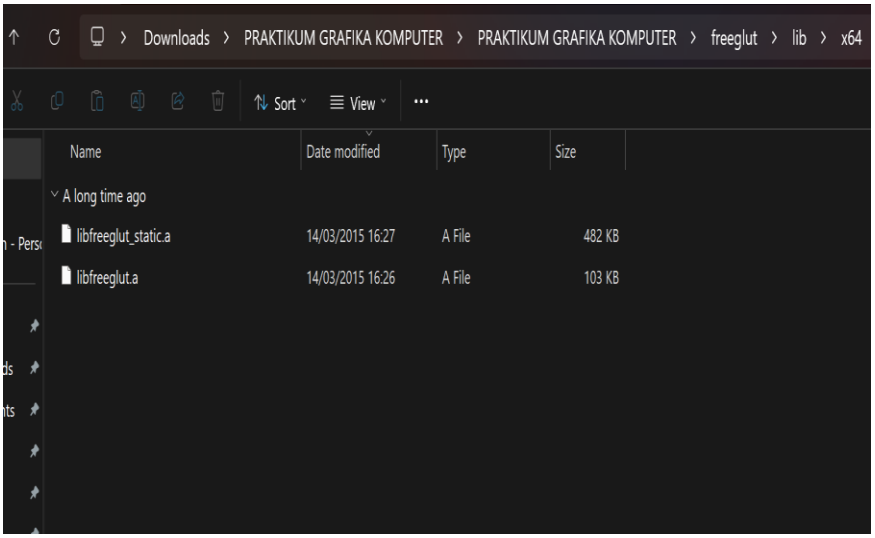
Gambar 2. 1 Tampilan folder Freeglut/include/GL

4. Pastekan file tersebut ke : C:\Program Files (x86)\Dev-Cpp\MinGW64\x86_64-w64-mingw32\include\GL



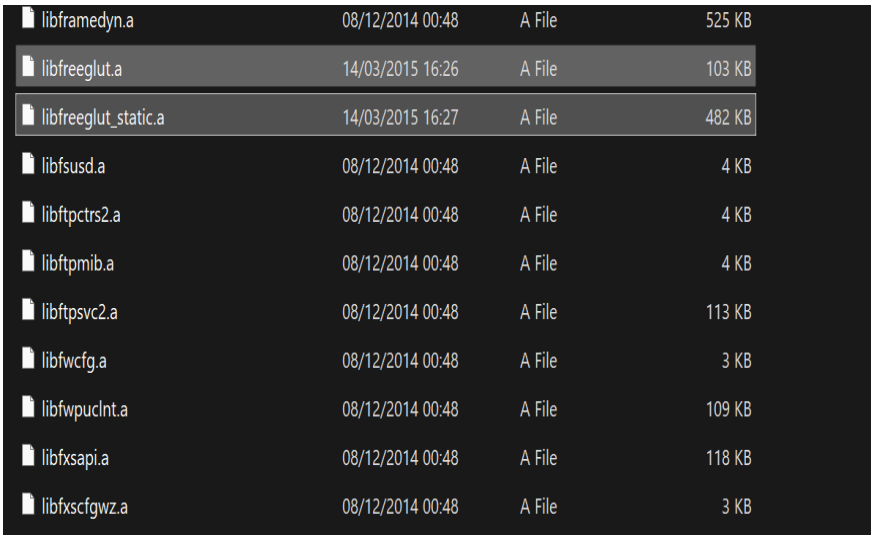
Gambar 2. 2 File di Folder include\GL

5. Kemudian kembali buka folder freeglut, kemudian masuk ke folder lib, lalu buka folder x64, lalu copy seluruh file di dalamnya.



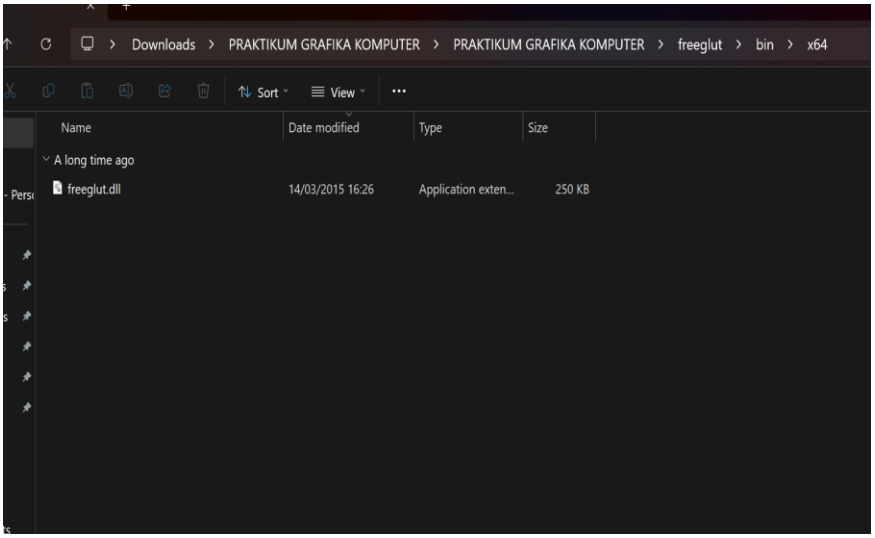
Gambar 2. 3 folder freeglut lib

6. Pastekan file yang di copy ke : C:\Program Files (x86)\Dev-Cpp\MinGW64\x86_64-w64-mingw32\lib



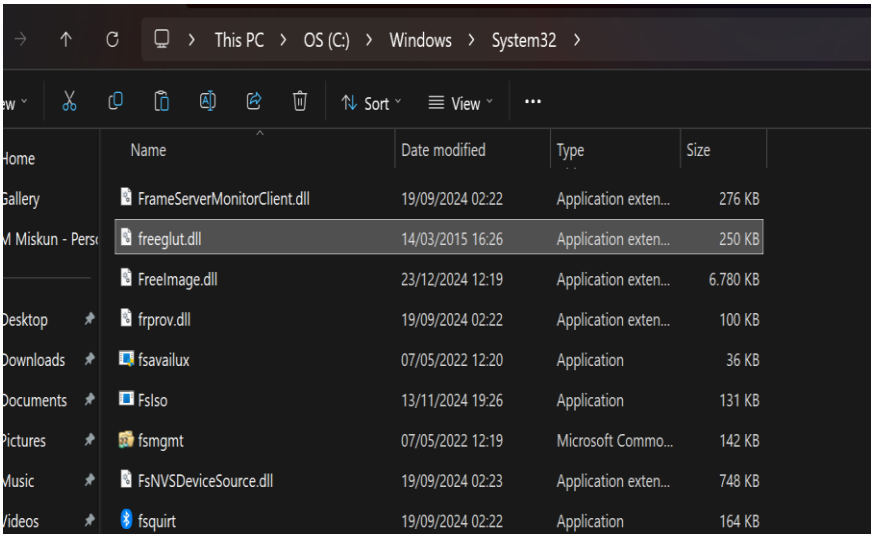
Gambar 2. 4 file lib yang di copy

7. Kemudian kembali buka folder freeglut, kemudian masuk ke folder bin, lalu buka folder x64, lalu copy seluruh file “freeglut.dll”



Gambar 2. 5 folder freeglut bin

8. Pastekan ke lokasi folder : C:\Windows\System32

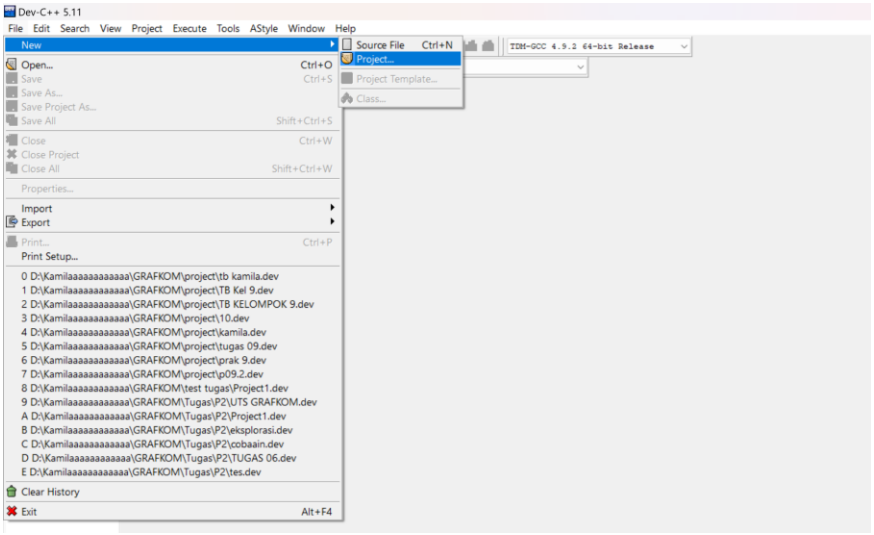


Gambar 2. 6 File bin yang di copy

2.3 Cara Kerja OpenGL

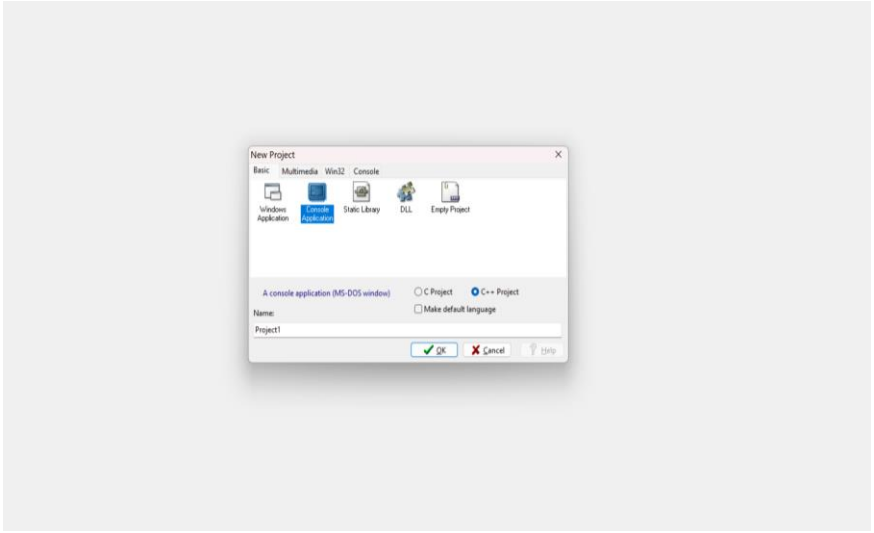
Langkah-langkah untuk mengatur dan menjalankan OpenGL menggunakan Dev-C++

1. Buka aplikasi Dev C yang sudah terkonfigurasi, kemudian pilih file-New Project



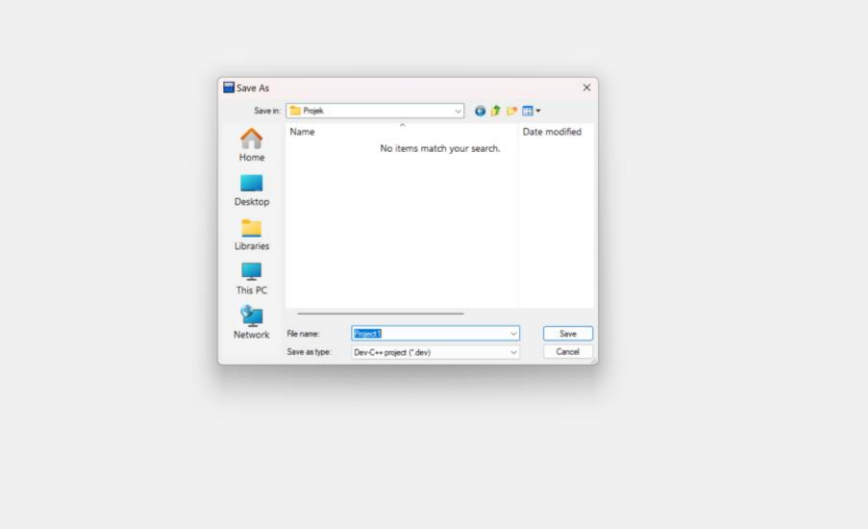
Gambar 2. 7 Tampilan awal Dev-C++

2. Pilih *console application, Rename* file kemudian klik Oke, kemudian akan diminta lokasi penyimpanan file, jika sudah klik save



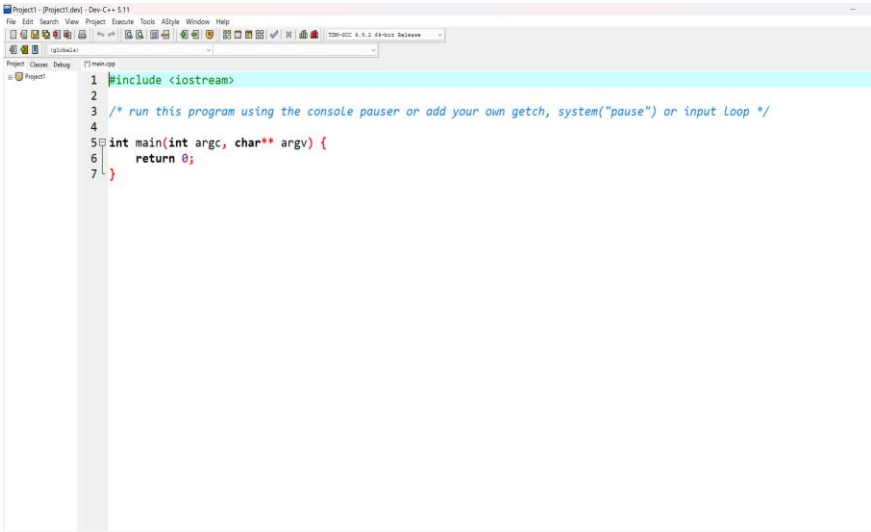
Gambar 2. 8 Penamaan File

3. kemudian akan diminta lokasi penyimpanan file, jika sudah klik save



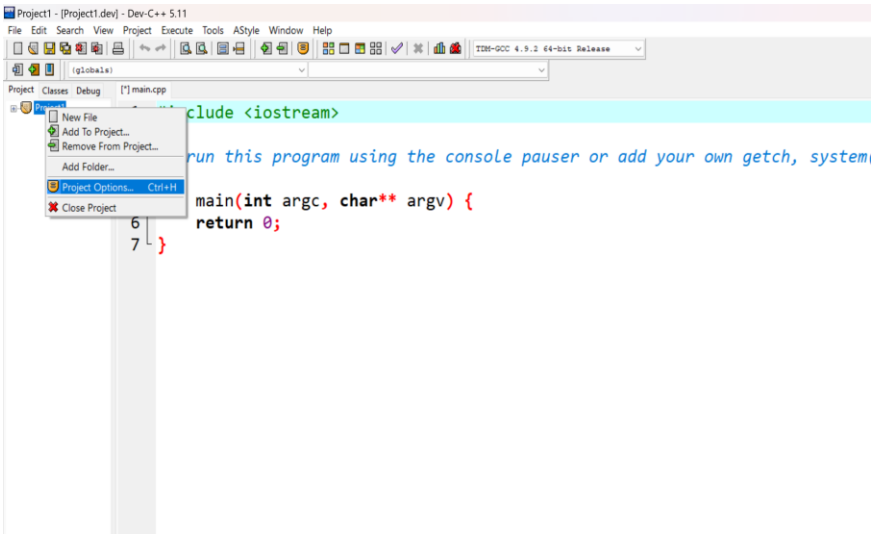
Gambar 2. 9 Pemilihan lokasi penyimpanan

4. Lalu akan muncul tampilan area kerja seperti berikut



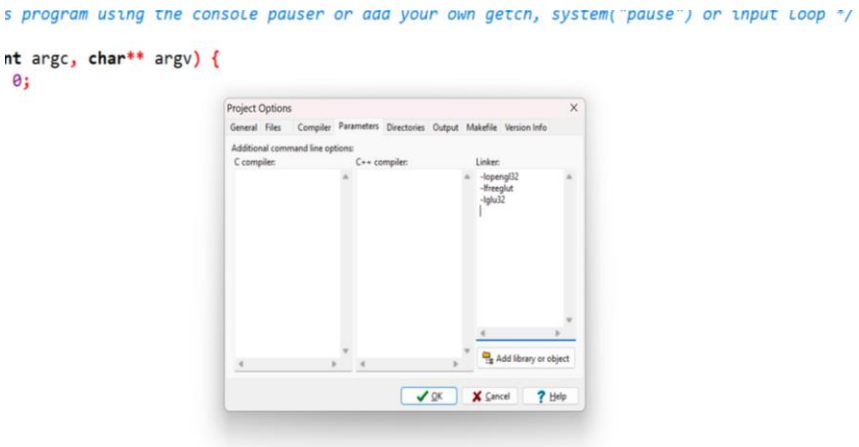
Gambar 2. 10 Tampilan area kerja

5. Kemudian pilih menu Project, dilanjutkan pilih project options



Gambar 2. 11 Pembuatan *Project* baru

6. Pada Project Options, pilih Parameter, dan pada bagian linker ketik perintah berikut:
   -lopengl32
   -lfreeglut
   -lglu32
   kemudian klik Oke.



Gambar 2. 12 Pemasukan Parameter

7. Kemudian coba masukkan source code ke area kerja

2.4 Pembuatan *Pyramid Of Giza* Di OpenGL

Pembuatan *Pyramid* dalam OpenGL dimulai dengan mendefinisikan geometrinya. Secara konsep, *Pyramid* dibangun dari empat segitiga untuk sisi-sisi tegak dan satu segi empat sebagai alas. Dalam implementasinya di OpenGL, setiap bidang ini direpresentasikan melalui

serangkaian *vertex*. *Vertex* adalah titik-titik sudut dalam ruang tiga dimensi, yang posisinya ditentukan oleh koordinat (x, y, z). Selain *vertex*, setiap bidang juga memiliki normal, yaitu vektor yang tegak lurus terhadap permukaan bidang tersebut. Normal ini krusial untuk perhitungan pencahayaan, karena menentukan bagaimana cahaya berinteraksi dengan permukaan objek.

Setelah geometri objek *Pyramid* ditetapkan, OpenGL menyediakan fungsi-fungsi transformasi seperti glTranslatef() untuk memindahkan objek, glRotatef() untuk memutarnya, dan glScalef() untuk mengubah ukurannya. Transformasi-transformasi ini diterapkan pada *matriks modelview*, yang mengontrol bagaimana objek diposisikan dan diorientasikan dalam ruang virtual. Untuk meningkatkan realisme visual, efek pencahayaan diterapkan dengan mendefinisikan karakteristik sumber cahaya (misalnya, posisi dan warna), serta sifat-sifat material objek, termasuk warna *ambient*, *diffuse*, dan *specular*. OpenGL kemudian menghitung bagaimana cahaya berinteraksi dengan permukaan objek berdasarkan vektor normal di setiap *vertex*.

Dengan memadukan representasi geometris, transformasi objek, dan efek pencahayaan, OpenGL mampu merender piramida dalam sebuah adegan dengan tampilan tiga dimensi yang sangat realistis.

BAB III
HASIL

3.1 Source Code

Untuk Source code kalian masukan source code yang telah kalian buat lalu masukan ke dalam kolom dibawah ini :

```
#include <GL/glut.h>                    //FADILAH

float rotateX = 0.0, rotateY = 0.0; // Variabel untuk rotasi
float zoom = -8.0; // Variabel untuk zoom
bool showCarte = true; // Variabel untuk mengontrol apakah garis
kartesius ditampilkan


void drawPyramid(float x, float y, float z, float scale) {
      //FADILAH
    glPushMatrix();
    glTranslatef(x, y, z);
    glScalef(scale, scale, scale);

      glEnable(GL_LIGHTING);
      glEnable(GL_NORMALIZE);

    glBegin(GL_TRIANGLES);
    glColor3f(0.82, 0.71, 0.55);

    // Sisi depan (normal menghadap ke depan)
    glNormal3f(0.0, 0.5, 1.0); // Normal untuk sisi depan
    glVertex3f(0.0, 2.0, 0.0);
    glVertex3f(-1.5, -1.0, 1.5);
    glVertex3f(1.5, -1.0, 1.5);

    // Sisi kanan (normal menghadap ke kanan)
    glNormal3f(1.0, 0.5, 0.0); // Normal untuk sisi kanan
    glVertex3f(0.0, 2.0, 0.0);
    glVertex3f(1.5, -1.0, 1.5);
    glVertex3f(1.5, -1.0, -1.5);

    // Sisi belakang (normal menghadap ke belakang)
    glNormal3f(0.0, 0.5, -1.0); // Normal untuk sisi belakang
    glVertex3f(0.0, 2.0, 0.0);
    glVertex3f(1.5, -1.0, -1.5);
    glVertex3f(-1.5, -1.0, -1.5);

    // Sisi kiri (normal menghadap ke kiri)
    glNormal3f(-1.0, 0.5, 0.0); // Normal untuk sisi kiri
    glVertex3f(0.0, 2.0, 0.0);
    glVertex3f(-1.5, -1.0, -1.5);
    glVertex3f(-1.5, -1.0, 1.5);

    glEnd();

    // Dasar Pyramid (normal menghadap ke bawah)
    glBegin(GL_QUADS);
```

```
    glNormal3f(0.0, -1.0, 0.0); // Normal untuk dasar
    glVertex3f(-1.5, -1.0, 1.5);
    glVertex3f(1.5, -1.0, 1.5);
    glVertex3f(1.5, -1.0, -1.5);
    glVertex3f(-1.5, -1.0, -1.5);
    glEnd();

     glDisable(GL_NORMALIZE);
    glPopMatrix();
}

void drawMatahari(float x, float y, float z, float scale) {
     //FADILAH
    glPushMatrix();
    glTranslatef(x, y, z);
    glScalef(scale, scale, scale);
    glColor3f(1.0, 0.84, 0.0); // Warna kuning untuk matahari
    glutSolidSphere(0.5, 50, 50);
    glPopMatrix();
}

void drawBatu(float x, float y, float z, float scale) {
           //KAMILA
    glPushMatrix();
    glTranslatef(x, y, z);
    glScalef(scale, scale, scale);
    glColor3f(0.5, 0.5, 0.5); // Warna abu-abu gelap untuk batu
    glutSolidSphere(0.5, 20, 20);
    glPopMatrix();
}

void drawPohon(float x, float y, float z) {
     //KAMILA
    glPushMatrix();
    glTranslatef(x, y, z);

    // Batang pohon (lebih kecil)
    glColor3f(0.55, 0.27, 0.07); // Warna coklat untuk batang
    glBegin(GL_QUADS);
    glVertex3f(-0.05, 0.0, -0.05);
    glVertex3f(0.05, 0.0, -0.05);
    glVertex3f(0.05, 1.0, -0.05); // Tinggi batang menjadi 1.0
    glVertex3f(-0.05, 1.0, -0.05);

    glVertex3f(-0.05, 0.0, 0.05);
    glVertex3f(0.05, 0.0, 0.05);
    glVertex3f(0.05, 1.0, 0.05);
    glVertex3f(-0.05, 1.0, 0.05);

    glVertex3f(-0.05, 0.0, -0.05);
    glVertex3f(-0.05, 0.0, 0.05);
    glVertex3f(-0.05, 1.0, 0.05);
    glVertex3f(-0.05, 1.0, -0.05);

    glVertex3f(0.05, 0.0, -0.05);
    glVertex3f(0.05, 0.0, 0.05);
    glVertex3f(0.05, 1.0, 0.05);
```

```
    glVertex3f(0.05, 1.0, -0.05);
    glEnd();

    // Daun kelapa (posisi dan dimensi disesuaikan)
    glColor3f(0.0, 0.5, 0.0); // Warna hijau untuk daun
    glBegin(GL_TRIANGLES);

    // Daun pertama (kiri depan)
    glVertex3f(0.0, 1.0, 0.0); // Dasar daun tepat di ujung batang
    glVertex3f(-0.7, 1.2, -0.3); // Ujung daun lebih panjang
    glVertex3f(-0.4, 1.2, -0.6);

    // Daun kedua (kanan depan)
    glVertex3f(0.0, 1.0, 0.0);
    glVertex3f(0.7, 1.2, -0.3);
    glVertex3f(0.4, 1.2, -0.6);

    // Daun ketiga (kiri belakang)
    glVertex3f(0.0, 1.0, 0.0);
    glVertex3f(-0.7, 1.2, 0.3);
    glVertex3f(-0.4, 1.2, 0.6);

    // Daun keempat (kanan belakang)
    glVertex3f(0.0, 1.0, 0.0);
    glVertex3f(0.7, 1.2, 0.3);
    glVertex3f(0.4, 1.2, 0.6);

    glEnd();

    glPopMatrix();
}

void drawLantai() {                                //FADILAH
    glPushMatrix();

    glColor3f(0.96, 0.87, 0.70); // Warna krem untuk lantai

    // Gambarkan lantai dengan normal untuk setiap sisi
    glBegin(GL_QUADS);

    // Sisi atas (normal menghadap ke atas)
    glNormal3f(0.0, 1.0, 0.0);
    glVertex3f(-5.0, -1.0, 5.0);
    glVertex3f(5.0, -1.0, 5.0);
    glVertex3f(5.0, -1.0, -5.0);
    glVertex3f(-5.0, -1.0, -5.0);

    // Sisi bawah (normal menghadap ke bawah)
    glNormal3f(0.0, -1.0, 0.0);
    glVertex3f(-5.0, -1.5, 5.0);
    glVertex3f(5.0, -1.5, 5.0);
    glVertex3f(5.0, -1.5, -5.0);
    glVertex3f(-5.0, -1.5, -5.0);

    // Sisi depan (normal menghadap ke depan)
    glNormal3f(0.0, 0.0, 1.0);
    glVertex3f(-5.0, -1.0, 5.0);
```

```
    glVertex3f(5.0, -1.0, 5.0);
    glVertex3f(5.0, -1.5, 5.0);
    glVertex3f(-5.0, -1.5, 5.0);

    // Sisi belakang (normal menghadap ke belakang)
    glNormal3f(0.0, 0.0, -1.0);
    glVertex3f(-5.0, -1.0, -5.0);
    glVertex3f(5.0, -1.0, -5.0);
    glVertex3f(5.0, -1.5, -5.0);
    glVertex3f(-5.0, -1.5, -5.0);

    // Sisi kiri (normal menghadap ke kiri)
    glNormal3f(-1.0, 0.0, 0.0);
    glVertex3f(-5.0, -1.0, 5.0);
    glVertex3f(-5.0, -1.0, -5.0);
    glVertex3f(-5.0, -1.5, -5.0);
    glVertex3f(-5.0, -1.5, 5.0);

    // Sisi kanan (normal menghadap ke kanan)
    glNormal3f(1.0, 0.0, 0.0);
    glVertex3f(5.0, -1.0, 5.0);
    glVertex3f(5.0, -1.0, -5.0);
    glVertex3f(5.0, -1.5, -5.0);
    glVertex3f(5.0, -1.5, 5.0);

    glEnd();

    glPopMatrix();
}

void drawBackSky() {        //KAMILA
    glPushMatrix();

    glBegin(GL_QUADS);

    // Sisi belakang (menghadap kamera)
    glColor3f(0.96, 0.87, 0.70); // Warna krem (bawah)
    glNormal3f(0.0, 0.0, -1.0);
    glVertex3f(-5.0, -1.5, -6.0); // Kiri bawah
    glVertex3f(5.0, -1.5, -6.0);  // Kanan bawah
    glColor3f(0.53, 0.81, 0.98); // Warna biru muda (atas)
    glVertex3f(5.0, 3.0, -6.0);   // Kanan atas
    glVertex3f(-5.0, 3.0, -6.0);  // Kiri atas

    // Sisi depan
    glColor3f(0.96, 0.87, 0.70); // Warna krem (bawah)
    glNormal3f(0.0, 0.0, 1.0);
    glVertex3f(-5.0, -1.5, -5.0); // Kiri bawah
    glVertex3f(5.0, -1.5, -5.0);  // Kanan bawah
    glColor3f(0.53, 0.81, 0.98); // Warna biru muda (atas)
    glVertex3f(5.0, 3.0, -5.0);   // Kanan atas
    glVertex3f(-5.0, 3.0, -5.0);  // Kiri atas

    // Sisi kiri
    glColor3f(0.96, 0.87, 0.70); // Warna krem (bawah)
    glNormal3f(-1.0, 0.0, 0.0);
    glVertex3f(-5.0, -1.5, -5.0); // Depan bawah
```

```
    glVertex3f(-5.0, -1.5, -6.0); // Belakang bawah
    glColor3f(0.53, 0.81, 0.98); // Warna biru muda (atas)
    glVertex3f(-5.0, 3.0, -6.0);  // Belakang atas
    glVertex3f(-5.0, 3.0, -5.0);  // Depan atas

    // Sisi kanan
    glColor3f(0.96, 0.87, 0.70); // Warna krem (bawah)
    glNormal3f(1.0, 0.0, 0.0);
    glVertex3f(5.0, -1.5, -5.0); // Depan bawah
    glVertex3f(5.0, -1.5, -6.0); // Belakang bawah
    glColor3f(0.53, 0.81, 0.98); // Warna biru muda (atas)
    glVertex3f(5.0, 3.0, -6.0);  // Belakang atas
    glVertex3f(5.0, 3.0, -5.0);  // Depan atas

    // Sisi bawah
    glColor3f(0.96, 0.87, 0.70); // Warna krem
    glNormal3f(0.0, -1.0, 0.0);
    glVertex3f(-5.0, -1.5, -5.0); // Depan kiri
    glVertex3f(5.0, -1.5, -5.0);  // Depan kanan
    glVertex3f(5.0, -1.5, -6.0);  // Belakang kanan
    glVertex3f(-5.0, -1.5, -6.0); // Belakang kiri

    // Sisi atas
    glColor3f(0.53, 0.81, 0.98); // Warna biru muda
    glNormal3f(0.0, 1.0, 0.0);
    glVertex3f(-5.0, 3.0, -5.0);  // Depan kiri
    glVertex3f(5.0, 3.0, -5.0);   // Depan kanan
    glVertex3f(5.0, 3.0, -6.0);   // Belakang kanan
    glVertex3f(-5.0, 3.0, -6.0);  // Belakang kiri

    glEnd();

    glPopMatrix();
}

void drawRightSky() {            //IMAM
    glPushMatrix();

    glBegin(GL_QUADS);

    // Sisi kanan
    glColor3f(0.96, 0.87, 0.70); // Warna krem (bawah)
    glNormal3f(1.0, 0.0, 0.0);
    glVertex3f(5.0, -1.5, -5.0); // Bawah belakang
    glVertex3f(5.0, -1.5, 5.0);  // Bawah depan
    glColor3f(0.53, 0.81, 0.98); // Warna biru muda (atas)
    glVertex3f(5.0, 3.0, 5.0);   // Atas depan
    glVertex3f(5.0, 3.0, -5.0);  // Atas belakang

    // Sisi kiri
    glColor3f(0.96, 0.87, 0.70); // Warna krem (bawah)
    glNormal3f(-1.0, 0.0, 0.0);
    glVertex3f(6.0, -1.5, -5.0); // Bawah belakang
    glVertex3f(6.0, -1.5, 5.0);  // Bawah depan
    glColor3f(0.53, 0.81, 0.98); // Warna biru muda (atas)
    glVertex3f(6.0, 3.0, 5.0);   // Atas depan
    glVertex3f(6.0, 3.0, -5.0);  // Atas belakang
```

```
    // Sisi depan
    glColor3f(0.96, 0.87, 0.70); // Warna krem (bawah)
    glNormal3f(0.0, 0.0, 1.0);
    glVertex3f(5.0, -1.5, 5.0);  // Kanan bawah
    glVertex3f(6.0, -1.5, 5.0);  // Kiri bawah
    glColor3f(0.53, 0.81, 0.98); // Warna biru muda (atas)
    glVertex3f(6.0, 3.0, 5.0);   // Kiri atas
    glVertex3f(5.0, 3.0, 5.0);   // Kanan atas

    // Sisi belakang
    glColor3f(0.96, 0.87, 0.70); // Warna krem (bawah)
    glNormal3f(0.0, 0.0, -1.0);
    glVertex3f(5.0, -1.5, -5.0); // Kanan bawah
    glVertex3f(6.0, -1.5, -5.0); // Kiri bawah
    glColor3f(0.53, 0.81, 0.98); // Warna biru muda (atas)
    glVertex3f(6.0, 3.0, -5.0);  // Kiri atas
    glVertex3f(5.0, 3.0, -5.0);  // Kanan atas

    // Sisi bawah
    glColor3f(0.96, 0.87, 0.70); // Warna krem
    glNormal3f(0.0, -1.0, 0.0);
    glVertex3f(5.0, -1.5, -5.0); // Belakang kanan
    glVertex3f(6.0, -1.5, -5.0); // Belakang kiri
    glVertex3f(6.0, -1.5, 5.0);  // Depan kiri
    glVertex3f(5.0, -1.5, 5.0);  // Depan kanan

    // Sisi atas
    glColor3f(0.53, 0.81, 0.98); // Warna biru muda
    glNormal3f(0.0, 1.0, 0.0);
    glVertex3f(5.0, 3.0, -5.0);  // Belakang kanan
    glVertex3f(6.0, 3.0, -5.0);  // Belakang kiri
    glVertex3f(6.0, 3.0, 5.0);   // Depan kiri
    glVertex3f(5.0, 3.0, 5.0);   // Depan kanan

    glEnd();

    glPopMatrix();
}


void drawCarte() {                //IMAM
    if (!showCarte) return; // Tidak menggambar jika showAxis
false

    glPushMatrix();

    // Sumbu X (merah)
    glColor3f(1.0, 0.0, 0.0);
    glBegin(GL_LINES);
    glVertex3f(-50.0, 0.0, 0.0);
    glVertex3f(50.0, 0.0, 0.0);
    glEnd();

    // Sumbu Y (hijau)
    glColor3f(0.0, 1.0, 0.0);
    glBegin(GL_LINES);
```

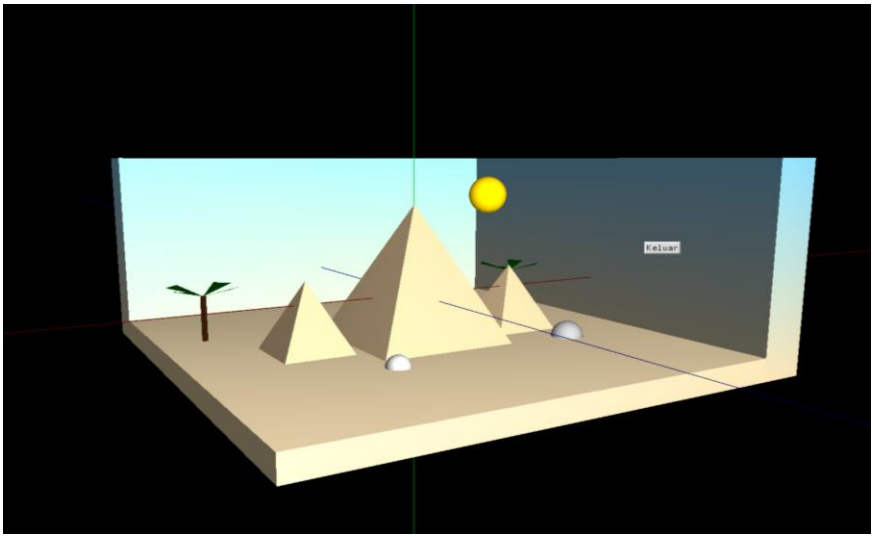
```
    glVertex3f(0.0, -50.0, 0.0);
    glVertex3f(0.0, 50.0, 0.0);
    glEnd();

    // Sumbu Z (biru)
    glColor3f(0.0, 0.0, 1.0);
    glBegin(GL_LINES);
    glVertex3f(0.0, 0.0, -50.0);
    glVertex3f(0.0, 0.0, 50.0);
    glEnd();

    glPopMatrix();
}

void display() {            //FADILAH
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glLoadIdentity();

    glTranslatef(0.0, 0.0, zoom); // Pindahkan kamera untuk zoom
    glRotatef(rotateX, 1.0, 0.0, 0.0); // Rotasi pada sumbu X
    glRotatef(rotateY, 0.0, 1.0, 0.0); // Rotasi pada sumbu Y

    drawCarte(); // Gambar garis kartesius (tergantung showAxis)

    drawLantai(); // Gambar lantai
    drawBackSky();
    drawRightSky();

    drawMatahari(4.0, 2.0, -3.0, 1.0);

    // Gambar tiga Pyramid
    drawPyramid(-2.5, -0.5, 0.0, 0.5); // Pyramid kiri lebih kecil
dan lebih bawah
    drawPyramid(0.0, 0.0, 0.0, 1.0);   // Pyramid tengah normal
    drawPyramid(2.5, -0.5, 0.0, 0.5);  // Pyramid kanan lebih
kecil dan lebih bawah

    // Gambar pohon kelapa
    drawPohon(-4.0, -1.0, -2.0);
    drawPohon(4.0, -1.0, -2.0);

    // Gambar batu-batu
    drawBatu(-1.5, -1.0, 2.0, 0.5);
    drawBatu(1.0, -1.0, -2.5, 0.3);
    drawBatu(3.0, -1.0, 1.5, 0.7);

    glutSwapBuffers();

}

void reshape(int w, int h) {          //KAMILA
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, (double)w / (double)h, 1.0, 100.0);
    glMatrixMode(GL_MODELVIEW);
```

```
}


void keyboard(unsigned char key, int x, int y) {       //KAMILA
    switch (key) {
        case 'w': // Rotasi ke atas
            rotateX -= 5.0;
            break;
        case 's': // Rotasi ke bawah
            rotateX += 5.0;
            break;
        case 'a': // Rotasi ke kiri
            rotateY -= 5.0;
            break;
        case 'd': // Rotasi ke kanan
            rotateY += 5.0;
            break;
        case 'i': // Zoom in
            zoom += 0.5;
            break;
        case 'o': // Zoom out
            zoom -= 0.5;
            break;
        case 'x': // Toggle garis kartesius
            showCarte = !showCarte;
            break;
    }
    glutPostRedisplay(); // Refresh tampilan
}

void MenuKeluar(int option) {          //KAMILA
    switch (option) {
        case 0: // Opsi keluar
            exit(0);
            break;
    }
}

// Fungsi untuk membuat menu klik kanan
void createMenu() {                    //KAMILA
    glutCreateMenu(MenuKeluar);
    glutAddMenuEntry("Keluar", 0); // Tambahkan entri menu
"Keluar"
    glutAttachMenu(GLUT_RIGHT_BUTTON); // Kaitkan menu dengan klik
kanan
}

void init() {              //IMAM

    glEnable(GL_DEPTH_TEST);           // Aktifkan depth test
    glEnable(GL_COLOR_MATERIAL);
    glEnable(GL_LIGHT0); // Sumber cahaya 0

    glClearColor(0.0, 0.0, 0.0, 1.0); // Warna latar belakang
hitam

    // Atur parameter sumber cahaya
```

```
      GLfloat lightPosition[] = {6.0, 3.0, -6.0, 1.0}; // Posisi
sumber cahaya
    GLfloat lightAmbient[] = {0.2, 0.2, 0.2, 1.0};  // Cahaya
sekitar
    GLfloat lightDiffuse[] = {1.0, 1.0, 1.0, 1.0};  // Cahaya
difus
    GLfloat lightSpecular[] = {1.0, 1.0, 1.0, 1.0}; // Cahaya
spesular

    glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);
    glLightfv(GL_LIGHT0, GL_AMBIENT, lightAmbient);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, lightDiffuse);
    glLightfv(GL_LIGHT0, GL_SPECULAR, lightSpecular);
}

int main(int argc, char** argv) {           //IMAM
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(800, 600);
    glutCreateWindow("Pyramid 3D");

    init();

    createMenu();

    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutKeyboardFunc(keyboard); // Fungsi untuk input keyboard

    glutMainLoop();
    return 0;
}
```

3.2 Output

Setelah menyusun source code sesuai dengan kebutuhan, hasil output dari program dapat dilihat pada gambar berikut. Output ini menunjukkan bagaimana program bekerja sesuai dengan perintah yang telah ditentukan dalam source code. Berikut adalah tampilan hasilnya



Gambar 3. 1 Output Source Code

3.3 Penjelasan

Output yang dihasilkan dari kode diatas adalah tampilan grafis 3D yang menampilkan objek seperti piramida coklat, matahari kuning, batu abu-abu, pohon dengan batang coklat dan daun hijau, serta lantai krem yang menutupi bagian bawah layar. Latar belakang biru muda memberikan kesan ruang yang luas. Jika variabel showCarte diaktifkan, garis koordinat 3D akan muncul dengan warna merah, hijau, dan biru. Sistem pencahayaan yang diaktifkan memberikan efek pencahayaan realistis pada objek-objek tersebut.

Pengguna dapat memutar objek menggunakan tombol keyboard (a, d, w, s) dan mengatur zoom (i, o) dan x untuk menampilkan atau menghilangkan garis kartecius. Interaksi berbasis keyboard memungkinkan pengguna mengontrol rotasi, zoom, dan menampilkan garis kartecius.selain itu juga ketika pengguna mengklik kanan pada jendela aplikasi, sebuah menu akan muncul dengan opsi "Keluar." Jika pengguna memilih opsi tersebut, program akan langsung menutup dengan menjalankan perintah exit(0).Berikut penjelasan output yang dihasilkan dan mengapa hasilnya seperti itu:

```
void drawPyramid(float x, float y, float z, float scale) {
      //FADILAH
    glPushMatrix();
    glTranslatef(x, y, z);
    glScalef(scale, scale, scale);

      glEnable(GL_LIGHTING);
      glEnable(GL_NORMALIZE);
```

```
glBegin(GL_TRIANGLES);
glColor3f(0.82, 0.71, 0.55);

// Sisi depan (normal menghadap ke depan)
glNormal3f(0.0, 0.5, 1.0); // Normal untuk sisi depan
glVertex3f(0.0, 2.0, 0.0);
glVertex3f(-1.5, -1.0, 1.5);
glVertex3f(1.5, -1.0, 1.5);

// Sisi kanan (normal menghadap ke kanan)
glNormal3f(1.0, 0.5, 0.0); // Normal untuk sisi kanan
glVertex3f(0.0, 2.0, 0.0);
glVertex3f(1.5, -1.0, 1.5);
glVertex3f(1.5, -1.0, -1.5);

// Sisi belakang (normal menghadap ke belakang)
glNormal3f(0.0, 0.5, -1.0); // Normal untuk sisi belakang
glVertex3f(0.0, 2.0, 0.0);
glVertex3f(1.5, -1.0, -1.5);
glVertex3f(-1.5, -1.0, -1.5);

// Sisi kiri (normal menghadap ke kiri)
glNormal3f(-1.0, 0.5, 0.0); // Normal untuk sisi kiri
glVertex3f(0.0, 2.0, 0.0);
glVertex3f(-1.5, -1.0, -1.5);
glVertex3f(-1.5, -1.0, 1.5);

glEnd();

// Dasar Pyramid (normal menghadap ke bawah)
glBegin(GL_QUADS);
glNormal3f(0.0, -1.0, 0.0); // Normal untuk dasar
glVertex3f(-1.5, -1.0, 1.5);
glVertex3f(1.5, -1.0, 1.5);
glVertex3f(1.5, -1.0, -1.5);
glVertex3f(-1.5, -1.0, -1.5);
glEnd();
```

```
      glDisable(GL_NORMALIZE);
    glPopMatrix();
}
```

Fungsi ini digunakan untuk menggambar *pyramid* 3D dengan sisi-sisi berbentuk segitiga dan dasar berbentuk persegi panjang. *Pyramid* ini menggunakan teknik pencahayaan untuk memberikan efek visual yang lebih nyata dengan mendefinisikan normal setiap sisi objek.

```
void drawMatahari(float x, float y, float z, float scale) {
      //FADILAH
    glPushMatrix();
    glTranslatef(x, y, z);
    glScalef(scale, scale, scale);
    glColor3f(1.0, 0.84, 0.0); // Warna kuning untuk matahari
    glutSolidSphere(0.5, 50, 50);
    glPopMatrix();
}
```

Fungsi ini digunakan untuk menggambar bola 3D yang mewakili matahari menggunakan fungsi glutSolidSphere. Bola tersebut diberi warna kuning (RGB: 1.0, 0.84, 0.0) dan diposisikan sesuai parameter x, y, dan z.

```
void drawBatu(float x, float y, float z, float scale) {
           //KAMILA
    glPushMatrix();
    glTranslatef(x, y, z);
    glScalef(scale, scale, scale);
    glColor3f(0.5, 0.5, 0.5); // Warna abu-abu gelap untuk batu
    glutSolidSphere(0.5, 20, 20);
    glPopMatrix();
}
```

Fungsi ini menggambar batu berbentuk bola dengan warna abu-abu. Batu ini juga digambar menggunakan glutSolidSphere, namun dengan parameter skala dan posisi yang berbeda.

```
void drawPohon(float x, float y, float z) {
//KAMILA
    glPushMatrix();
    glTranslatef(x, y, z);

    // Batang pohon (lebih kecil)
    glColor3f(0.55, 0.27, 0.07); // Warna coklat untuk batang
    glBegin(GL_QUADS);
    glVertex3f(-0.05, 0.0, -0.05);
    glVertex3f(0.05, 0.0, -0.05);
    glVertex3f(0.05, 1.0, -0.05); // Tinggi batang menjadi 1.0
    glVertex3f(-0.05, 1.0, -0.05);

    glVertex3f(-0.05, 0.0, 0.05);
    glVertex3f(0.05, 0.0, 0.05);
    glVertex3f(0.05, 1.0, 0.05);
    glVertex3f(-0.05, 1.0, 0.05);

    glVertex3f(-0.05, 0.0, -0.05);
    glVertex3f(-0.05, 0.0, 0.05);
    glVertex3f(-0.05, 1.0, 0.05);
    glVertex3f(-0.05, 1.0, -0.05);

    glVertex3f(0.05, 0.0, -0.05);
    glVertex3f(0.05, 0.0, 0.05);
    glVertex3f(0.05, 1.0, 0.05);
    glVertex3f(0.05, 1.0, -0.05);
    glEnd();

    // Daun kelapa (posisi dan dimensi disesuaikan)
    glColor3f(0.0, 0.5, 0.0); // Warna hijau untuk daun
    glBegin(GL_TRIANGLES);

    // Daun pertama (kiri depan)
    glVertex3f(0.0, 1.0, 0.0); // Dasar daun tepat di ujung batang
    glVertex3f(-0.7, 1.2, -0.3); // Ujung daun lebih panjang
    glVertex3f(-0.4, 1.2, -0.6);
```

```
    // Daun kedua (kanan depan)
    glVertex3f(0.0, 1.0, 0.0);
    glVertex3f(0.7, 1.2, -0.3);
    glVertex3f(0.4, 1.2, -0.6);

    // Daun ketiga (kiri belakang)
    glVertex3f(0.0, 1.0, 0.0);
    glVertex3f(-0.7, 1.2, 0.3);
    glVertex3f(-0.4, 1.2, 0.6);

    // Daun keempat (kanan belakang)
    glVertex3f(0.0, 1.0, 0.0);
    glVertex3f(0.7, 1.2, 0.3);
    glVertex3f(0.4, 1.2, 0.6);

    glEnd();

    glPopMatrix();
}
```

Fungsi ini menggambar pohon dengan batang berbentuk prisma dan daun berbentuk segitiga. Batang pohon digambar dengan glbegin(gl_quads) untuk membuat sisi persegi panjang, dan daun pohon digambar dengan glbegin(gl_triangles).

```
void drawLantai() {                                    //FADILAH
    glPushMatrix();

    glColor3f(0.96, 0.87, 0.70); // Warna krem untuk lantai

    // Gambarkan lantai dengan normal untuk setiap sisi
    glBegin(GL_QUADS);

    // Sisi atas (normal menghadap ke atas)
    glNormal3f(0.0, 1.0, 0.0);
    glVertex3f(-5.0, -1.0, 5.0);
    glVertex3f(5.0, -1.0, 5.0);
    glVertex3f(5.0, -1.0, -5.0);
    glVertex3f(-5.0, -1.0, -5.0);
```

```
// Sisi bawah (normal menghadap ke bawah)
glNormal3f(0.0, -1.0, 0.0);
glVertex3f(-5.0, -1.5, 5.0);
glVertex3f(5.0, -1.5, 5.0);
glVertex3f(5.0, -1.5, -5.0);
glVertex3f(-5.0, -1.5, -5.0);

// Sisi depan (normal menghadap ke depan)
glNormal3f(0.0, 0.0, 1.0);
glVertex3f(-5.0, -1.0, 5.0);
glVertex3f(5.0, -1.0, 5.0);
glVertex3f(5.0, -1.5, 5.0);
glVertex3f(-5.0, -1.5, 5.0);

// Sisi belakang (normal menghadap ke belakang)
glNormal3f(0.0, 0.0, -1.0);
glVertex3f(-5.0, -1.0, -5.0);
glVertex3f(5.0, -1.0, -5.0);
glVertex3f(5.0, -1.5, -5.0);
glVertex3f(-5.0, -1.5, -5.0);

// Sisi kiri (normal menghadap ke kiri)
glNormal3f(-1.0, 0.0, 0.0);
glVertex3f(-5.0, -1.0, 5.0);
glVertex3f(-5.0, -1.0, -5.0);
glVertex3f(-5.0, -1.5, -5.0);
glVertex3f(-5.0, -1.5, 5.0);

// Sisi kanan (normal menghadap ke kanan)
glNormal3f(1.0, 0.0, 0.0);
glVertex3f(5.0, -1.0, 5.0);
glVertex3f(5.0, -1.0, -5.0);
glVertex3f(5.0, -1.5, -5.0);
glVertex3f(5.0, -1.5, 5.0);

glEnd();
```

```
    glPopMatrix();
}
```

Fungsi ini menggambar lantai sebagai sebuah bidang datar (dengan sisi-sisi berbentuk kuadrat). Lantai ini juga memiliki normal untuk setiap sisi agar pencahayaan dapat diterapkan dengan benar.

```
void drawBackSky() {        //KAMILA
    glPushMatrix();

    glBegin(GL_QUADS);

    // Sisi belakang (menghadap kamera)
    glColor3f(0.96, 0.87, 0.70); // Warna krem (bawah)
    glNormal3f(0.0, 0.0, -1.0);
    glVertex3f(-5.0, -1.5, -6.0); // Kiri bawah
    glVertex3f(5.0, -1.5, -6.0);  // Kanan bawah
    glColor3f(0.53, 0.81, 0.98); // Warna biru muda (atas)
    glVertex3f(5.0, 3.0, -6.0);   // Kanan atas
    glVertex3f(-5.0, 3.0, -6.0);  // Kiri atas

    // Sisi depan
    glColor3f(0.96, 0.87, 0.70); // Warna krem (bawah)
    glNormal3f(0.0, 0.0, 1.0);
    glVertex3f(-5.0, -1.5, -5.0); // Kiri bawah
    glVertex3f(5.0, -1.5, -5.0);  // Kanan bawah
    glColor3f(0.53, 0.81, 0.98); // Warna biru muda (atas)
    glVertex3f(5.0, 3.0, -5.0);   // Kanan atas
    glVertex3f(-5.0, 3.0, -5.0);  // Kiri atas

    // Sisi kiri
    glColor3f(0.96, 0.87, 0.70); // Warna krem (bawah)
    glNormal3f(-1.0, 0.0, 0.0);
    glVertex3f(-5.0, -1.5, -5.0); // Depan bawah
    glVertex3f(-5.0, -1.5, -6.0); // Belakang bawah
    glColor3f(0.53, 0.81, 0.98); // Warna biru muda (atas)
```

```
    glVertex3f(-5.0, 3.0, -6.0);  // Belakang atas
    glVertex3f(-5.0, 3.0, -5.0);  // Depan atas

    // Sisi kanan
    glColor3f(0.96, 0.87, 0.70); // Warna krem (bawah)
    glNormal3f(1.0, 0.0, 0.0);
    glVertex3f(5.0, -1.5, -5.0); // Depan bawah
    glVertex3f(5.0, -1.5, -6.0); // Belakang bawah
    glColor3f(0.53, 0.81, 0.98); // Warna biru muda (atas)
    glVertex3f(5.0, 3.0, -6.0);  // Belakang atas
    glVertex3f(5.0, 3.0, -5.0);  // Depan atas

    // Sisi bawah
    glColor3f(0.96, 0.87, 0.70); // Warna krem
    glNormal3f(0.0, -1.0, 0.0);
    glVertex3f(-5.0, -1.5, -5.0); // Depan kiri
    glVertex3f(5.0, -1.5, -5.0);  // Depan kanan
    glVertex3f(5.0, -1.5, -6.0);  // Belakang kanan
    glVertex3f(-5.0, -1.5, -6.0); // Belakang kiri

    // Sisi atas
    glColor3f(0.53, 0.81, 0.98); // Warna biru muda
    glNormal3f(0.0, 1.0, 0.0);
    glVertex3f(-5.0, 3.0, -5.0);  // Depan kiri
    glVertex3f(5.0, 3.0, -5.0);   // Depan kanan
    glVertex3f(5.0, 3.0, -6.0);   // Belakang kanan
    glVertex3f(-5.0, 3.0, -6.0);  // Belakang kiri

    glEnd();

    glPopMatrix();
}
```

Fungsi drawBackSky() menggambarkan sebuah kubus yang berfungsi sebagai latar belakang langit, Fungsi ini juga menggunakan primitif OpenGL GL_QUADS untuk menggambar setiap sisi kubus dengan empat titik sudut (vertex). Selain itu, fungsi ini mengatur normal pada setiap sisi untuk membantu pencahayaan yang benar sesuai dengan orientasi permukaan. Kubus ini dibuat dengan gradasi warna dari krem di bagian bawah ke biru muda di

bagian atas, sehingga menciptakan ilusi langit.

```
void drawRightSky() {            //IMAM
    glPushMatrix();

    glBegin(GL_QUADS);

    // Sisi kanan
    glColor3f(0.96, 0.87, 0.70); // Warna krem (bawah)
    glNormal3f(1.0, 0.0, 0.0);
    glVertex3f(5.0, -1.5, -5.0); // Bawah belakang
    glVertex3f(5.0, -1.5, 5.0);  // Bawah depan
    glColor3f(0.53, 0.81, 0.98); // Warna biru muda (atas)
    glVertex3f(5.0, 3.0, 5.0);   // Atas depan
    glVertex3f(5.0, 3.0, -5.0);  // Atas belakang

    // Sisi kiri
    glColor3f(0.96, 0.87, 0.70); // Warna krem (bawah)
    glNormal3f(-1.0, 0.0, 0.0);
    glVertex3f(6.0, -1.5, -5.0); // Bawah belakang
    glVertex3f(6.0, -1.5, 5.0);  // Bawah depan
    glColor3f(0.53, 0.81, 0.98); // Warna biru muda (atas)
    glVertex3f(6.0, 3.0, 5.0);   // Atas depan
    glVertex3f(6.0, 3.0, -5.0);  // Atas belakang

    // Sisi depan
    glColor3f(0.96, 0.87, 0.70); // Warna krem (bawah)
    glNormal3f(0.0, 0.0, 1.0);
    glVertex3f(5.0, -1.5, 5.0);  // Kanan bawah
    glVertex3f(6.0, -1.5, 5.0);  // Kiri bawah
    glColor3f(0.53, 0.81, 0.98); // Warna biru muda (atas)
    glVertex3f(6.0, 3.0, 5.0);   // Kiri atas
    glVertex3f(5.0, 3.0, 5.0);   // Kanan atas

    // Sisi belakang
    glColor3f(0.96, 0.87, 0.70); // Warna krem (bawah)
    glNormal3f(0.0, 0.0, -1.0);
    glVertex3f(5.0, -1.5, -5.0); // Kanan bawah
    glVertex3f(6.0, -1.5, -5.0); // Kiri bawah
    glColor3f(0.53, 0.81, 0.98); // Warna biru muda (atas)
```

```
    glVertex3f(6.0, 3.0, -5.0);  // Kiri atas
    glVertex3f(5.0, 3.0, -5.0);  // Kanan atas

    // Sisi bawah
    glColor3f(0.96, 0.87, 0.70); // Warna krem
    glNormal3f(0.0, -1.0, 0.0);
    glVertex3f(5.0, -1.5, -5.0); // Belakang kanan
    glVertex3f(6.0, -1.5, -5.0); // Belakang kiri
    glVertex3f(6.0, -1.5, 5.0);  // Depan kiri
    glVertex3f(5.0, -1.5, 5.0);  // Depan kanan

    // Sisi atas
    glColor3f(0.53, 0.81, 0.98); // Warna biru muda
    glNormal3f(0.0, 1.0, 0.0);
    glVertex3f(5.0, 3.0, -5.0);  // Belakang kanan
    glVertex3f(6.0, 3.0, -5.0);  // Belakang kiri
    glVertex3f(6.0, 3.0, 5.0);   // Depan kiri
    glVertex3f(5.0, 3.0, 5.0);   // Depan kanan

    glEnd();

    glPopMatrix();
```

Fungsi drawRightSky() menggambarkan sebuah kubus yang berfungsi sebagai latar belakang langit di sebelah kanan , Fungsi ini juga menggunakan primitif OpenGL GL_QUADS untuk menggambar setiap sisi kubus dengan empat titik sudut (vertex). Selain itu, fungsi ini mengatur normal pada setiap sisi untuk membantu pencahayaan yang benar sesuai dengan orientasi permukaan. Kubus ini dibuat dengan gradasi warna dari krem di bagian bawah ke biru muda di bagian atas, sehingga menciptakan ilusi langit.

```
void drawCarte() {                //IMAM
    if (!showCarte) return; // Tidak menggambar jika showAxis false

    glPushMatrix();

    // Sumbu X (merah)
    glColor3f(1.0, 0.0, 0.0);
    glBegin(GL_LINES);
```

```
    glVertex3f(-50.0, 0.0, 0.0);
    glVertex3f(50.0, 0.0, 0.0);
    glEnd();

    // Sumbu Y (hijau)
    glColor3f(0.0, 1.0, 0.0);
    glBegin(GL_LINES);
    glVertex3f(0.0, -50.0, 0.0);
    glVertex3f(0.0, 50.0, 0.0);
    glEnd();

    // Sumbu Z (biru)
    glColor3f(0.0, 0.0, 1.0);
    glBegin(GL_LINES);
    glVertex3f(0.0, 0.0, -50.0);
    glVertex3f(0.0, 0.0, 50.0);
    glEnd();

    glPopMatrix();
}
```

Fungsi drawCarte() digunakan untuk menggambar tiga sumbu koordinat (X, Y, dan Z) dalam ruang 3D. Fungsi ini hanya dijalankan jika variabel showCarte bernilai true, sehingga dapat dikontrol sesuai kebutuhan. Sumbu X digambar dalam warna merah dari titik (-50, 0, 0) hingga (50, 0, 0), sumbu Y dalam warna hijau dari (0, -50, 0) hingga (0, 50, 0), dan sumbu Z dalam warna biru dari (0, 0, -50) hingga (0, 0, 50). Semua sumbu ini digambar menggunakan primitif OpenGL GL_LINES untuk menciptakan garis lurus antara dua titik.

```
void display() {            //FADILAH
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glLoadIdentity();

    glTranslatef(0.0, 0.0, zoom); // Pindahkan kamera untuk zoom
    glRotatef(rotateX, 1.0, 0.0, 0.0); // Rotasi pada sumbu X
    glRotatef(rotateY, 0.0, 1.0, 0.0); // Rotasi pada sumbu Y

    drawCarte(); // Gambar garis kartesius (tergantung showAxis)
```

```
    drawLantai(); // Gambar lantai
    drawBackSky();
    drawRightSky();

    drawMatahari(4.0, 2.0, -3.0, 1.0);

    // Gambar tiga piramida
    drawPyramid(-2.5, -0.5, 0.0, 0.5); // Piramida kiri lebih kecil
dan lebih bawah
    drawPyramid(0.0, 0.0, 0.0, 1.0);   // Piramida tengah normal
    drawPyramid(2.5, -0.5, 0.0, 0.5);  // Piramida kanan lebih
kecil dan lebih bawah

    // Gambar pohon kelapa
    drawPohon(-4.0, -1.0, -2.0);
    drawPohon(4.0, -1.0, -2.0);

    // Gambar batu-batu
    drawBatu(-1.5, -1.0, 2.0, 0.5);
    drawBatu(1.0, -1.0, -2.5, 0.3);
    drawBatu(3.0, -1.0, 1.5, 0.7);

    glutSwapBuffers();

}
```

Fungsi display() digunakan untuk menggambar seluruh tampilan adegan di layar. Fungsi ini memulai dengan membersihkan layar dan menyiapkan ulang posisi kamera. Kamera dipindahkan lebih dekat atau jauh menggunakan glTranslatef agar terlihat lebih besar atau kecil (zoom) dan diputar sesuai sudut yang ditentukan pada sumbu X dan Y untuk mengatur sudut pandang.

Setelah itu, fungsi mulai menggambar berbagai elemen dalam adegan. Pertama, sumbu koordinat (kartesius) digambar menggunakan drawCarte() jika diaktifkan. Kemudian, lantai, langit belakang, dan langit kanan digambar dengan memanggil drawLantai(), drawBackSky(), dan drawRightSky(). Fungsi drawMatahari() digunakan untuk menggambar matahari di tempat tertentu.

Selanjutnya, tiga piramida digambar di posisi yang berbeda: piramida kiri dan kanan

lebih kecil, sementara piramida tengah lebih besar. Pohon kelapa juga digambar di sisi kiri dan kanan, sedangkan beberapa batu ditempatkan di berbagai lokasi untuk menambahkan detail pada pemandangan.

Terakhir, fungsi menggunakan glutSwapBuffers() untuk menampilkan hasil gambar ke layar dengan halus, sehingga adegan terlihat tanpa kedipan. Fungsi ini bertanggung jawab untuk memastikan semua bagian tampilan digambar dengan benar di tempat yang sesuai.

```
void reshape(int w, int h) {           //KAMILA
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, (double)w / (double)h, 1.0, 100.0);
    glMatrixMode(GL_MODELVIEW);
}
```

Fungsi reshape() digunakan untuk menyesuaikan tampilan (viewport) ketika ukuran jendela aplikasi berubah. Fungsi ini mengatur viewport baru sesuai dengan lebar (w) dan tinggi (h) jendela. Kemudian, fungsi ini mengubah mode matriks ke proyeksi dan menetapkan perspektif untuk tampilan 3D dengan sudut pandang 45 derajat dan rasio aspek yang sesuai dengan ukuran jendela. Setelah itu, mode matriks dikembalikan ke GL_MODELVIEW untuk menggambar objek 3D.

```
void keyboard(unsigned char key, int x, int y) {       //KAMILA
    switch (key) {
        case 'w': // Rotasi ke atas
            rotateX -= 5.0;
            break;
        case 's': // Rotasi ke bawah
            rotateX += 5.0;
            break;
        case 'a': // Rotasi ke kiri
            rotateY -= 5.0;
            break;
        case 'd': // Rotasi ke kanan
            rotateY += 5.0;
            break;
        case 'i': // Zoom in
```

```
            zoom += 0.5;
            break;
        case 'o': // Zoom out
            zoom -= 0.5;
            break;
        case 'x': // Toggle garis kartesius
            showCarte = !showCarte;
            break;
    }
    glutPostRedisplay(); // Refresh tampilan
}
```

Fungsi keyboard() menangani input dari keyboard untuk melakukan interaksi pada tampilan 3D. Fungsi ini merespons beberapa tombol untuk mengubah tampilan objek. Berikut adalah fungsinya:

- 'w': Memutar tampilan ke atas (mengurangi rotasi pada sumbu X).
- 's': Memutar tampilan ke bawah (menambah rotasi pada sumbu X).
- 'a': Memutar tampilan ke kiri (mengurangi rotasi pada sumbu Y).
- 'd': Memutar tampilan ke kanan (menambah rotasi pada sumbu Y).
- 'i': Melakukan zoom in (menambah nilai zoom).
- 'o': Melakukan zoom out (mengurangi nilai zoom).
- 'x': Menyembunyikan atau menampilkan garis kartesius (menukar status showCarte).

Setelah input diterima, tampilan akan di-refresh menggunakan glutPostRedisplay().

```
void MenuKeluar(int option) {          //KAMILA
    switch (option) {
        case 0: // Opsi keluar
            exit(0);
            break;
    }
}

// Fungsi untuk membuat menu klik kanan
void createMenu() {                    //KAMILA
    glutCreateMenu(MenuKeluar);
    glutAddMenuEntry("Keluar", 0); // Tambahkan entri menu "Keluar"
    glutAttachMenu(GLUT_RIGHT_BUTTON); // Kaitkan menu dengan klik
kanan
```

```
}
```

Fungsi MenuKeluar() digunakan untuk menangani pilihan menu yang dipilih melalui klik kanan. Dalam hal ini, ketika pilihan "Keluar" dipilih, program akan berhenti dengan perintah exit(0).

Fungsi createMenu() digunakan untuk membuat menu klik kanan. Berikut adalah langkah-langkah yang dilakukan dalam fungsi ini:

- glutCreateMenu(MenuKeluar): Membuat menu dan mengaitkannya dengan fungsi MenuKeluar untuk menangani pilihan menu.
- glutAddMenuEntry("Keluar", 0): Menambahkan entri menu dengan label "Keluar" yang terkait dengan nilai opsi 0.
- glutAttachMenu(GLUT_RIGHT_BUTTON): Mengaitkan menu dengan tombol kanan mouse, sehingga menu akan muncul saat tombol kanan diklik.

Dengan cara ini, menu "Keluar" akan muncul saat pengguna mengklik tombol kanan mouse, dan jika dipilih, program akan keluar.

```
void init() {               //IMAM

    glEnable(GL_DEPTH_TEST);           // Aktifkan depth test
    glEnable(GL_COLOR_MATERIAL);
    glEnable(GL_LIGHT0); // Sumber cahaya 0

    glClearColor(0.0, 0.0, 0.0, 1.0); // Warna latar belakang hitam

    // Atur parameter sumber cahaya
GLfloat lightPosition[] = {6.0, 3.0, -6.0, 1.0}; // Posisi sumber
cahaya
    GLfloat lightAmbient[] = {0.2, 0.2, 0.2, 1.0};  // Cahaya
sekitar
    GLfloat lightDiffuse[] = {1.0, 1.0, 1.0, 1.0};  // Cahaya difus
    GLfloat lightSpecular[] = {1.0, 1.0, 1.0, 1.0}; // Cahaya
```

```
spesular

    glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);
    glLightfv(GL_LIGHT0, GL_AMBIENT, lightAmbient);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, lightDiffuse);
    glLightfv(GL_LIGHT0, GL_SPECULAR, lightSpecular);
}
```

Fungsi init() bertujuan untuk mengonfigurasi pengaturan dasar dalam aplikasi grafis OpenGL. Pertama, fungsi ini mengaktifkan pengujian kedalaman dengan glEnable(GL_DEPTH_TEST), yang memastikan objek lebih dekat ke kamera dapat menutupi objek yang lebih jauh. Selanjutnya, fungsi ini mengaktifkan pengaturan material dan sumber cahaya melalui glEnable(GL_COLOR_MATERIAL) dan glEnable(GL_LIGHT0). Latar belakang diubah menjadi hitam dengan glClearColor(0.0, 0.0, 0.0, 1.0). Untuk pencahayaan, posisi sumber cahaya pertama diatur pada koordinat (6.0, 3.0, -6.0), di mana nilai terakhir 1.0 menunjukkan bahwa cahaya ini berupa cahaya titik. Selain itu, pencahayaan ambient, difus, dan spesular juga diatur untuk memberikan pencahayaan dasar, pencahayaan utama, serta sorotan pada permukaan halus. Dengan pengaturan ini, aplikasi siap untuk merender objek 3D dengan pencahayaan yang realistis.

```
int main(int argc, char** argv) {           //IMAM
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(800, 600);
    glutCreateWindow("Piramida 3D");

    init();

    createMenu();

    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutKeyboardFunc(keyboard); // Fungsi untuk input keyboard

    glutMainLoop();
    return 0;
}
```

Fungsi main() ini menginisialisasi dan menjalankan aplikasi grafis menggunakan GLUT. Pertama, GLUT diinisialisasi dengan glutInit() untuk menyiapkan argumen program. Selanjutnya, tampilan jendela diatur dengan glutInitDisplayMode untuk menentukan mode tampilan menggunakan double buffering, warna RGB, dan depth buffering. Ukuran jendela ditetapkan sebesar 800x600 piksel dan jendela dengan judul "Piramida 3D" dibuat. Fungsi init() kemudian dipanggil untuk mengonfigurasi pengaturan dasar seperti pencahayaan dan pengaturan kedalaman.

Selanjutnya, menu dengan opsi keluar dibuat menggunakan createMenu(). Fungsi untuk merender tampilan, mengatur perubahan ukuran jendela, dan menangani input keyboard dihubungkan melalui glutDisplayFunc, glutReshapeFunc, dan glutKeyboardFunc. Akhirnya, glutMainLoop() dijalankan untuk memulai loop utama aplikasi yang memungkinkan tampilan grafis dan interaksi dengan pengguna secara dinamis.

BAB IV

4.1. Kesimpulan

Secara keseluruhan, praktikum ini berhasil memenuhi tujuan utama yaitu memahami penggunaan OpenGL sebagai salah satu library grafis yang mendukung pengembangan visualisasi 2D dan 3D. Melalui serangkaian langkah praktikum, peserta berhasil mempelajari berbagai aspek teknis OpenGL, mulai dari konfigurasi perangkat lunak, implementasi fungsi-fungsi dasar, hingga penerapan transformasi objek. Praktikum ini tidak hanya memberikan wawasan teoritis tetapi juga memperkuat keterampilan teknis dalam membuat dan mengelola aplikasi grafis yang interaktif dan estetik.

Pembuatan objek 3D seperti piramida, matahari, batu, dan pohon menjadi contoh nyata bagaimana OpenGL dapat digunakan untuk menciptakan lingkungan visual yang realistis. Dengan memanfaatkan fungsi-fungsi seperti glTranslatef, glRotatef, dan glScalef, mampu memanipulasi posisi, rotasi, dan skala objek sesuai kebutuhan. Penerapan pencahayaan menggunakan parameter seperti ambient, diffuse, dan specular juga menambah dimensi realisme pada tampilan grafis.

Interaksi yang dirancang, seperti rotasi objek, zoom, dan pengaturan tampilan garis koordinat melalui input keyboard, memberikan fleksibilitas dan pengalaman yang lebih baik. Selain itu, adanya fitur menu berbasis klik kanan (GUI) menambah fungsionalitas sederhana namun efektif dalam pengelolaan aplikasi. Lingkungan visual yang diciptakan, termasuk penggunaan gradasi warna pada latar belakang dan pencahayaan pada objek.

DAFTAR PUSTAKA

[1] Muhammad Adnani And Achmad Zakki Falani, "Implementasi Open Gl Untuk Pembuatan Objek 3d," *Journal Zetroem*, Vol. 3, No. 1, Pp. 1–6, 2021, Doi: 10.36526/Ztr.V3i1.1249.

[2] M. B. Priyantono And A. A. Rachmawan, "Implementasi Sistem Simulasi Penampilan Tata Surya Berbasis 3D Menggunakan Opengl," *Jurnal Teknologi Informasi*, Vol. 4, No. 1, Pp. 91–95, 2020, Doi: 10.36294/Jurti.V4i1.1231.

[3] R. Iskandar, M. Melaniawati, And R. Candra, "Pembuatan Garis Besar Font (Outline Fonts) Menggunakan Program Opengl 32," *Semnasteknomedia Online*, Vol. 1, No. 1, Pp. 4–25, 2013.

[4] M. Y. Arianti, N. Fitriani, D. Khairani, And S. T. Adinda, "Rapor, Nilai, Dev C++ Analisis Nilai Akhir Rapor Dengan Program C++ Smk Islamiyah Sei Kamah Ii 2021," *Pkm-P*, Vol. 5, No. 2, Pp. 164–167, 2021.

[5] R. Aryadi, Suyanto, And Widodo, "Aplikasi Testing Interface Video Graphics Array Card Menggunakan Vb.Net," *Jurnal Sibernetika*, Vol. 5, No. 2, Pp. 209–215, 2020.

[6] A. Basuki And N. Ramadijanti, "Pengantar Grafika Komputer," *Laboratorium Computer Vision–(PENS-ITS)*.