LAPORAN PRATIKUM GRAFIK KOMPUTER

Diajukan untuk memenuhi Tugas mata kuliah Pratikum Grafik Komputer

PEMBUATAN OBJEK 3D ALAT-ALAT DORAEMON MENGGUNAKAN OPENGL

Dosen Pengampu : Sri Rahayu, M.Kom

Instruktur Pratikum : Arul Budi Kalimat, S.Kom

Disusun oleh

Kelompok 1 :

Muhammad Fathul Barry

2306122

Wilyandi Fajri

2306070

Sautan Ali Arrozak

2306102

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN ILMU KOMPUTER

INSTITUT TEKNOLOGI GARUT

2025

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 Pembuatan Objek 3D Menggunakan OpenGL.

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, 13 Januari 2025

Kelompok 4

DAFTAR ISI

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi grafika komputer telah mengalami kemajuan pesat dalam beberapa tahun terakhir, khususnya dalam pembuatan objek 3D. Grafika komputer 3D merupakan representasi data geometrik 3D yang dihasilkan melalui pemrosesan dan pemberian efek cahaya terhadap grafika komputer 2D. Teknologi ini memungkinkan visualisasi objek dalam bentuk tiga dimensi yang lebih realistis dan interaktif dengan menggunakan library grafis seperti OpenGL[1].

OpenGL (Open Graphics Library) telah menjadi standar API yang dapat digunakan untuk membuat aplikasi berbasis grafik, baik dua dimensi maupun tiga dimensi. Dalam implementasinya, OpenGL membutuhkan suatu konsep interfacing untuk proteksi objek, yang umumnya dilakukan melalui window-based OpenGL dengan bantuan GLUT (OpenGL Utility Toolkit). GLUT sendiri memiliki keunggulan karena bersifat portable, mudah digunakan, dan memiliki fungsi callback untuk interaksi dengan pengguna[2], [3].

Dalam konteks pendidikan dan pengembangan aplikasi, pemahaman tentang grafika komputer 3D menjadi semakin penting karena ketersediaan tools dan library yang memudahkan proses pembuatan objek 3D. Hal ini didukung oleh perkembangan teknologi rendering yang memungkinkan visualisasi data dan simulasi yang lebih kompleks, seperti yang ditunjukkan dalam penelitian terkini tentang performa rendering 3D berbasis web menggunakan WebGL dan GLSL[3].

Pemodelan objek 3D dalam bentuk geometris dimaksudkan agar gambar dapat dimanipulasi tanpa kehilangan akurasi karena perhitungan dilakukan secara numeris. Dengan menggunakan OpenGL, pengembang dapat membuat berbagai objek 3D mulai dari bentuk primitif hingga model kompleks dengan menerapkan transformasi, pencahayaan, dan tekstur yang dapat diimplementasikan dalam berbagai aplikasi modern[1], [3].

1.2 Rumusan Masalah

1. Bagaimana cara mengimplementasikan objek 3D alat-alat Doraemon menggunakan OpenGL?
2. Bagaimana cara menerapkan transformasi geometri (translasi, rotasi, dan skala) pada objek 3D alat Doraemon?

3. Bagaimana cara mengatur pencahayaan dan material pada objek 3D untuk menghasilkan visualisasi yang realistis?
4. Bagaimana cara mengimplementasikan tekstur pada objek 3D alat Doraemon?

1.3 Tujuan

1. Mengimplementasikan pembuatan objek 3D alat-alat Doraemon dengan menggunakan library OpenGL
2. Menerapkan transformasi geometri untuk memanipulasi objek 3D alat Doraemon secara interaktif
3. Mengatur pencahayaan dan material objek 3D untuk menghasilkan visualisasi yang realistis
4. Mengimplementasikan tekstur pada objek 3D alat Doraemon untuk meningkatkan detail visual

BAB II
TINJAUAN PUSTAKA

2.1 OpenGL

OpenGL (Open Graphics Library) adalah sebuah library dengan berbagai fungsi yang digunakan untuk menggambar objek 2 dimensi maupun 3 dimensi. Library ini berperan dalam mendefinisikan cross-bahasa serta cross-platform API untuk menciptakan aplikasi grafis komputer[2].

OpenGL dikembangkan pertama kali oleh Silicon Graphics Inc pada tahun 1992 dan digunakan dalam berbagai aplikasi seperti CAD, virtual reality, simulasi penerbangan, visualisasi informasi, dan industri game46. Sejak 2006, OpenGL dikelola oleh konsorsium teknologi non-profit Khronos Group[2].

2.2 Konfigurasi OpenGL pada Dev C++ atau VSCode

Persiapan File :

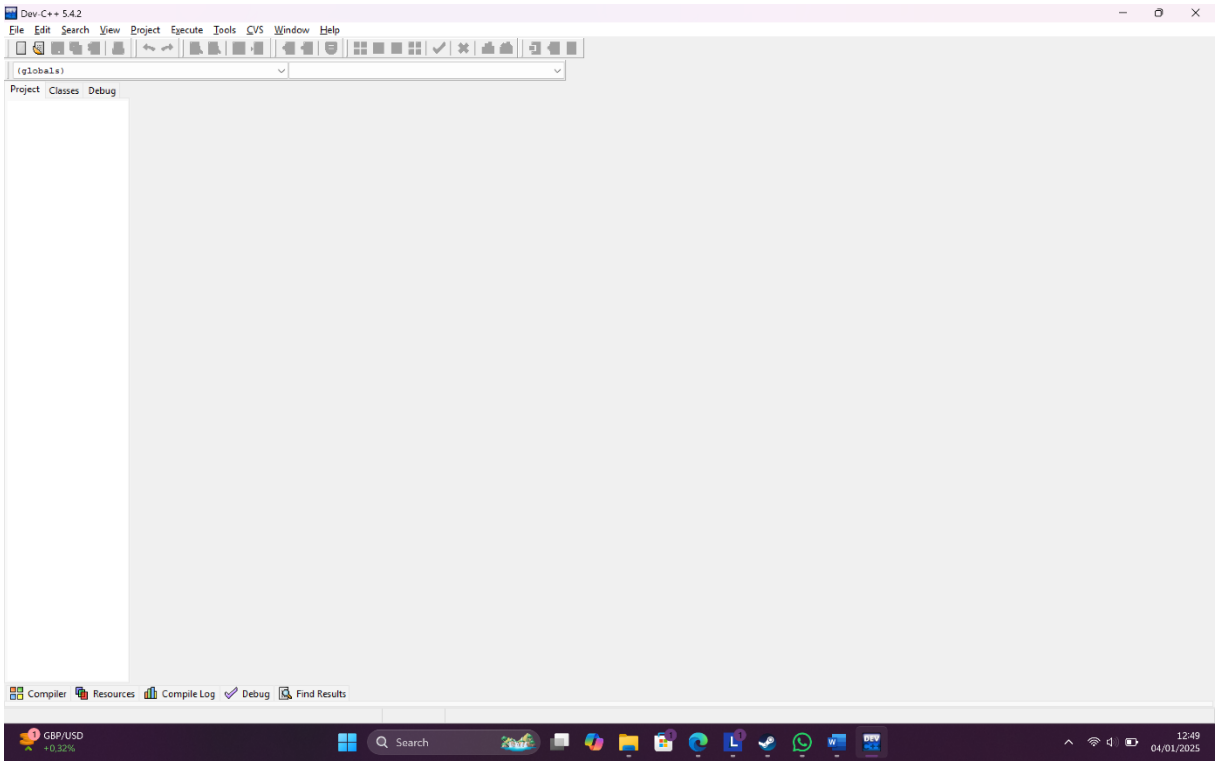
1. Download dan install Dev C++
2. Download freeglut untuk OpenGL
3. Ekstrak file freeglut yang telah didownload

Konfigurasi File :

1. Copy file dari folder bin\x64 ke C:\windows\system32
2. Copy file dari folder include\GL ke C:\Program Files (x86)\DevCpp\MinGW64\x86_64-w64-mingw32\include\GL
3. Copy file dari folder lib\x64 ke C:\Program Files (x86)\DevCpp\MinGW64\x86_64-w64-mingw32\lib

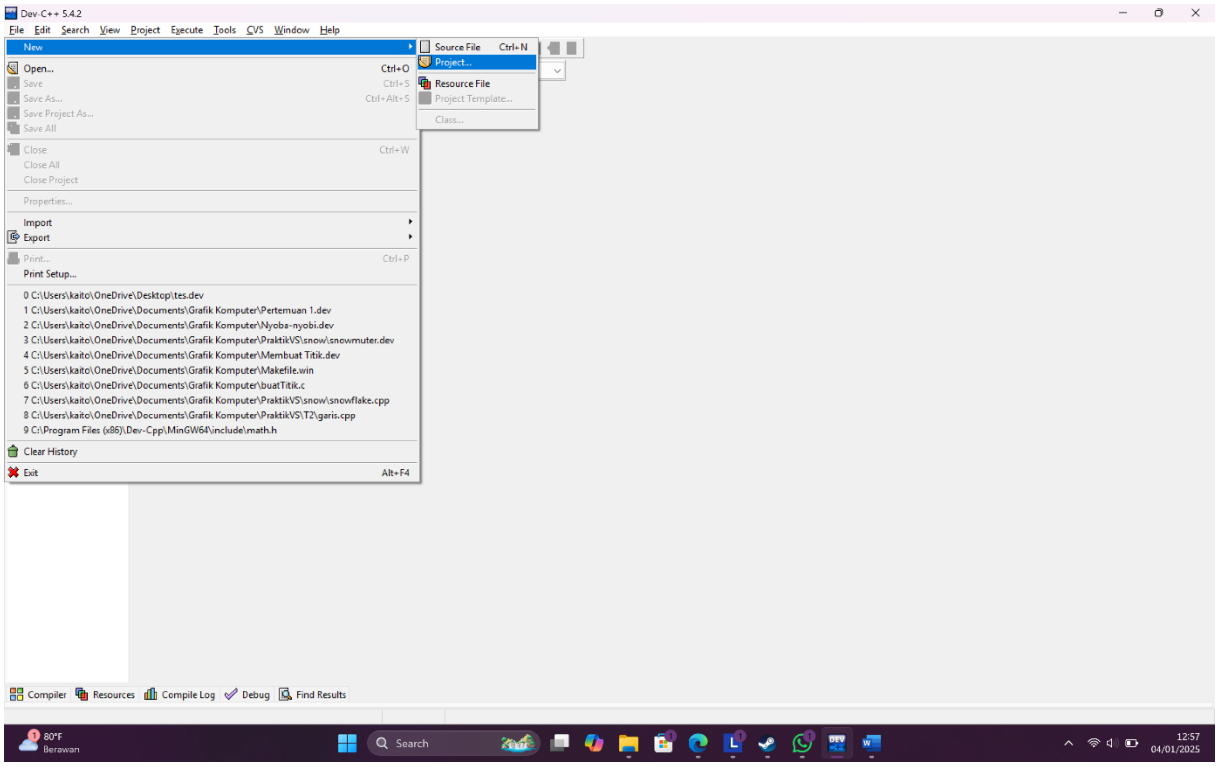
Berikut langkah-langkah untuk mengkonfigurasi OpenGL pada Dev C++ :

1. Buka Dev C++



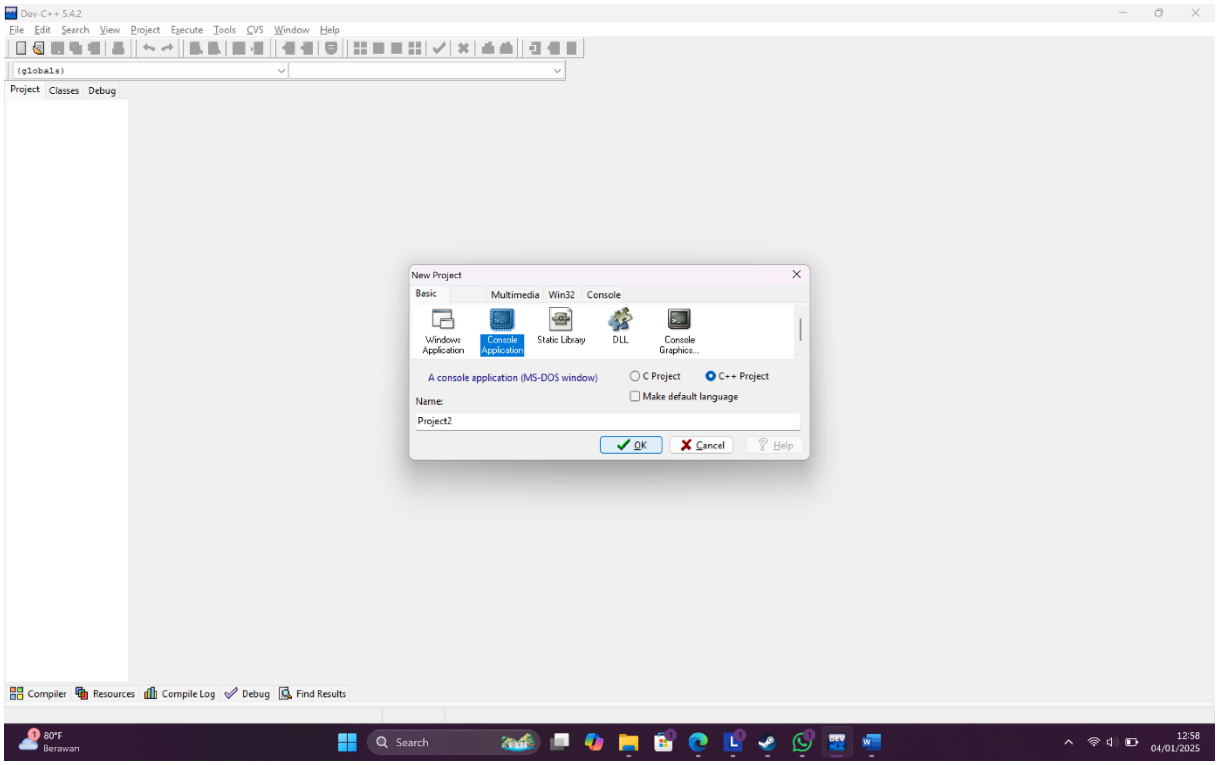
Gambar 2. 1 Menu awal Dev C++

2. Pilih File > New > Project



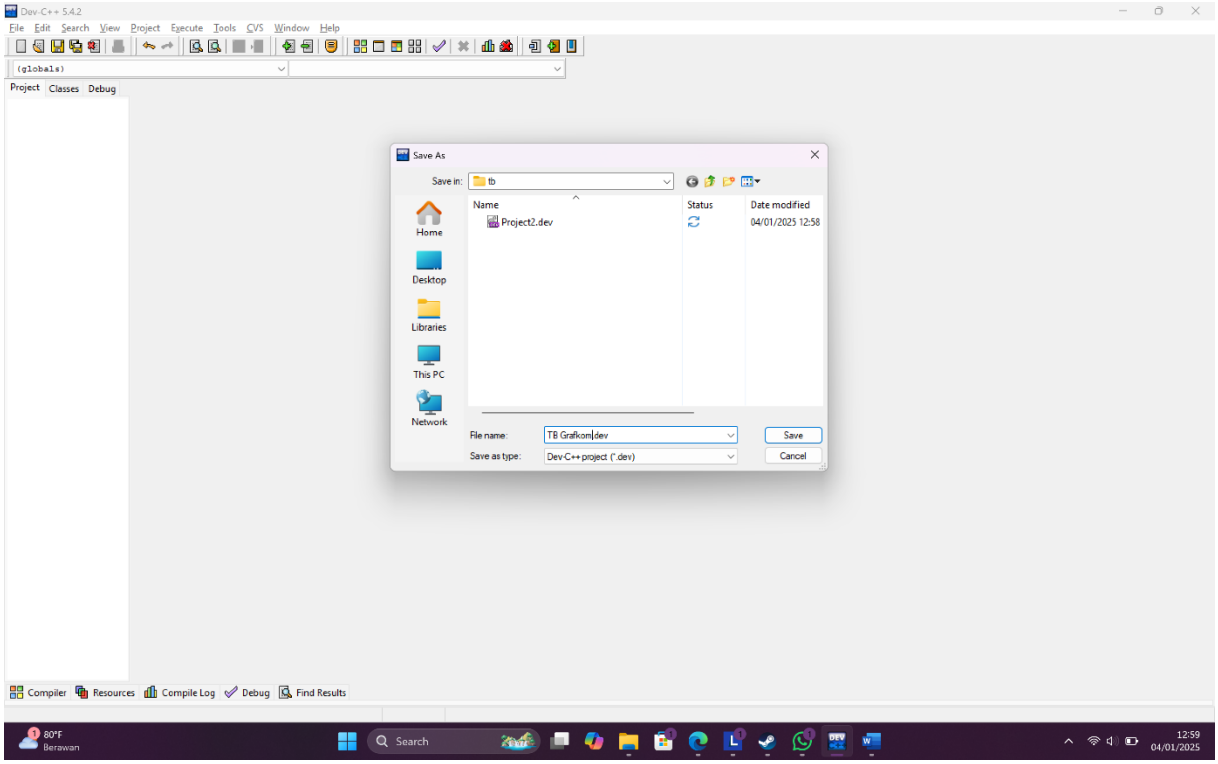
Gambar 2. 2 Menu project

3. Pilih Console Application > C++ Project



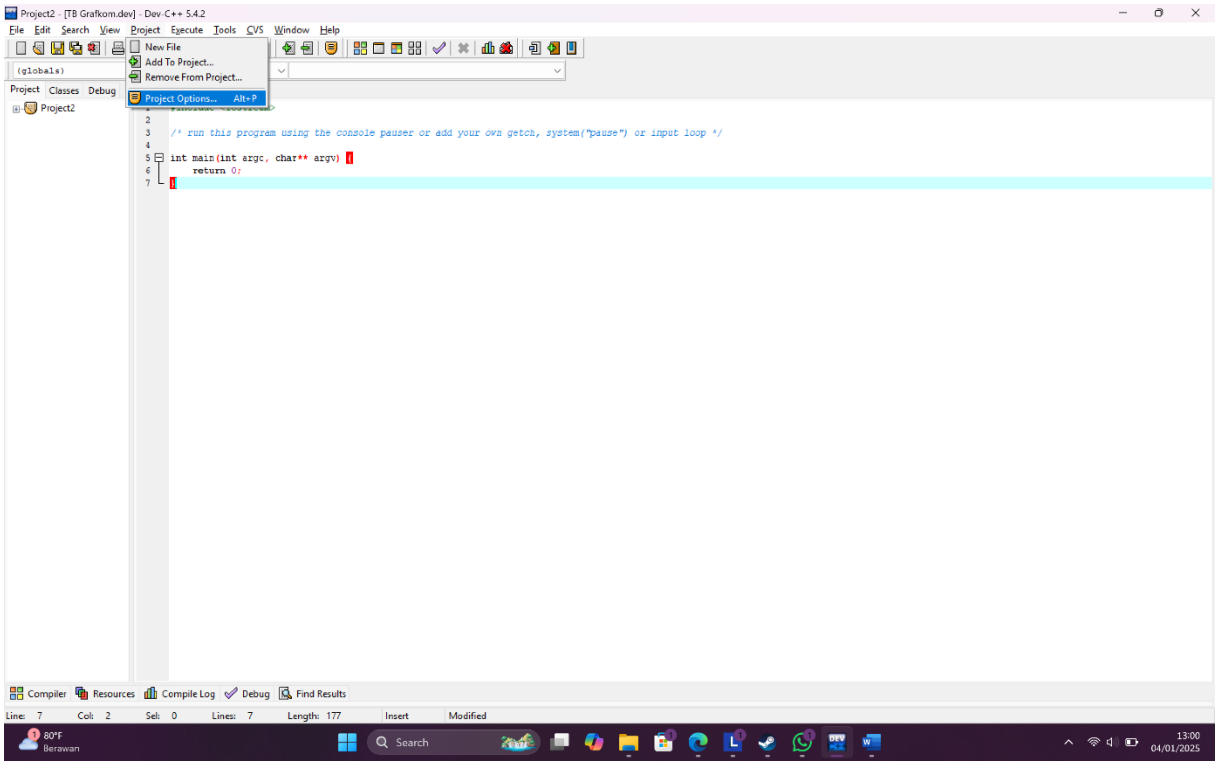
Gambar 2. 3 Console Application

4. Simpan project dengan nama yang diinginkan



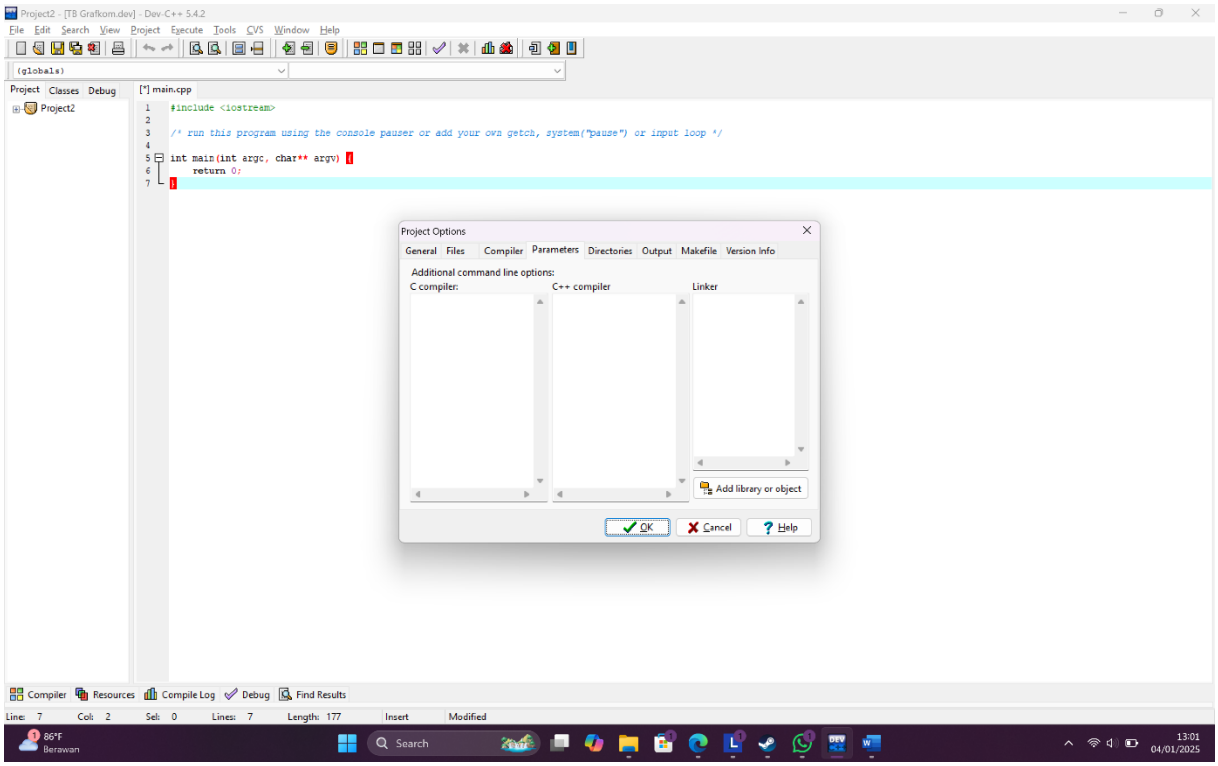
Gambar 2. 4 Nama project

5. Klik kanan pada nama project, pilih Project Options



Gambar 2. 5 Project options

6. Pilih tab Parameters



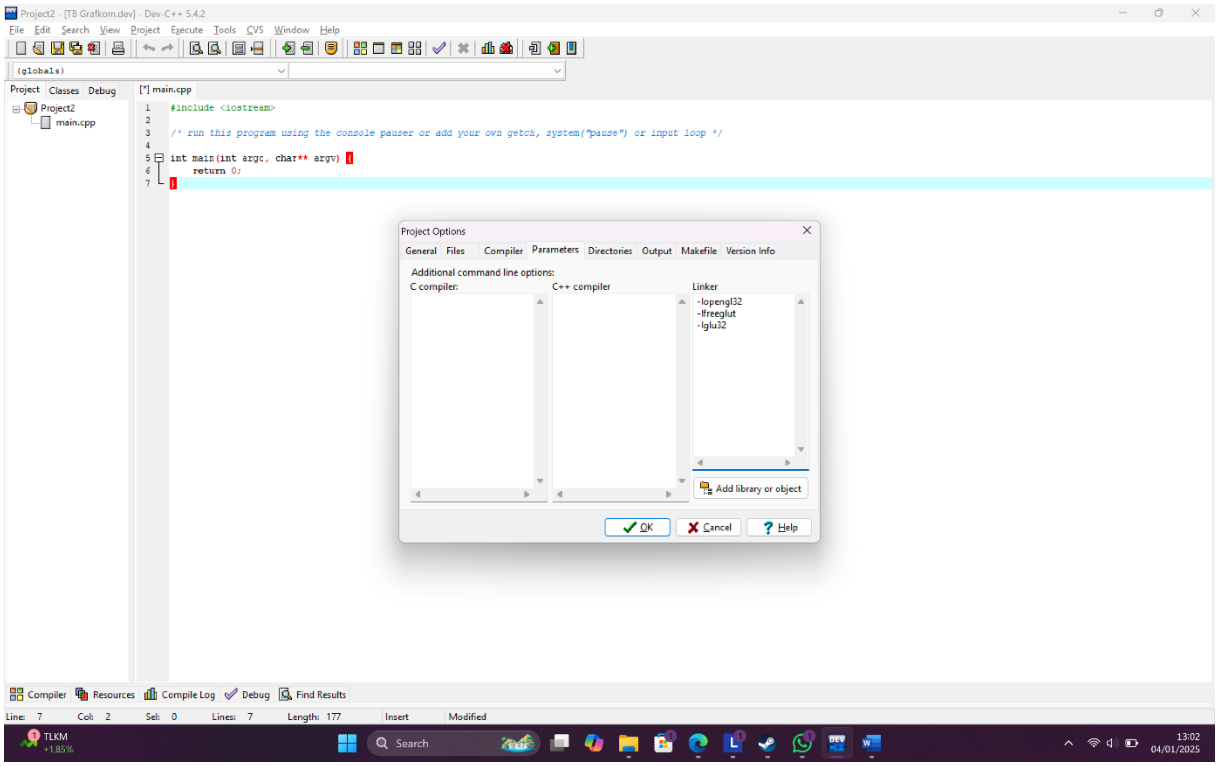
Gambar 2. 6 Parameter

7. Pada bagian Linker, tambahkan:

   -lopengl32

-lfreeglut

-lglu32



Gambar 2. 7 Isi Linker

2.3 Cara Kerja OpenGL

OpenGL merupakan sistem grafis yang bekerja melalui serangkaian tahapan pemrosesan yang saling terhubung. Proses dimulai dengan Input Data, di mana OpenGL menerima berbagai data geometri seperti vertex, koordinat, dan warna, serta parameter material dan tekstur yang akan digunakan dalam rendering. Data ini kemudian memasuki tahap Vertex Processing yang melakukan transformasi geometri pada setiap vertex, termasuk perhitungan pencahayaan dan transformasi proyeksi untuk menentukan posisi akhir objek dalam ruang 3D[2], [3].

Setelah vertex diproses, tahap Primitive Assembly menggabungkan vertex-vertex tersebut menjadi bentuk geometri dasar seperti titik, garis, atau poligon yang akan membentuk objek 3D. Hasil dari assembly ini kemudian memasuki proses Rasterization, di mana primitive geometri dikonversi menjadi fragmen-fragmen yang merepresentasikan pixel pada layar. Dalam tahap ini, sistem menentukan pixel mana yang akan terpengaruh oleh objek dan menginterpolasi berbagai atribut vertex seperti warna dan tekstur[2], [3].

Fragmen yang dihasilkan selanjutnya melalui Fragment Processing, di mana berbagai operasi seperti penerapan tekstur, pengujian kedalaman, dan efek pencahayaan diterapkan pada level fragmen. Tahap terakhir adalah Frame Buffer Operations, di mana hasil pemrosesan final disimpan dalam frame buffer dan kemudian ditampilkan pada layar. Seluruh proses ini berjalan secara otomatis dan teroptimasi pada GPU modern, memungkinkan rendering grafis yang efisien dan berkualitas tinggi[2], [3].

2.4 PEMBUATAN OBJEK 3D ALAT-ALAT DORAEMON MENGGUNAKAN OPENGL

BAB III
HASIL

3.1 Source Code

Berikut Source code yang telah dibuat untuk membuat program alat Doraemon dibawah ini :

```
#include <GL/glew.h> // Menggunakan GLEW untuk mengakses semua
fungsi OpenGL di Windows
#include <GL/glut.h> // Menggunakan GLUT untuk membuat jendela dan
menangani input
#include <FreeImage.h> // Menggunakan FreeImage untuk memuat
gambar
#include <stdio.h> // Menggunakan stdio untuk fungsi printf
#include <math.h> // Menggunakan math untuk fungsi matematika

//-------------------------------------------------------Fathul-
-------------------------------------------------------//
float tx = 0.0f, ty = 0.0f, tz = 0.0f;    // Translasi
float rx = 0.0f, ry = 0.0f, rz = 0.0f;    // Rotasi
float sx = 1.0f, sy = 1.0f, sz = 1.0f;    // Skala
bool showAxis = true;                      // Toggle sumbu
//-------------------------------------------------------------------
-------------------------------------------------------//

//-------------------------------------------------------Sautan-
-------------------------------------------------------//
float tx2 = 0.0f, ty2 = 0.0f, tz2 = 0.0f;    // Translasi
float rx2 = 0.0f, ry2 = 0.0f, rz2 = 0.0f;    // Rotasi
float sx2 = 1.0f, sy2 = 1.0f, sz2 = 1.0f;    // Skala
bool showAxis2 = true;                      // Untuk menampilkan
sumbu koordinat
GLuint texture_Earth_ID;
//-------------------------------------------------------------------
-------------------------------------------------------//

//-------------------------------------------------------
Wilyandi-------------------------------------------------------//
```

```
bool showAxis3 = true;      // Menentukan apakah sumbu koordinat
akan ditampilkan
bool isRotate = false;      // Menandakan apakah baling-baling
sedang berputar
bool is2DMode = false;      // Menentukan apakah mode tampilan
adalah 2D
float angle = 0.0f;         // sudut rotasi baling-baling
float speed = 3.0f;         // Mengatur kecepatan rotasi baling-
baling
//-----------------------------------------------------------------
-------------------------------------------------------//

//-------------------------------------------------------------Fathul-
-------------------------------------------------------//
void initHehe() { // Fungsi inisialisasi
    glClearColor(0.0, 0.682, 0.937, 1.0); // Mengatur warna
background menjadi biru
    glEnable(GL_DEPTH_TEST); // Mengaktifkan depth testing

    // Aktifkan pencahayaan
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);

    // Konfigurasi cahaya
    GLfloat light_position[] = {1.0, 1.0, 1.0, 0.0};
    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[] = {1.0, 1.0, 1.0, 1.0};

    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
    glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
}

void drawAxes() {
    if (!showAxis) return; // Jika showAxis bernilai false, maka
keluar dari fungsi
```

```
    glDisable(GL_LIGHTING); // Menonaktifkan pencahayaan
    glLineWidth(2.0); // Menentukan ketebalan garis sumbu
    glBegin(GL_LINES); // Menggambar sumbu koordinat
    // X-axis (merah)
    glColor3f(1.0, 0.0, 0.0);
    glVertex3f(-10.0, 0.0, 0.0);
    glVertex3f(10.0, 0.0, 0.0);
    // Y-axis (hijau)
    glColor3f(0.0, 1.0, 0.0);
    glVertex3f(0.0, -10.0, 0.0);
    glVertex3f(0.0, 10.0, 0.0);
    // Z-axis (biru)
    glColor3f(0.0, 0.0, 1.0);
    glVertex3f(0.0, 0.0, -10.0);
    glVertex3f(0.0, 0.0, 10.0);
    glEnd();
    glEnable(GL_LIGHTING); // Mengaktifkan pencahayaan kembali
}

void displayHehe() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); //
Membersihkan buffer warna dan kedalaman

    glLoadIdentity(); // Memuat matriks identitas ke matriks
modelview
    gluLookAt(5.0, 5.0, 15.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0); //
Menempatkan kamera pada posisi (5, 5, 15) dan mengarahkan ke titik
(0, 0, 0)

    drawAxes(); // Memanggil fungsi untuk menggambar sumbu
koordinat

    glPushMatrix(); // Menyimpan matriks model saat ini ke dalam
tumpukan
    // Transformasi
    glTranslatef(tx, ty, tz);
    glRotatef(rx, 1.0, 0.0, 0.0);
```

```
    glRotatef(ry, 0.0, 1.0, 0.0);
    glRotatef(rz, 0.0, 0.0, 1.0);
    glScalef(sx, sy, sz);

    // Material untuk torus (kuning)
    GLfloat yellow_material[] = {1.0, 1.0, 0.0, 1.0};
    glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE,
yellow_material);

    // Gambar solid torus
    glutSolidTorus(0.2, 1.5, 50, 50);

    glPopMatrix(); // Mengembalikan matriks model ke kondisi
sebelumnya
    glutSwapBuffers(); // Menukar buffer untuk menggambar (double
buffering)
}

void reshapeHehe(int w, int h) { // Fungsi untuk menangani event
reshape
    glViewport(0, 0, w, h); // Mengatur viewport sesuai ukuran
jendela
    glMatrixMode(GL_PROJECTION); // Beralih ke matriks proyeksi
    glLoadIdentity(); // Memuat matriks identitas ke matriks
proyeksi
    gluPerspective(45.0, (float)w/(float)h, 1.0, 100.0); //
Mengatur proyeksi perspektif
    glMatrixMode(GL_MODELVIEW); // Beralih ke matriks modelview
}

void keyboard(unsigned char key, int x, int y) { // Fungsi untuk
menangani event keyboard
    switch(key) {
        // Translasi
        case 'w': ty += 0.1f; break;
        case 's': ty -= 0.1f; break;
        case 'a': tx -= 0.1f; break;
        case 'd': tx += 0.1f; break;
```

```
        case 'q': tz -= 0.1f; break;
        case 'e': tz += 0.1f; break;

        // Rotasi
        case 'i': rx += 5.0f; break;
        case 'k': rx -= 5.0f; break;
        case 'j': ry -= 5.0f; break;
        case 'l': ry += 5.0f; break;
        case 'u': rz -= 5.0f; break;
        case 'o': rz += 5.0f; break;

        // Skala
        case '+': sx += 0.1f; sy += 0.1f; sz += 0.1f; break;
        case '-': sx -= 0.1f; sy -= 0.1f; sz -= 0.1f; break;

        // Toggle sumbu
        case 'h': showAxis = !showAxis; break;

        case 27: exit(0); break;
    }
    glutPostRedisplay(); // Meminta OpenGL untuk merender ulang
tampilan
}
//------------------------------------------------------------------
-----------------------------------------------------------//

//-----------------------------------------------------------Sautan-
-----------------------------------------------------------//
void keyboard2(unsigned char key, int x, int y) {
    switch(key) {
        // Translasi
       case 'w': ty2 += 0.1f; break; // Menambah posisi objek pada
sumbu Y (ke atas)
       case 's': ty2 -= 0.1f; break; // Mengurangi posisi objek
pada sumbu Y (ke bawah)
       case 'a': tx2 -= 0.1f; break; // Mengurangi posisi objek
pada sumbu X (ke kiri)
       case 'd': tx2 += 0.1f; break; // Menambah posisi objek pada
```

```
sumbu X (ke kanan)
       case 'q': tz2 -= 0.1f; break; // Mengurangi posisi objek
pada sumbu Z (menjauh)
       case 'e': tz2 += 0.1f; break; // Menambah posisi objek pada
sumbu Z (mendekat)



        // Rotasi
        case 'i': rx2 += 5.0f; break; // Menambah rotasi pada
sumbu X (memutar ke depan)
        case 'k': rx2 -= 5.0f; break; // Mengurangi rotasi pada
sumbu X (memutar ke belakang)
        case 'j': ry2 -= 5.0f; break; // Mengurangi rotasi pada
sumbu Y (memutar ke kiri)
        case 'l': ry2 += 5.0f; break; // Menambah rotasi pada
sumbu Y (memutar ke kanan)
        case 'u': rz2 -= 5.0f; break; // Mengurangi rotasi pada
sumbu Z (memutar ke arah negatif Z)
        case 'o': rz2 += 5.0f; break; // Menambah rotasi pada
sumbu Z (memutar ke arah positif Z)



        // Skala
        case '+': sx2 += 0.1f; sy2 += 0.1f; sz2 += 0.1f; break; //
Menambah skala objek pada semua sumbu (memperbesar)
        case '-': sx2 -= 0.1f; sy2 -= 0.1f; sz2 -= 0.1f; break; //
Mengurangi skala objek pada semua sumbu (memperkecil)



        // Toggle sumbu
        case 'h': showAxis2 = !showAxis2; break; //
Mengaktifkan/mematikan visualisasi sumbu koordinat



        case 27: exit(0); break; // Menutup program saat tombol
ESC (kode ASCII 27) ditekan

    }
```

```
    glutPostRedisplay(); // Meminta OpenGL untuk merender ulang
tampilan

}
//-----------------------------------------------------------------
-------------------------------------------------------//

//-------------------------------------------------------
Wilyandi-------------------------------------------------------//
//fungsi keyboard
void keyboard3(unsigned char key, int x, int y) {
    switch(key) {
        // Toggle sumbu
        case 'h': showAxis3 = !showAxis3; break; //
Mengaktifkan/mematikan visualisasi sumbu koordinat

        case 27: exit(0); break; // Menutup program saat tombol
ESC (kode ASCII 27) ditekan
    }
    glutPostRedisplay();
}

//inisialisasi
void initBambu() {
    glClearColor(0.0, 0.682, 0.937, 1.0); // Mengatur warna
background menjadi biru
    glEnable(GL_DEPTH_TEST); //Aktifkan depth test biar objek 3D
tdk saling tembus
}

// Kalau ukuran jendela berubah,sesuaikan perspektifnya
void reshapeBambu(int w, int h) {
    glViewport(0, 0, w, h); //set area gambar
    glMatrixMode(GL_PROJECTION); //ganti ke mode proyeksi untuk
atur perspektif
    glLoadIdentity();  //Reset transformasi sebelumnya
    gluPerspective(45.0, (double)w / (double)h, 1.0,
100.0);  //Perspektif kamera
```

```
    glMatrixMode(GL_MODELVIEW); //Kembali ke mode gambar objek
}

//gambar kartesius
void drawAxesBambu() {
    if (!showAxis3) return; // Jika showAxis3 bernilai false, maka
keluar dari fungsi
    glLineWidth(2.0f); // Menentukan ketebalan garis sumbu
    glBegin(GL_LINES);  // Mulai menggambar garis

    // Sumbu X (merah)
    glColor3f(1.0f, 0.0f, 0.0f); //warna merah
    glVertex3f(-10.0f, 0.0f, 0.0f); //titik awal
    glVertex3f(10.0f, 0.0f, 0.0f); //titik akhir

    // Sumbu Y (hijau)
    glColor3f(0.0f, 1.0f, 0.0f);
    glVertex3f(0.0f, -10.0f, 0.0f);
    glVertex3f(0.0f, 10.0f, 0.0f);

    // Sumbu Z (biru)
    glColor3f(0.0f, 0.0f, 1.0f);
    glVertex3f(0.0f, 0.0f, -10.0f);
    glVertex3f(0.0f, 0.0f, 10.0f);

    glEnd();
}

// Gambar baling-baling
void drawPropeller() {
    glPushMatrix();  //Menyimpan keadaan matriks saat ini
    glRotatef(angle, 0.0f, 1.0f, 0.0f); //Putar baling-baling di
sumbu Y

    //Baling-baling
    glPushMatrix();
    glScalef(7.0f, 0.2f, 0.5f); //ukuran baling baling
    glColor3f(1.0f, 0.8f, 0.0f); //mengatur warna baling baling
```

```
    glutSolidCube(1.0f); //bentuk baling baling berupa cube
    glPopMatrix();  // Mengembalikan keadaan matriks ke sebelumnya

    glPopMatrix();  // Mengembalikan keadaan matriks ke sebelumnya

    //poros baling baling
    glColor3f(1.0f, 0.7f, 0.0f); //warna poros baling baling
    glutSolidSphere(0.5f, 20, 20); //poros berbentuk bola
}

//Gambar tiang baling baling
void drawStand() {
    //Tiang
    glPushMatrix();
    glColor3f(1.0f, 0.8f, 0.0f); //warna tiang
    glScalef(0.2f, 5.0f, 0.2f); //bentuk tiang panjang
    glutSolidCube(1.0f);
    glPopMatrix();

    //dasar tiang
    glPushMatrix();
    glColor3f(1.0f, 0.7f, 0.0f); //warna tiang
    glTranslatef(0.0f, -2.5f, 0.0f); //posisi dasar
    glutSolidSphere(0.8f, 20, 20); //dasar bentuk bola
    glPopMatrix();
}

//Fungsi utama untuk menggambar semua di window
void displayBambu() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
//Bersihkan layar
    glLoadIdentity();   //Menyetel ulang matriks transformasi

    gluLookAt(10.0, 10.0, 17.0,        //posisi kamera
              0.0, 0.0, 0.0,           //Fokus kamera
              0.0, 1.0, 0.0);          //Arah atas kamera

    drawAxesBambu();                   //Gambar kartesius
```

```
    drawStand();                          //Gambar tiang penyangga

    //pindahkan baling baling di atas tiang
    glPushMatrix();
    glTranslatef(0.0f, 2.5f, 0.0f);    //Posisi baling baling
    drawPropeller(); //memanggil objek baling baling bambu
    glPopMatrix();

    glutSwapBuffers();                 //Tampilkan ke layar
}

//fungsi untuk proses menu GUI
void processMenu(int option) {
    switch (option) {
        case 1:
            isRotate = true;    //Memutar baling baling
            break;
        case 2:
            isRotate = false;   //berhenti berputar
            break;
        case 3:
            exit(0);    //keluar dri program
            break;
    }
    glutPostRedisplay();
}

//Buat menu klik kanan
void createMenu() {
    glutCreateMenu(processMenu);    //Hubungkan menu dengan fungsi
processMenu
    glutAddMenuEntry("Putar", 1);   //Tambah opsi "Putar"
    glutAddMenuEntry("Berhenti", 2);    //Tambah opsi "Berhenti"
    glutAddMenuEntry("Keluar", 3);    //Tambah opsi "Keluar"
    glutAttachMenu(GLUT_RIGHT_BUTTON); //Tampilkan menu saat klik
kanan
}
```

```
//fungsi idle untuk mengatur rotasi baling baling
void idleBambu() {
    if (isRotate) {                    //Kalau isRotate true,
putar baling baling
        angle += speed;                //tambah sudut rotasi
        if (angle > 360.0f) angle -= 360.0f; //kembali ke 0 kalau
sudut lebih dari 360
    }
    glutPostRedisplay();
}
//-------------------------------------------------------------------
-----------------------------------------------------//

//-----------------------------------------------------------Sautan-
-----------------------------------------------------//
void createObject() {
    GLUquadric* object = gluNewQuadric(); // Membuat objek kuadrik
OpenGL untuk tekstur
    gluQuadricTexture(object, GL_TRUE); // Mengaktifkan tekstur
pada objek kuadrik
    gluQuadricNormals(object, GLU_SMOOTH); // Menambahkan normal
yang mulus untuk pencahayaan



    float panjang = 3.0f;  // Sesuaikan dengan ukuran yang
diinginkan
    float lebar = 2.0f;    // Sesuaikan dengan ukuran yang
diinginkan
    float tinggi = 0.5f;   // Sesuaikan dengan ukuran yang
diinginkan

    glBegin(GL_QUADS);
    // Sisi depan
    glTexCoord2f(0.0f, 0.0f); glVertex3f(-panjang, -tinggi,
lebar);
    glTexCoord2f(1.0f, 0.0f); glVertex3f(panjang, -tinggi, lebar);
    glTexCoord2f(1.0f, 1.0f); glVertex3f(panjang, tinggi, lebar);
    glTexCoord2f(0.0f, 1.0f); glVertex3f(-panjang, tinggi, lebar);
```

```
    // Sisi belakang
    glTexCoord2f(1.0f, 0.0f); glVertex3f(-panjang, -tinggi, -
lebar);
    glTexCoord2f(1.0f, 1.0f); glVertex3f(-panjang, tinggi, -
lebar);
    glTexCoord2f(0.0f, 1.0f); glVertex3f(panjang, tinggi, -lebar);
    glTexCoord2f(0.0f, 0.0f); glVertex3f(panjang, -tinggi, -
lebar);

    // Sisi atas
    glTexCoord2f(0.0f, 1.0f); glVertex3f(-panjang, tinggi, -
lebar);
    glTexCoord2f(0.0f, 0.0f); glVertex3f(-panjang, tinggi, lebar);
    glTexCoord2f(1.0f, 0.0f); glVertex3f(panjang, tinggi, lebar);
    glTexCoord2f(1.0f, 1.0f); glVertex3f(panjang, tinggi, -lebar);

    // Sisi bawah
    glTexCoord2f(1.0f, 1.0f); glVertex3f(-panjang, -tinggi, -
lebar);
    glTexCoord2f(0.0f, 1.0f); glVertex3f(panjang, -tinggi, -
lebar);
    glTexCoord2f(0.0f, 0.0f); glVertex3f(panjang, -tinggi, lebar);
    glTexCoord2f(1.0f, 0.0f); glVertex3f(-panjang, -tinggi,
lebar);

    // Sisi kanan
    glTexCoord2f(1.0f, 0.0f); glVertex3f(panjang, -tinggi, -
lebar);
    glTexCoord2f(1.0f, 1.0f); glVertex3f(panjang, tinggi, -lebar);
    glTexCoord2f(0.0f, 1.0f); glVertex3f(panjang, tinggi, lebar);
    glTexCoord2f(0.0f, 0.0f); glVertex3f(panjang, -tinggi, lebar);

    // Sisi kiri
    glTexCoord2f(0.0f, 0.0f); glVertex3f(-panjang, -tinggi, -
lebar);
    glTexCoord2f(1.0f, 0.0f); glVertex3f(-panjang, -tinggi,
lebar);
```

```
    glTexCoord2f(1.0f, 1.0f); glVertex3f(-panjang, tinggi, lebar);
    glTexCoord2f(0.0f, 1.0f); glVertex3f(-panjang, tinggi, -
lebar);
    glEnd();
}

void displayKonnyaku(){
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); //
Membersihkan buffer warna dan kedalaman
    glPushMatrix(); // Menyimpan matriks model saat ini ke dalam
tumpukan
    glTranslatef(tx2, ty2, tz2); // Melakukan translasi objek
berdasarkan nilai variabel
    glRotatef(rx2, 1.0, 0.0, 0.0); // Rotasi pada sumbu X
    glRotatef(ry2, 0.0, 1.0, 0.0); // Rotasi pada sumbu Y
    glRotatef(rz2, 0.0, 0.0, 1.0); // Rotasi pada sumbu Z
    glScalef(sx2, sy2, sz2); // Skala objek sesuai nilai variabel


    //BIND TExture ke Object di bawahnya, dengan ID yang sudah di
simpan
    //tadi
    glBindTexture(GL_TEXTURE_2D, texture_Earth_ID); //
Mengaktifkan tekstur pada objek
    createObject(); // Memanggil fungsi untuk menggambar objek


    glPopMatrix(); // Mengembalikan matriks model ke kondisi
sebelumnya
    glutSwapBuffers(); // Menukar buffer untuk menggambar (double
buffering)
    glutPostRedisplay(); // Meminta OpenGL untuk merender ulang
}

    GLuint textureID = 0;

int loadTexture(const char* path) {
    //untuk menyimpan Data Texture di ID spesifik!
```

```
    glGenTextures(1, &textureID);
    //kode di bawah untuk Memproses Pembacaan/Penyimpanan Buffer
dari
    //Gambar
    void* imgData;
    int imgWidth;
    int imgHeight;
    FREE_IMAGE_FORMAT format = FreeImage_GetFIFFromFilename(path);
// Mendapatkan format file gambar
    if (format == FIF_UNKNOWN) {
    printf("Unknown file type for texture image file %s\n", path);
    return -1;
 }
    FIBITMAP* bitmap = FreeImage_Load(format, path, 0); // Memuat
gambar
    if (!bitmap) {
    printf("Failed to load image %s\n", path);
    return -1;
 }
    FIBITMAP* bitmap2 = FreeImage_ConvertTo24Bits(bitmap); //
Mengonversi gambar ke 24-bit RGB
    FreeImage_Unload(bitmap);
    imgData = FreeImage_GetBits(bitmap2); // Mendapatkan data
piksel gambar
    imgWidth = FreeImage_GetWidth(bitmap2); // Mendapatkan lebar
gambar
    imgHeight = FreeImage_GetHeight(bitmap2); // Mendapatkan
tinggi gamba
    if (imgData) {
    printf("Texture image loaded from file %s, size %dx%d\n",
path,
    imgWidth, imgHeight);
    int format;
    if ( FI_RGBA_RED == 0 )
    format = GL_RGB;
    else
    format = GL_BGR;
    glBindTexture(GL_TEXTURE_2D, textureID); // Mengikat tekstur
```

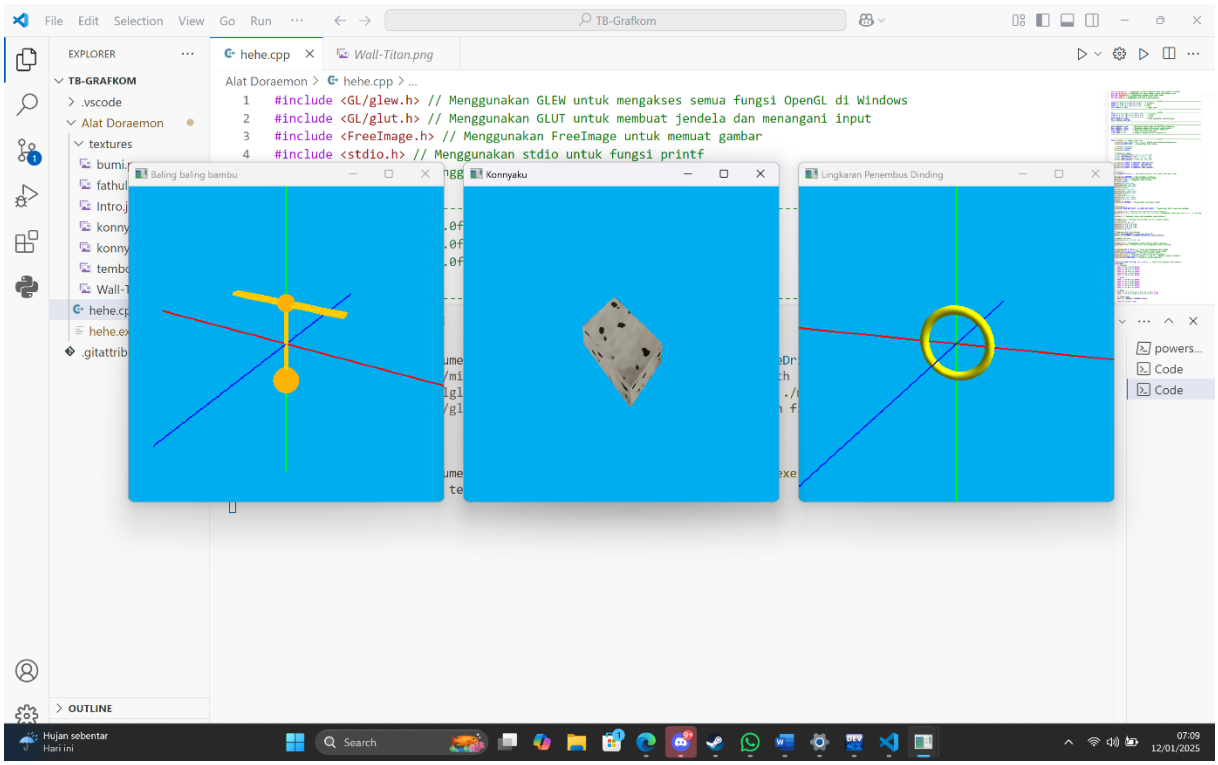
```
dengan ID tertentu
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, imgWidth, imgHeight,
0,
    format, GL_UNSIGNED_BYTE, imgData); // Mengisi data tekstur
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR); // Mengatur filter tekstur
    textureID++;
    return textureID-1;
 }
    else {
        printf("Failed to get texture data from %s\n", path);
 }
 return -1;
}

void initProjection(){
    glClearColor(0.0, 0.682, 0.937, 1.0); // Mengatur warna
background menjadi biru
    glEnable(GL_DEPTH_TEST); // Mengaktifkan pengujian kedalaman
untuk 3D
    glEnable(GL_POLYGON_SMOOTH); // Menghaluskan tepi poligon
    glShadeModel(GL_SMOOTH);
    //untuk meaktifkan texture di Polygon
    glEnable(GL_TEXTURE_2D);
    //untuk Mengubah matrik menjadi texture Rendering di OpenGL
    glMatrixMode(GL_TEXTURE);
    //END
    glMatrixMode(GL_PROJECTION); // Beralih ke matriks proyeksi
    glLoadIdentity(); // Memuat identitas awal
    gluPerspective(45.0, 1800/900, 1.0, 100.0); // Mengatur
proyeksi perspektif
    gluLookAt(10.0, 10.0, 10.0,
            0.0, 0.0, 0.0,
            0.0, 1.0, 0.0);
    glMatrixMode(GL_MODELVIEW); // Beralih ke mode matriks
model/view
    texture_Earth_ID = loadTexture("textures/konnyaku2.png"); //
Memuat tekstur dari file
```

```
    //texture_bulan_ID = loadTexture("textures/bulan.png");
}
//-----------------------------------------------------------------
-----------------------------------------------------//

int main(int argc, char** argv) { // Fungsi utama
    glutInit(&argc, argv); // Menginisialisasi GLUT
    glewInit(); // Menginisialisasi GLEW
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH); //
Mengatur mode tampilan GLUT

//-----------------------------------------------------------Fathul-
-----------------------------------------------------//
    glutInitWindowSize(400, 400); // Mengatur ukuran jendela
    glutInitWindowPosition(1000, 200); // Mengatur posisi
jendela
    int window1 = glutCreateWindow("Lingkaran Penembus Dinding");
// Membuat jendela dengan judul tertentu
    initHehe(); // Memanggil fungsi inisialisasi
    glutDisplayFunc(displayHehe); // Memanggil fungsi display
    glutReshapeFunc(reshapeHehe); // Memanggil fungsi reshape
    glutKeyboardFunc(keyboard); // Memanggil fungsi keyboard
//-----------------------------------------------------------------
-----------------------------------------------------//

//-----------------------------------------------------
Wilyandi-----------------------------------------------------//
    glutInitWindowSize(400, 400); // Mengatur ukuran jendela
    glutInitWindowPosition(150, 200); // Mengatur posisi jendela
    int window2 = glutCreateWindow("Baling baling bambu"); //
Membuat jendela dengan judul "Baling baling bambu"
    initBambu(); // Memanggil fungsi untuk inisialisasi pengaturan
OpenGL
    glutDisplayFunc(displayBambu); // Memanggil fungsi
displayBambu
    glutReshapeFunc(reshapeBambu); // memanggil fungsi
reshapeBambu
    glutKeyboardFunc(keyboard3); // memanggil fungsi keyboard
```

```
    glutIdleFunc(idleBambu); // memanggil fungsi idlebambu
    createMenu(); // Membuat menu klik kanan
//-------------------------------------------------------------------
-----------------------------------------------------//

//-----------------------------------------------------------Sautan-
-----------------------------------------------------//
    glutInitWindowSize(400, 400); // Mengatur ukuran jendela
menjadi 400x400 piksel
    glutInitWindowPosition(575, 200); // Mengatur posisi jendela
di layar pada koordinat (575, 200)
    int window3 = glutCreateWindow("Konnyaku"); // Membuat jendela
dengan judul "Konnyaku"
    initProjection(); // Memanggil fungsi untuk inisialisasi
pengaturan proyeksi
    glutDisplayFunc(displayKonnyaku); // Menetapkan fungsi
displayKonnyaku untuk menggambar objek
    glutKeyboardFunc(keyboard2); // Menetapkan fungsi keyboard2
untuk menangani input dari keyboard
//-------------------------------------------------------------------
-----------------------------------------------------//

    glutMainLoop(); //Memulai loop utama GLUT untuk menjalankan
program OpenGL

    return 0;
}
```

3.2 Output

Output dari program adalah 3 windows yang muncul bersamaan dengan ada masing-masing satu alat dari Doraemon yaitu Lingkaran Penembus Dinding, Konnyaku Penerjemah dan Baling-baling Bambu.



Gambar 3. 1 Output program

3.3 Penjelasan

Source code kami menghasilkan 3 alat Doraemon yaitu Lingkaran Penembus Dinding, Konnyaku Penerjemah dan Baling-baling Bambu. Dengan memunculkan 3 windows yang masing-masing terdapat satu alat dari Doraemon. Dengan menggunakan *glutCreateWindow* untuk membuat window dengan judul tertentu serta memanggil beberapa *function* untuk setiap objek kita bisa memunculkan ketiga objek sekaligus dalam 1 program. Berikut beberapa poin yang lebih jelas :

- int window1 = glutCreateWindow("Lingkaran Penembus Dinding"); // Membuat window pertama dengan judul “Lingkaran Penembus Dinding”
- glutInitWindowSize(400, 400); // Mengatur ukuran jendela
- glutInitWindowPosition(1000, 200); // Mengatur posisi jendela
- initHehe(); // Memanggil fungsi inisialisasi untuk Lingkaran Penembus Dinding
- glutDisplayFunc(displayHehe); // Memanggil fungsi display Lingkaran Penembus Dinding
- glutReshapeFunc(reshapeHehe); // Memanggil fungsi reshape Lingkaran Penembus Dinding
- glutKeyboardFunc(keyboard); // Memanggil fungsi keyboard Lingkaran Penembus

Dinding

- int window2 = glutCreateWindow("Baling baling bambu"); // Membuat window dengan judul "Baling baling bambu"
- initBambu(); // Memanggil fungsi untuk inisialisasi
- glutDisplayFunc(displayBambu); // Memanggil fungsi displayBambu
- glutReshapeFunc(reshapeBambu); // memanggil fungsi reshapeBambu
- glutKeyboardFunc(keyboard3); // memanggil fungsi keyboard
- glutIdleFunc(idleBambu); // memanggil fungsi idlebambu agar bambu berputar
- createMenu(); // Memanggil fungsi menu
- int window3 = glutCreateWindow("Konnyaku"); // Membuat window dengan judul "Konnyaku"
- initProjection(); // Memanggil fungsi untuk inisialisasi pengaturan proyeksi
- glutDisplayFunc(displayKonnyaku); // Menetapkan fungsi displayKonnyaku untuk menggambar objek
- glutKeyboardFunc(keyboard2); // Menetapkan fungsi keyboard2 untuk menangani input dari keyboard

BAB IV

4.1. Kesimpulan

Dari praktikum yang dilakukan, dapat disimpulkan bahwa tujuan praktikum telah tercapai dengan baik. Implementasi pembuatan objek 3D alat-alat Doraemon menggunakan OpenGL berhasil dilakukan melalui serangkaian tahapan, mulai dari konfigurasi OpenGL pada Dev C++ hingga pengaplikasian transformasi geometri seperti translasi, rotasi, dan skala.

Adapun beberapa hal yang dapat dipelajari selama praktikum ini adalah sebagai berikut:

1. Pemahaman mendalam tentang penggunaan library OpenGL untuk membangun objek 3D.
2. Cara menerapkan pencahayaan dan material pada objek 3D untuk menghasilkan visualisasi yang lebih realistis.
3. Teknik implementasi tekstur untuk meningkatkan detail visual pada objek 3D.
4. Pentingnya transformasi geometri dalam memanipulasi objek secara interaktif.

Melalui praktikum ini, pemahaman mengenai konsep dasar grafika komputer 3D, khususnya dengan menggunakan OpenGL, semakin mendalam. Hasil yang diperoleh juga menunjukkan potensi besar OpenGL dalam pengembangan aplikasi grafis yang realistis dan interaktif.

DAFTAR PUSTAKA

[1] A. Wahyudi and R. Fadilah, “Transformasi Geometri dalam Visualisasi 3D Berbasis OpenGL,” *Jurnal Informatika dan Rekayasa Perangkat Lunak*, 2022.

[2] A. Fauzi and T. Riyanto, *Pengantar Grafika Komputer: Dasar-dasar dan Implementasi OpenGL*. Yogyakarta: Andi Offset, 2020.

[3] N. H. Sari and R. Hidayat, “Penggunaan Teknologi OpenGL dalam Pembuatan Visualisasi 3D untuk Media Pembelajaran Interaktif,” *Jurnal Teknologi Informasi*, 2019.