LAPORAN PRATIKUM GRAFIK KOMPUTER

Diajukan untuk memenuhi Tugas mata kuliah Pratikum Grafik Komputer

PENERAPAN TRANSFORMASI 3D PADA OBJEK MAKANAN MANIS

Dosen Pengampu : Sri Rahayu, M.Kom

Instruktur Pratikum : Arul Budi Kalimat, S.Kom

Disusun oleh

Kelompok : 2

Fathir Miftah Nursalim
2306135

Salma Aulia Nisa
2306143

Gea Natasya
2306154

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 Penerapan Transformasi 3D pada Objek Makanan Manis dalam 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, 09 Januari 2025

Kelompok 2

DAFTAR ISI

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Teknologi grafika komputer telah berkembang pesat, terutama dalam bidang visualisasi tiga dimensi (3D). Salah satu implementasi yang menarik dari grafika komputer adalah pembuatan objek 3D dengan efek transformasi, seperti translasi, rotasi, dan skalasi. Transformasi ini memungkinkan objek untuk dimanipulasi dalam ruang, sehingga memberikan pengalaman visual yang dinamis dan interaktif.

OpenGL (Open Graphics Library) adalah salah satu pustaka yang populer digunakan dalam pengembangan aplikasi grafis. OpenGL menyediakan berbagai fungsi untuk menggambar dan memanipulasi objek grafis, mulai dari yang sederhana hingga kompleks. Dalam praktikum ini, OpenGL digunakan untuk membuat tiga objek 3D bertema makanan manis, yaitu donat, coklat batang, dan es kul-kul. Ketiga objek ini dirancang dengan efek transformasi 3D yang mencakup translasi, rotasi, dan skalasi untuk menampilkan dinamika visual yang realistis[1].

Pemrograman grafis dengan OpenGL tidak hanya bertujuan untuk menghasilkan visualisasi yang estetis, tetapi juga untuk memahami konsep dasar grafika komputer, seperti sistem koordinat Kartesius, pencahayaan (lighting), dan manipulasi geometri. Dengan memanfaatkan fungsi transformasi 3D, mahasiswa dapat mengembangkan keterampilan dalam merancang dan mengimplementasikan objek grafis yang kompleks.

Penelitian yang mendukung pentingnya penggunaan OpenGL dalam pendidikan grafika komputer menyebutkan bahwa integrasi antara teori dan praktik, seperti visualisasi objek 3D, dapat meningkatkan pemahaman mahasiswa tentang konsep-konsep grafis[2]. Selain itu, eksperimen dengan transformasi 3D juga memungkinkan eksplorasi yang lebih mendalam terhadap manipulasi geometri dalam ruang.

Perkembangan teknologi grafika komputer telah memberikan kontribusi besar dalam berbagai bidang, termasuk hiburan, simulasi, pendidikan, hingga desain produk. Dalam dunia pendidikan, penggunaan teknologi ini mempermudah pengajaran konsep-konsep yang kompleks melalui visualisasi yang menarik dan interaktif. Salah satu teknik yang sering digunakan dalam pemrograman grafika komputer adalah transformasi 3D, yang melibatkan translasi, rotasi, dan skalasi untuk memanipulasi objek dalam ruang. Transformasi ini memungkinkan pengguna untuk memahami bagaimana objek dapat dimodifikasi dan

dianimasikan secara dinamis.

Pemilihan tema makanan manis, seperti donat, coklat batang, dan es kul-kul, tidak hanya menambah daya tarik visual, tetapi juga memberikan kesempatan untuk mengaplikasikan berbagai teknik grafika pada bentuk yang berbeda-beda. Setiap objek memiliki karakteristik geometris yang unik, sehingga menantang mahasiswa untuk menerapkan prinsip dasar OpenGL secara kreatif. Selain itu, penggunaan sistem pencahayaan dan koordinat Kartesius dalam aplikasi ini bertujuan untuk memberikan pemahaman yang lebih mendalam tentang bagaimana aspek visual dapat dikelola secara efektif dalam ruang 3D.

Praktikum ini juga mendukung pengembangan keterampilan teknis mahasiswa dalam pemrograman dan desain grafis, yang merupakan keterampilan esensial di era digital saat ini. Dengan mengintegrasikan teori dan praktik, mahasiswa diharapkan dapat memahami bagaimana visualisasi berbasis komputer dapat digunakan untuk menciptakan simulasi yang realistis, estetis, dan interaktif. Hasil dari praktikum ini tidak hanya menjadi bukti pemahaman terhadap konsep grafika komputer, tetapi juga dapat menjadi dasar untuk pengembangan aplikasi grafis yang lebih kompleks di masa depan.

1.2 Rumusan Masalah

1. Apa yang dimkasud dengan OpenGL ?
2. Apa itu Scene dan Graphical User Interface (GUI)?
3. Bagaimana cara kerja dari OpenGL ?
4. Bagaimana membuat Penerapan Transformasi 3D pada Objek Makanan Manis dalam OpenGL?

1.3 Tujuan

1. Untuk mengethaui apa itu OpenGL
2. Untuk mengetahui apa itu Scene dan Graphical User Interface (GUI)
3. Untuk mengetahui cara kerja dari OpenGL
4. Untuk mengetahui cara pembuatan Penerapan Transformasi 3D pada Objek Makanan Manis dalam OpenGL

BAB II
TINJAUAN PUSTAKA

2.1 OpenGL

OpenGL (Open Graphics Library) adalah sebuah API (Application Programming Interface) yang digunakan untuk merender grafik 2D dan 3D. OpenGL dirancang untuk menyediakan antarmuka yang konsisten dan efisien bagi pengembang aplikasi grafis, memungkinkan mereka untuk menciptakan visualisasi yang kompleks dan interaktif. Sejak diperkenalkan oleh Silicon Graphics Inc. pada tahun 1992, OpenGL telah menjadi salah satu standar industri untuk pemrograman grafis dan digunakan secara luas dalam berbagai aplikasi, mulai dari game hingga simulasi ilmiah dan visualisasi data[3]. Berikut adalah karakteristik dari OpenGL yaitu:

1) Cross-Platform dan Multi-Bahasa

OpenGL bersifat lintas platform, yang berarti dapat dijalankan di berbagai sistem operasi seperti Windows, Linux, dan macOS. Selain itu, OpenGL mendukung berbagai bahasa pemrograman termasuk C, C++, Python, dan Java. Ini memberikan fleksibilitas kepada pengembang untuk memilih bahasa yang mereka kuasai atau yang paling sesuai untuk proyek mereka.

2) Antarmuka Pemrograman Aplikasi (API)

OpenGL menyediakan lebih dari 250 fungsi yang memungkinkan pengembang untuk menggambar objek grafis menggunakan bentuk primitif seperti titik, garis, dan poligon. API ini dirancang agar mudah digunakan dan memungkinkan pengembang untuk fokus pada logika aplikasi tanpa harus terlalu terlibat dengan detail implementasi perangkat keras.

3) Akselerasi Perangkat Keras

OpenGL dirancang untuk memanfaatkan kemampuan Graphics Processing Unit (GPU) modern. Dengan menggunakan GPU, OpenGL dapat melakukan rendering grafis dengan kecepatan tinggi dan efisiensi yang lebih baik dibandingkan dengan pemrosesan CPU saja. Hal ini sangat penting dalam aplikasi real-time seperti game di mana kinerja sangat krusial.

2.2 Scene

Scene adalah lingkungan yang dapat ditampilkan pada satu layar penuh. Dalam dunia game, scene dapat merujuk ke layar menu utama, gameplay, atau game over screen. Konsep scene di OpenGL mirip dengan halaman atau view di aplikasi modern[4].Manajemen scene

adalah proses mengatur elemen-elemen di dalamnya serta beralih di antara beberapa scene. Dalam pengembangan grafik komputer, ini penting untuk menciptakan aplikasi yang interaktif dan berlapis, seperti game atau simulasi.Manajemen scene melibatkan:

a. Inisialisasi: Menyiapkan elemen-elemen yang dibutuhkan untuk scene tertentu.
b. Rendering: Menampilkan elemen-elemen scene pada layar.
c. Transisi Antar Scene: Berpindah dari satu scene ke yang lain (misalnya dari menu screen
d. ke main game)[4].

Penggunaan Scene:

1) Game Development: Setiap level atau menu dalam game biasanya diimplementasikan sebagai scene.
2) Simulasi 3D: Dalam simulasi medis atau teknik, scene digunakan untuk memvisualisasikan data.
3) Aplikasi VR/AR: Scene menjadi ruang virtual yang kompleks dengan elemen interaktif[4].

2.3 Graphical User Interface (GUI)

GUI (Graphical User Interface) adalah antarmuka pengguna berbasis grafis yang memungkinkan interaksi dengan perangkat lunak menggunakan elemen visual seperti tombol, ikon, dan jendela, bukan hanya perintah teks[5].

Elemen GUI:

a. Tombol: Elemen interaktif yang dapat diklik untuk menjalankan suatu perintah.
b. Slider: Elemen GUI yang memungkinkan pengguna memilih nilai dari rentang tertentu.
c. Teks: Menampilkan informasi atau label di layar.
d. Input: Area di mana pengguna dapat memasukkan data, misalnya melalui keyboard.

2.4 Cara Kerja OpenGL

OpenGL beroperasi dalam model klien-server di mana aplikasi klien mengirimkan perintah rendering ke GPU[6]. Proses kerja OpenGL dapat diuraikan sebagai berikut:

1) Inisialisasi

Sebelum menggunakan OpenGL, pengembang harus menginisialisasi konteks OpenGL di mana semua perintah akan dieksekusi. Ini termasuk pengaturan jendela tampilan dan atribut rendering.

2) Definisi Objek Grafis

Pengembang mendefinisikan objek grafis dengan menggunakan vertex yang mewakili titik-titik pada objek tersebut. Atribut seperti warna dan tekstur juga ditentukan pada tahap ini.

3) Transformasi Geometri

Setelah objek didefinisikan, transformasi dilakukan untuk memanipulasi objek dalam ruang 3D. Transformasi ini mencakup translasi (pergeseran posisi), rotasi (perputaran), dan skala (perubahan ukuran). Ini dilakukan dengan menggunakan matriks transformasi.

4) Rasterisasi

Setelah objek ditransformasikan, proses rasterisasi mengubah representasi geometris menjadi pixel yang akan ditampilkan di layar. Proses ini menentukan pixel mana yang terpengaruh oleh primitif yang sedang diproses.

5) Pencahayaan dan Shading

OpenGL mendukung berbagai teknik pencahayaan untuk memberikan efek visual yang realistis pada objek. Teknik shading seperti flat shading, Gouraud shading, dan Phong shading digunakan untuk menambahkan kedalaman dan detail pada objek.

6) Rendering

Setelah semua perhitungan selesai, OpenGL merender gambar akhir ke dalam framebuffer. Framebuffer adalah area memori tempat gambar akhir disimpan sebelum ditampilkan di layar[6].

2.5 Penerapan Transformasi 3D pada Objek Makanan Manis Di OpenGL

Program ini berjudul Penerapan Transformasi 3D pada Objek Makanan Manis di OpenGL yang Dimana mengimplementasikan teknik transformasi 3D untuk memvisualisasikan objek-objek makanan manis, seperti donat, coklat, dan es kul-kul. Aplikasi ini dirancang menggunakan pustaka OpenGL dan memanfaatkan fungsi-fungsi dasar untuk melakukan translasi, rotasi, dan skala pada objek 3D. Visualisasi dimulai dengan adegan loading sebelum pengguna dapat melihat atau berinteraksi dengan objek.

Fungsi utama seperti Donat, Coklat, dan EsKulKul bertugas menggambarkan setiap jenis objek dengan detail. Donat ditampilkan dengan seres yang tersebar secara radial, batang coklat dipecah menjadi segmen-segmen, sementara es kul-kul memadukan bentuk silinder dan bola berwarna-warni. Transformasi objek dilakukan melalui pengaturan variabel global untuk rotasi, translasi, dan skala. Misalnya, pengguna dapat menambah atau mengurangi skala dengan menekan tombol + atau -, serta memindahkan posisi objek menggunakan tombol W(atas), A(kiri), S(bawah), dan D(kanan).

Program ini juga mendukung pencahayaan 3D yang realistis melalui pengaturan ambient dan diffuse lighting. Sistem navigasi melibatkan menu interaktif yang memungkinkan pengguna memilih objek yang ingin ditampilkan. Selain itu, fungsi koordinat Kartesius digunakan untuk menampilkan sistem sumbu sebagai panduan navigasi 3D.

Keseluruhan, program ini menunjukkan penerapan transformasi 3D yang interaktif dan estetis, serta memberikan pengalaman visual yang menarik melalui elemen-elemen makanan manis.

BAB III
HASIL

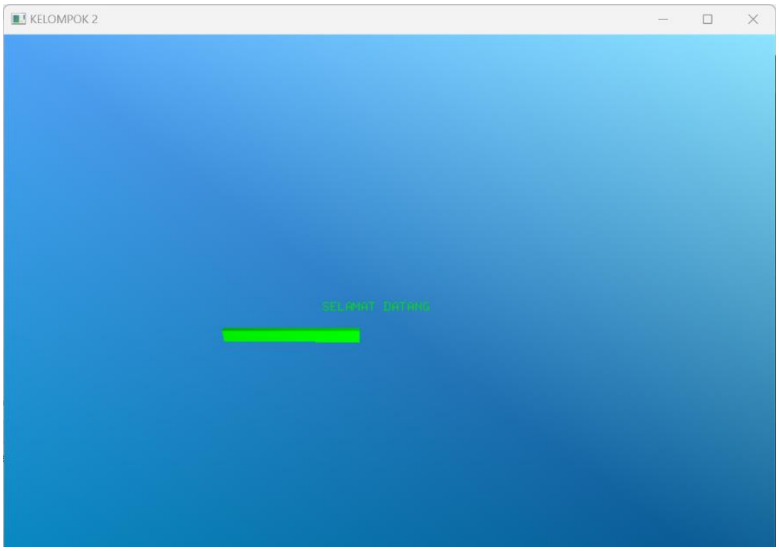
3.1 Source Code

Untuk source codenya bisa dilihat di Github dibawah ini:

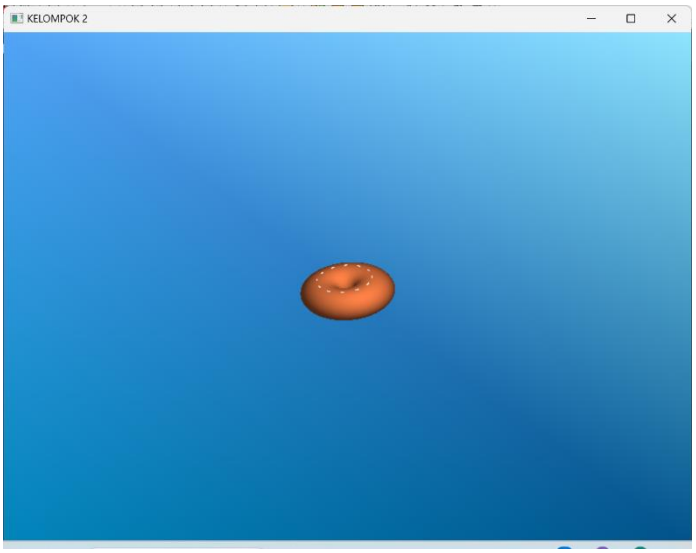
```
https://github.com/SalmaAulia29/TB-PGRAFIKA-KOMPUTER
```

3.2 Output

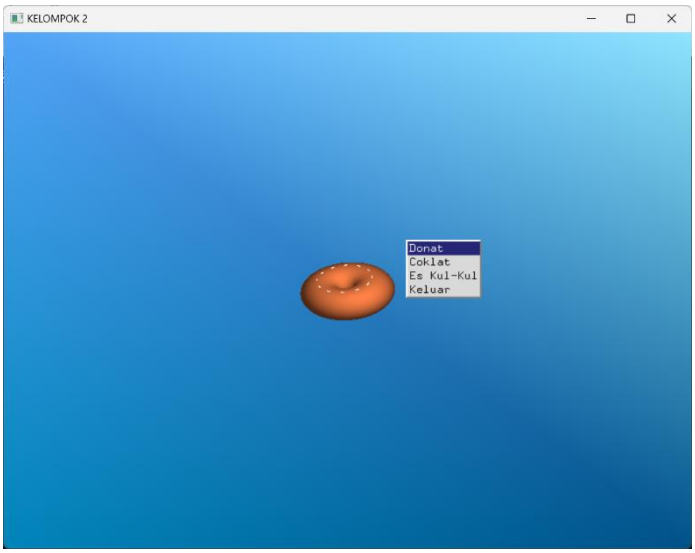
Output dari hasil source code diatas adalah:



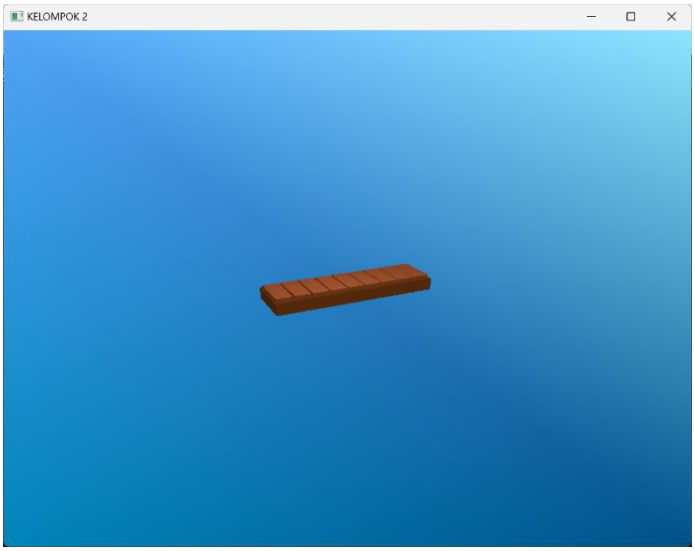
Gambar 1 Muncul Loading Scene



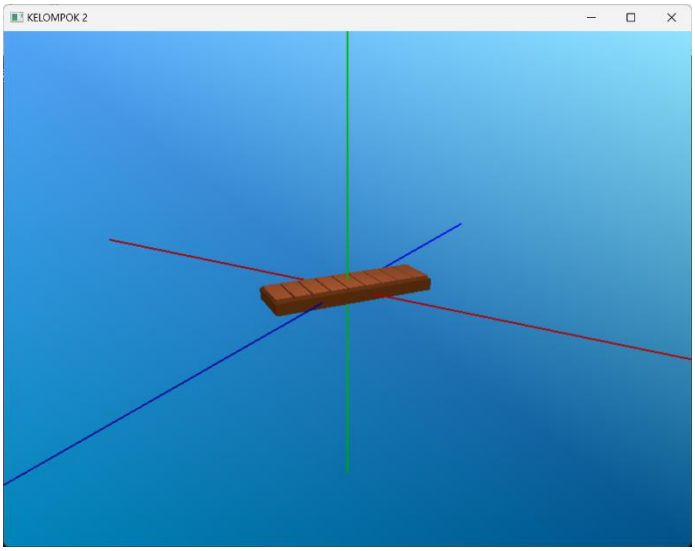
Gambar 2 Saat Loading selesai maka akan beralih ke Donat

- 

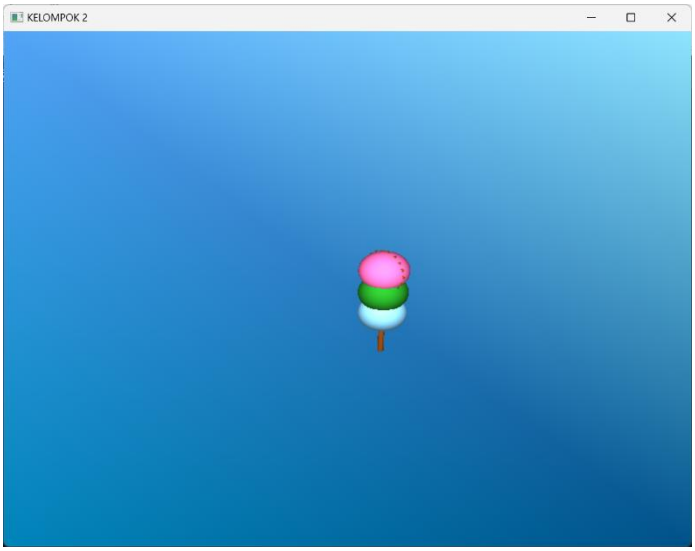
Gambar 3 GUI yang berisi menu untuk pindah ke objek yang lainnya



Gambar 4 objek Coklat



Gambar 5 Klik C maka akan muncul sebuat koordinat kartesius pada semua objek



Gambar 6 Objek Es Kulkul

3.3 Penjelasan

Berikan penjelasan dari source code kalian, dan jelaskan output yang dihasilkan ? mengapa source code yang ada menghasilkan output seperti itu? Jelaskan dengan deskripsi paragraph dan poin-poin

Kode dan Penjelasan

`#include <iostream>`
`#include <GL/glut.h>`
`#include <GL/freeglut.h>`
`#include <math.h>`
`using namespace std; // unt`

Kode ini mengimpor beberapa library yang diperlukan untuk membuat aplikasi grafis menggunakan OpenGL dan GLUT, serta menangani input/output di C++.

- #include <iostream>: Mengimpor pustaka standar untuk input/output seperti cout (untuk menampilkan informasi ke layar) dan cin (untuk mengambil input dari pengguna).
- #include <GL/glut.h>: Mengimpor pustaka GLUT yang memungkinkan pembuatan jendela dan pengelolaan input/output grafis.
- #include <GL/freeglut.h>: Mengimpor pustaka FreeGLUT, yang merupakan versi bebas dari GLUT untuk mendukung pengembangan aplikasi grafis menggunakan OpenGL.
- #include <math.h>: Mengimpor pustaka matematika untuk fungsi-fungsi matematis seperti trigonometri (misalnya sin, cos) yang sering digunakan dalam grafis 3D.
- using namespace std;: Menggunakan namespace standar C++ agar kita dapat mengakses fitur dari pustaka standar (seperti cout, cin, endl) tanpa perlu menuliskan std:: di depan setiap penggunaan.

```
struct Movement {
    float rotate = 0.0;
    float scale = 1.0;
    float translateX = 0.0;
    float translateY = 0.0;
} donat, coklat, esKulKul;
```

Struktur Movement digunakan untuk mendefinisikan transformasi objek dalam aplikasi grafis 3D. Setiap objek, seperti donat, coklat, dan esKulKul, memiliki atribut berikut:

- rotate: Menyimpan nilai rotasi objek (dalam derajat).
- scale: Menyimpan faktor skala objek.
- translateX: Menyimpan nilai translasi pada sumbu X.
- translateY: Menyimpan nilai translasi pada sumbu Y.

Setiap objek memiliki nilai awal yang berbeda, dan atribut ini akan diperbarui selama eksekusi untuk mengatur posisi, skala, dan rotasi objek dalam ruang 3D.

```
bool isMoving = true;
bool is2DMode = false;
float lightPos[] = {10.0f, 10.0f, 10.0f, 1.0f};
int Object = 1;
bool hiddenCarte = false;
bool Loading = true;
float loadingProgress = 0.0;
```

Kode tersebut mendefinisikan beberapa variabel global yang digunakan dalam aplikasi grafis 3D, kemungkinan besar terkait dengan OpenGL:

isMoving: Sebuah flag boolean yang menandakan apakah objek sedang bergerak atau dianimasikan. Secara default, nilainya `true`, artinya objek akan bergerak kecuali jika diubah.

- is2DMode: Sebuah flag boolean untuk mengubah antara mode 2D dan 3D. Secara default, nilainya `false`, yang berarti aplikasi dimulai dalam mode 3D.
- lightPos[]: Sebuah array yang mendefinisikan posisi sumber cahaya dalam scene. Posisi ini direpresentasikan oleh koordinat (10.0f, 10.0f, 10.0f) dalam ruang 3D, dan `1.0f` di akhir menunjukkan bahwa cahaya berada di ruang dunia (bukan sebagai cahaya arah).
- Object: Sebuah integer untuk melacak objek mana yang sedang dipilih (1 untuk Donat, 2 untuk Coklat, 3 untuk Es Kul-Kul). Variabel ini memungkinkan interaksi dengan objek tertentu dalam scene.
- hiddenCarte: Sebuah flag boolean yang menentukan apakah elemen tertentu (kemungkinan besar "Carte") disembunyikan atau tidak. Secara default, nilainya `false`, yang berarti elemen tersebut terlihat.
- Loading: Sebuah flag boolean untuk melacak apakah proses pemuatan sedang berlangsung. Secara default, nilainya `true`, yang mungkin mengontrol visibilitas atau animasi layar pemuatan.
- loadingProgress: Sebuah nilai float yang melacak kemajuan proses pemuatan.

```
void Kartesius() {
    glLineWidth(2.0);
    glBegin(GL_LINES);
    // Sumbu X (Merah)
    glColor3f(1.0, 0.0, 0.0);
    glVertex3f(-50.0, 0.0, 0.0);
    glVertex3f(50.0, 0.0, 0.0);
    // Sumbu Y (Hijau)
    glColor3f(0.0, 1.0, 0.0);
    glVertex3f(0.0, -50.0, 0.0);
    glVertex3f(0.0, 50.0, 0.0);
    // Sumbu Z (Biru)
    glColor3f(0.0, 0.0, 1.0);
    glVertex3f(0.0, 0.0, -50.0);
    glVertex3f(0.0, 0.0, 50.0);
    glEnd();
}
```

Fungsi `Kartesius` digunakan untuk menggambar sistem koordinat 3D menggunakan tiga sumbu, yaitu sumbu X, Y, dan Z. Fungsi ini dimulai dengan mengatur lebar garis menggunakan `glLineWidth(2.0)`. Kemudian, dengan menggunakan

`glBegin(GL_LINES)`, tiga garis digambar untuk merepresentasikan sumbu X, Y, dan Z. Sumbu X digambar dengan warna merah, sumbu Y dengan warna hijau, dan sumbu Z dengan warna biru. Setiap sumbu digambar dengan dua titik, yang menentukan panjang dan arah dari sumbu tersebut di ruang 3D. Setelah menggambar, `glEnd()` digunakan untuk menandai akhir dari penggambaran garis. Fungsi ini memberikan visualisasi dasar untuk sistem koordinat 3D pada tampilan grafis.

```
void Text(float x, float y, float z, string text, int fontSize) {
    glColor3ub(0, 255, 0);
    glPushMatrix();
    glTranslatef(x, y, z);
    glScalef(fontSize / 8.0f, fontSize / 13.0f, 1.0f);
    glRasterPos2f(0, 0);
    glutBitmapString(GLUT_BITMAP_8_BY_13, (const unsigned char *)text.c_str());
    glPopMatrix();
}
```

Fungsi `Text` digunakan untuk menggambar teks pada layar dengan menggunakan OpenGL. Fungsi ini menerima parameter posisi `(x, y, z)` untuk menentukan lokasi teks, teks itu sendiri sebagai string, dan ukuran font yang akan digunakan. Pertama, fungsi ini mengatur warna teks menjadi hijau dengan `glColor3ub`. Kemudian, dengan menggunakan transformasi `glTranslatef`, teks dipindahkan ke posisi yang diinginkan. Skala font disesuaikan berdasarkan parameter `fontSize` untuk mengubah ukuran teks sesuai kebutuhan. Fungsi `glRasterPos2f` digunakan untuk menentukan posisi awal teks pada layar, dan teks kemudian digambar dengan `glutBitmapString` menggunakan font bitmap `GLUT_BITMAP_8_BY_13`. Setelah menggambar teks, transformasi yang diterapkan pada matriks dipulihkan dengan `glPopMatrix`.

```
void LoadingScene() {
    glPushMatrix();
    glColor3ub(255, 255, 255);
    Text(-10.0, 0.0, 0.0, "SELAMAT DATANG", 90);

    glPushMatrix();
    glRotatef(30, 0, 1, 0);
    glColor3ub(0, 255, 0);
    float progressWidth = (loadingProgress / 100.0) * 40.0;
    glTranslatef(-20.0 + progressWidth / 2.0, -2.0, 0.1);
    glScalef(progressWidth, 2.0, 1.0);
    glutSolidCube(1.0);
    glPopMatrix();

    loadingProgress += 1.0;
    if (loadingProgress >= 100.0) {
        Loading = false;
        loadingProgress = 0.0;
        Object = 1;
    }
    glPopMatrix();
}
```

Fungsi `LoadingScene` menggambar tampilan layar pemuatan dengan teks dan indikator progres. Fungsi ini pertama-tama menampilkan teks "SELAMAT DATANG" di posisi tertentu dengan rotasi menggunakan `Text`, kemudian menggambar indikator progres berupa kubus yang panjangnya disesuaikan dengan nilai `loadingProgress`. Kubus ini diwarnai hijau dan ditranslasikan serta diskalakan berdasarkan lebar progres. Progres muatan bertambah sedikit demi sedikit setiap kali fungsi ini dipanggil, dan jika mencapai

100%, maka pemuatan dianggap selesai (`Loading` diset ke `false`), dan objek yang ditampilkan akan berubah menjadi objek pertama (`Object = 1`). Setelah itu, progres pemuatan direset kembali ke 0.

```
void Background() {
    glDisable(GL_DEPTH_TEST);
    glBegin(GL_QUADS);

    // Warna sudut kiri bawah
    glColor3f(0.0f, 0.5f, 0.7f);
    glVertex2f(-1.0f, -1.0f);

    // Warna sudut kanan bawah
    glColor3f(0.0f, 0.3f, 0.5f);
    glVertex2f(1.0f, -1.0f);

    // Warna sudut kanan atas
    glColor3f(0.5f, 0.8f, 1.0f);
    glVertex2f(1.0f, 1.0f);

    // Warna sudut kiri atas
    glColor3f(0.3f, 0.6f, 0.9f);
    glVertex2f(-1.0f, 1.0f);

    glEnd();
    glEnable(GL_DEPTH_TEST);
}
```

Fungsi `Background` menggambar latar belakang dengan gradasi warna menggunakan OpenGL. Fungsi ini memulai dengan menonaktifkan pengujian kedalaman (`glDisable(GL_DEPTH_TEST)`) agar objek latar belakang tidak terganggu oleh objek lainnya. Kemudian, latar belakang digambar menggunakan primitif quad (persegi panjang) yang terdiri dari empat sudut, masing-masing diberi warna yang berbeda untuk menciptakan gradasi warna. Warna pada sudut-sudut persegi panjang ditentukan menggunakan `glColor3f`, yang mencakup variasi warna biru kehijauan di sudut kiri bawah, biru lebih gelap di sudut kanan bawah, biru muda di sudut kanan atas, dan biru lebih cerah di sudut kiri atas. Setelah menggambar, pengujian kedalaman diaktifkan kembali dengan `glEnable(GL_DEPTH_TEST)` untuk memastikan objek lainnya bisa di-render dengan benar setelah latar belakang.

```
void Donat() {
    glPushMatrix();
    glTranslatef(donat.translateX, donat.translateY, 0.0);
    glRotatef(donat.rotate, 1.0, 0.0, 0.0);
    glScalef(donat.scale, donat.scale, donat.scale);

    // Warna dasar donat
    glColor3ub(160, 82, 45);
    glutSolidTorus(3.0, 4.0, 50, 50);

    // Tambahkan seres (taburan) di atas donat
    glPushMatrix();
    glColor3ub(255, 255, 255);
    glTranslatef(0.0, 0.0, 3.1);
    for (int i = 0; i < 360; i += 30) {
        glPushMatrix();
        glRotatef(i, 0.0, 0.0, 1.0);
        glTranslatef(4.0, 0.0, 0.0);
        glRotatef(90.0, 1.0, 0.0, 0.0);
        glutSolidCylinder(0.1, 0.5, 20, 20);
        glPopMatrix();
    }
    glPopMatrix();

    glPopMatrix();
}
```

Fungsi `Donat` menggambar objek donat dengan menggunakan transformasi translasi,

rotasi, dan skala. Objek ini terdiri dari dua bagian utama: cincin donat dan taburan (seres). Cincin donat digambar dengan `glutSolidTorus` menggunakan warna coklat, sedangkan taburan di atas donat digambar sebagai silinder kecil (meises) dengan `glutSolidCylinder` yang diposisikan secara merata mengelilingi cincin donat. Taburan ini diputar menggunakan rotasi pada sumbu Z dan dipindahkan ke posisi yang tepat dengan translasi. Semua transformasi dilakukan secara independen menggunakan `glPushMatrix` dan `glPopMatrix` untuk memastikan setiap bagian memiliki transformasi yang terpisah.

```
void Coklat() {
    glPushMatrix();
    glTranslatef(coklat.translateX, coklat.translateY, 0.0);
    glRotatef(coklat.rotate, 0.0, 1.0, 0.0);
    glScalef(coklat.scale, coklat.scale, coklat.scale);

    // Warna batang coklat
    glColor3ub(139, 69, 19); // Warna coklat gelap untuk dasar batang

    // Batang utama (panjang)
    glPushMatrix();
    glScalef(10.0, 1.0, 3.0);
    glutSolidCube(1.0);
    glPopMatrix();

    // Segmen-segmen coklat
    glColor3ub(160, 82, 45); // Warna coklat sedikit lebih terang
    for (int i = -4; i <= 4; i++) {
        glPushMatrix();
        glTranslatef(i * 1.1, 0.5, 0.0); // Posisi segmen-segmen
        glScalef(1.0, 0.5, 2.5); // Skala segmen
        glutSolidCube(1.0);
        glPopMatrix();
    }

    glPopMatrix();
}
```

Fungsi `Coklat` menggambar objek coklat menggunakan transformasi seperti translasi, rotasi, dan skala. Objek ini terdiri dari dua bagian utama: batang coklat dan segmen-segmen coklat. Batang coklat digambar dengan bentuk kubus yang lebih besar menggunakan `glutSolidCube` dan diberi warna coklat gelap. Segmen-segmen coklat ditambahkan dalam bentuk kubus yang lebih kecil dan diposisikan secara horizontal menggunakan translasi dan skala, dengan warna coklat lebih terang. Semua bagian objek ini diposisikan dan dirotasi sesuai dengan transformasi yang diberikan, dan penggunaan `glPushMatrix` serta `glPopMatrix` memastikan bahwa setiap bagian memiliki transformasi yang independen.

```
void EsKulKul() {
    glPushMatrix();
    glTranslatef(esKulKul.translateX, esKulKul.translateY, 0.0);
    glRotatef(esKulKul.rotate, 0.0, 1.0, 0.0);
    glScalef(esKulKul.scale, esKulKul.scale, esKulKul.scale);

    // Membalik eskulkul
    glRotatef(180, 1.0, 0.0, 0.0);

    // Posisi dasar es kul-kul
    glTranslatef(0.0, 18.0, 0.0);

    // Stick es kul-kul
    glPushMatrix();
    glColor3ub(139, 69, 19);
    glTranslatef(0.0, 0.0, -7.0);
    glRotatef(90.0, 1.0, 0.0, 0.0);
    glutSolidCylinder(0.5, 10.0, 20, 20);

    // Es utama (lapisan 1: biru)
    glColor3ub(135, 206, 250);
    glTranslatef(0.0, 0.0, 10.0);
    glutSolidSphere(4.0, 50, 50);
```

```
    // Lapisan ketiga (merah muda)
    glColor3ub(255, 105, 180);
    glTranslatef(0.0, 0.0, 5.0);
    glutSolidSphere(4.0, 50, 50);

    // Tambahkan meises pada lapisan merah muda (hanya di
    glPushMatrix();
    glColor3ub(139, 69, 19); // Warna cokelat gelap untuk
    for (int i = 0; i < 360; i += 20) {
        for (int j = -90; j <= 90; j += 30) {
            glPushMatrix();
            glRotatef(i, 0.0, 1.0, 0.0);
            glRotatef(j, 1.0, 0.0, 0.0);
            glTranslatef(4.2, 0.0, 0.0);
            glRotatef(90, 1.0, 0.0, 0.0);
            glutSolidCylinder(0.1, 0.3, 10, 10);
            glPopMatrix();
        }
    }
    glPopMatrix();

    glPopMatrix();
    glPopMatrix();
}
```

Fungsi `EsKulKul` menggambar objek es kul-kul dengan menggunakan transformasi seperti translasi, rotasi, dan skala. Objek ini terdiri dari tiga lapisan bola es (biru muda, hijau, dan merah muda) yang digambar menggunakan `glutSolidSphere`, dengan stik es kul-kul yang digambar sebagai silinder coklat menggunakan `glutSolidCylinder`. Selain itu, meises (taburan coklat) ditambahkan pada lapisan merah muda dengan rotasi untuk mendistribusikan meises secara merata di sekitar bola. Setiap bagian objek diberi transformasi independen menggunakan `glPushMatrix` dan `glPopMatrix`, memastikan posisi dan orientasi yang tepat.

```
void processMenu(int option) {
    switch (option) {
        case 1:
            Object = 1;
            break;
        case 2:
            Object = 2;
            break;
        case 3:
            Object = 3;
            break;
        case 4:
            exit(0);
    }
    glutPostRedisplay();
}
```

Fungsi processMenu menangani pilihan yang dibuat oleh pengguna dari menu konteks yang telah dibuat sebelumnya. Fungsi ini menerima parameter option yang merupakan nilai ID dari item menu yang dipilih. Berdasarkan nilai option, fungsi ini melakukan aksi tertentu:

- Case 1: Jika pengguna memilih "Donat", variabel Object diubah menjadi 1, yang menandakan objek yang aktif adalah donat.
- Case 2: Jika pengguna memilih "Coklat", variabel Object diubah menjadi 2, yang menandakan objek yang aktif adalah coklat.

- Case 3: Jika pengguna memilih "Es Kul-Kul", variabel Object diubah menjadi 3, yang menandakan objek yang aktif adalah esKulKul.
- Case 4: Jika pengguna memilih "Keluar", program akan dihentikan dengan perintah exit(0).

Setelah proses pemilihan selesai, glutPostRedisplay() dipanggil untuk memperbarui tampilan, sehingga perubahan yang dilakukan (seperti objek yang aktif) langsung terlihat pada layar.

```
void Menu() {
    glutCreateMenu(processMenu);
    glutAddMenuEntry("Donat", 1);
    glutAddMenuEntry("Coklat", 2);
    glutAddMenuEntry("Es Kul-Kul", 3);
    glutAddMenuEntry("Keluar", 4);
    glutAttachMenu(GLUT_RIGHT_BUTTON);
}
```

Fungsi Menu di atas digunakan untuk membuat menu konteks (popup menu) yang muncul ketika pengguna mengklik tombol kanan mouse. Di dalam fungsi ini, glutCreateMenu(processMenu) membuat menu baru dan menghubungkannya dengan fungsi processMenu yang akan menangani aksi yang dipilih dari menu tersebut. Kemudian, beberapa item menu ditambahkan menggunakan glutAddMenuEntry, Donat dengan 1, Coklat dengan 2, dan Es kulkul dengan 3.

Terakhir, glutAttachMenu(GLUT_RIGHT_BUTTON) mengaitkan menu ini dengan tombol kanan mouse, sehingga menu akan muncul saat tombol kanan ditekan.

```
void update(int value) {
    if (isMoving) {
        donat.rotate += 1.0;
        coklat.rotate += 1.2;
        esKulKul.rotate += 0.8;
        donat.scale = 1.0 + 0.1 * sin(donat.rotate * 3.14 / 180.0);
        esKulKul.scale = 1.0 + 0.1 * cos(esKulKul.rotate * 3.14 / 180.0);
    }
    glutPostRedisplay();
    glutTimerFunc(1000 / 60, update, 0);
}
```

Fungsi Pembaruan memperbarui animasi objek dalam aplikasi Anda. Jika animasi diaktifkan (ditunjukkan oleh variabel "isMoving" menjadi true), fungsi ini menambahkan nilai rotasi ke objek yang ada:

Objek Donut diputar 1,0 derajat, dan objek Chocolate Rotasi meningkat sebesar 1,2 derajat. Dan objek esKulKul akan bertambah 0,8 derajat setiap kali fungsi ini dipanggil. Selain itu, fungsi ini juga menskalakan objek berdasarkan fungsi trigonometri.

Skala objek Donut berubah menurut fungsi sinus rotasi, dan skala objek EsKulKul berubah menurut kosinus. Fungsi rotasinya. fungsi. Kedua objek tersebut memiliki variasi skala yang menciptakan efek animasi dinamis. Setelah rotasi dan skala diperbarui, fungsi glutPostRedisplay() dipanggil untuk memperbarui tampilan sehingga perubahannya segera

terlihat. Fungsi "glutTimerFunc(1000 / 60, update, 0)" digunakan untuk menjadwalkan panggilan fungsi "update" berikutnya pada interval 1/60 detik, memastikan animasi lancar 60 bingkai per detik.

```
void keyboard(unsigned char key, int x, int y) {
    switch (key) {
        case '+':
            // Perbesar skala objek saat tombol '+
            if (Object == 1)
                donat.scale += 0.1;
            else if (Object == 2)
                coklat.scale += 0.1;
            else if (Object == 3)
                esKulKul.scale += 0.1;
            break;
        case '-':
            // Perkecil skala objek saat tombol '-
            if (Object == 1)
                donat.scale -= 0.1;
            else if (Object == 2)
                coklat.scale -= 0.1;
            else if (Object == 3)
                esKulKul.scale -= 0.1;
            break;
        case 'w':
        case 'W':
            // Geser objek ke atas
            if (Object == 1)
                donat.translateY += 1.0;
            else if (Object == 2)
                coklat.translateY += 1.0;
```

```
        case 's':
        case 'S':
            // Geser objek ke bawah
            if (Object == 1)
                donat.translateY -= 1.0;
            else if (Object == 2)
                coklat.translateY -= 1.0;
            else if (Object == 3)
                esKulKul.translateY -= 1.0;
            break;
        case 'd':
        case 'D':
            // Geser objek ke kanan
            if (Object == 1)
                donat.translateX += 1.0;
            else if (Object == 2)
                coklat.translateX += 1.0;
            else if (Object == 3)
                esKulKul.translateX += 1.0;
            break;
        case 'a':
        case 'A':
            // Geser objek ke kiri
            if (Object == 1)
                donat.translateX -= 1.0;
            else if (Object == 2)
                coklat.translateX -= 1.0;
```

```
        case 'c':
        case 'C':
            hiddenCarte = !hiddenCarte;
            break;
        case 'r':
        case 'R':
            isMoving = !isMoving;
            break;
    }
    glutPostRedisplay(); // Refresh tampilan
}
```

Kode di atas adalah implementasi fungsi keyboard yang menangani input dari pengguna menggunakan tombol keyboard. Fungsi ini berfungsi untuk mengubah tampilan objek grafis dalam aplikasi berdasarkan input yang diberikan.

Secara rinci, setiap kali pengguna menekan tombol tertentu, program akan merespons dengan mengubah properti objek yang sedang aktif. Penjelasan setiap tombol adalah sebagai berikut:

- Tombol + akan memperbesar skala
- Tombol - akan memperkecil skala
- Tombol w atau W akan menggeser objek ke atas pada sumbu Y
- Tombol s atau S akan menggeser objek ke bawah pada sumbu Y
- Tombol d atau D akan menggeser objek ke kanan pada sumbu X
- Tombol a atau A akan menggeser objek ke kiri pada sumbu X
- Tombol c atau C akan mengubah status variabel hiddenCarte, yang bisa digunakan untuk menampilkan atau menyembunyikan peta atau elemen lain dalam aplikasi.
- Tombol r atau R akan mengubah status variabel isMoving, yang mungkin digunakan untuk mengaktifkan atau menonaktifkan gerakan objek atau interaksi lainnya.

Setelah setiap input, fungsi glutPostRedisplay() dipanggil untuk menyegarkan tampilan dan memuat perubahan yang dilakukan oleh pengguna.

```
void display() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();

    glMatrixMode(GL_PROJECTION);
    glPushMatrix();
    glLoadIdentity();
    glOrtho(-1, 1, -1, 1, -1, 1);
    drawGradientBackground();
    glPopMatrix();

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    if (showLoading) {
        renderLoadingScene();
    } else {
        if (hiddenCarte) {
            Kartesius();
        }
        switch (currentObject) {
            case 1:
                Donat();
                break;
            case 2:
                Coklat();
                break;
            case 3:
                EsKulKul();
                break;
        }
    }

    glutSwapBuffers();
}
```

Fungsi `display` bertugas menggambar elemen visual di jendela OpenGL, meliputi latar belakang dan objek utama, dengan pembaruan tampilan yang mulus menggunakan double buffering. Proses dimulai dengan membersihkan buffer warna dan kedalaman menggunakan `glClear`, lalu mengatur ulang matriks transformasi dengan `glLoadIdentity`.

Latar belakang digambar pertama kali dalam mode proyeksi ortogonal dengan bantuan fungsi `Background`, di mana pengaturan matriks sebelumnya dipulihkan setelahnya. Selanjutnya, objek utama digambar dalam mode model-view. Jika variabel `Loading` bernilai true, fungsi `LoadingScene` akan menampilkan layar loading. Jika `Loading` selesai, sistem koordinat kartesius akan digambar jika `hiddenCarte` aktif. Berdasarkan nilai variabel `Object`, objek donat (`Donat`), cokelat (`Coklat`), atau es krim

(`EsKulKul`) akan ditampilkan di depan latar belakang.

Fungsi ini diakhiri dengan `glutSwapBuffers` untuk menampilkan frame baru secara lancar tanpa kedipan. Hasilnya adalah tampilan interaktif yang terdiri dari latar belakang, objek utama, dan sistem koordinat yang diperbarui secara real-time.

```
void init3D() {
    glEnable(GL_DEPTH_TEST); // Aktifkan depth testing
    glEnable(GL_COLOR_MATERIAL);

    // Pencahayaan
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    GLfloat ambientLight[] = {0.3f, 0.3f, 0.3f, 1.0f};
    GLfloat diffuseLight[] = {1.0f, 1.0f, 1.0f, 1.0f};
    glLightfv(GL_LIGHT0, GL_AMBIENT, ambientLight);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuseLight);
    glLightfv(GL_LIGHT0, GL_POSITION, lightPos);

    // Viewport
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(70.0, 1.0, 1.0, 100.0);
    gluLookAt(30.0, 30.0, 50.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
    glMatrixMode(GL_MODELVIEW);
    glClearColor(0.0, 0.0, 0.0, 1.0);
}
```

Fungsi `init3D` berfungsi mengatur dasar rendering 3D dengan mengaktifkan fitur-fitur penting seperti depth testing untuk memastikan tampilan objek sesuai urutan kedalaman, serta material warna untuk kontrol langsung warna objek. Sistem pencahayaan diaktifkan dengan `GL_LIGHTING` dan `GL_LIGHT0`, menyediakan efek cahaya ambient untuk pencahayaan menyeluruh dan cahaya difus untuk detail permukaan objek. Perspektif ditetapkan melalui `gluPerspective` untuk memberikan efek kedalaman realistis, dengan posisi kamera diatur oleh `gluLookAt`, memungkinkan sudut pandang yang tepat. Latar belakang layar diatur hitam untuk memberikan kontras yang tajam terhadap objek. Fungsi ini memastikan tampilan objek 3D lebih hidup dan sesuai perspektif.

```
int main(int argc, char **argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(800, 600);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("KELOMPOK 2_TB GRAFKOM_E");
    init3D();
    glutDisplayFunc(display);
    Menu();
    glutKeyboardFunc(keyboard);
    glutTimerFunc(1000 / 60, update, 0);
    glutMainLoop();
    return 0;
}
```

Program ini merupakan aplikasi grafik 3D menggunakan OpenGL dan GLUT untuk mengelola jendela, rendering, dan event. Setelah inisialisasi GLUT dengan pengaturan mode tampilan, ukuran, dan posisi jendela, fungsi `init3D` mengatur render 3D seperti depth testing dan pencahayaan. Fungsi `display` digunakan untuk merender ulang jendela, sedangkan `keyboard` menangani input dari pengguna. Animasi diperbarui secara berkala dengan `glutTimerFunc` untuk mencapai 60 frame per detik. Program berjalan dalam loop

utama `glutMainLoop`, menghasilkan jendela grafik 3D interaktif yang menampilkan objek, animasi, atau tampilan statis sesuai logika yang diatur.

BAB IV
PENUTUP

4.1. Kesimpulan

Dalam praktikum ini, kami berhasil menerapkan transformasi 3D pada objek bertema makanan manis menggunakan OpenGL. Berbagai teknik transformasi seperti translasi, rotasi, dan skala digunakan untuk menciptakan visualisasi interaktif pada objek seperti donat, coklat batang, dan es kul-kul. Praktikum ini berhasil mencapai tujuan yang telah ditetapkan, yaitu memahami konsep dasar OpenGL, seperti pembuatan dan manipulasi objek 3D, serta implementasi pencahayaan dan koordinat Kartesius.

Melalui praktikum ini, kami mempelajari dasar-dasar penggunaan OpenGL, termasuk fungsi-fungsi API untuk pemrograman grafis. Selain itu, kami memahami konsep transformasi geometri dalam ruang 3D, seperti translasi, rotasi, dan skala, serta pentingnya pencahayaan dan perspektif dalam menciptakan visualisasi yang realistis. Praktikum ini juga memberikan wawasan tentang teknik pemrograman berbasis event untuk mengelola interaksi pengguna melalui keyboard dan mouse.

Secara keseluruhan, praktikum ini tidak hanya memberikan pemahaman teknis mengenai penggunaan OpenGL, tetapi juga memperkuat keterampilan desain grafis dan pemrograman interaktif, yang sangat relevan di era digital saat ini.

DAFTAR PUSTAKA

[1] D. Suhardiman *et al.*, “Pembuatan Simulasi Pergerakan Objek 3D (Tiga Dimensi) Menggunakan OpenGL,” *J. Inform.*, vol. 2, no. 1, pp. 1–4, 2015.

[2] J. Fuller, “The Official Guide to Learning OpenGL,” *World Wide Web Internet Web Inf. Syst.*, vol. 4, no. January 1997, p. 257, 2000, [Online]. Available: ftp://raphaello.univ-fcomte.fr/pub/OpenGL/RedBook.pdf

[3] F. S. . Hill, “Computer graphics : using OpenGL,” p. 922, 2001.

[4] A. Tujuan, “Modul 8 – scene”.

[5] M. Gui, A. Tujuan, M. Gui, B. D. Teori, and E. Gui, “GUI (Graphical User Interface) adalah antarmuka pengguna berbasis grafis yang memungkinkan interaksi dengan perangkat lunak menggunakan elemen visual seperti tombol ,”.

[6] “Pengertian OpenGL dan Cara Kerjanya - dufatancom.id.” Accessed: Jan. 09, 2025. [Online]. Available: https://www.dufatancom.id/2019/08/pengertian-opengl-dan-cara-kerjanya.html