

LAPORAN PRATIUM GRAFIK KOMPUTER

Diajukan untuk memenuhi Tugas mata kuliah Pratikum Grafik Komputer

RUMAH DI MUSIM SALJU

Dosen Pengampu : Sri Rahayu, M.Kom.

Instruktur Pratikum : Arul Budi Kalimat, S.Kom



Disusun oleh :

Restu Bagja Maulud
2306043

Tsani Hisni Amala
2306052

Faren Anjani M
2306052

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN ILMU KOMPUTER

INSTITUT TEKNOLOGI GARUT

2025

KATA PENGANTAR

Puji dan syukur kehadiran Allah SWT atas segala rahmat dan karunia-Nya sehingga kami dapat menyelesaikan Laporan Praktikum Grafik Komputer ini. Laporan ini dibuat sebagai salah satu tugas dari mata kuliah Grafik Komputer, dengan tujuan untuk memberikan pemahaman yang lebih baik tentang Membuat Grafika Komputer Dengan 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 8

DAFTAR ISI

KATA PENGANTAR	i
DAFTAR ISI.....	ii
DAFTAR GAMBAR	Error! Bookmark not defined.
BAB I PENDAHULUAN	1
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah	1
1.3 Tujuan.....	1
BAB II TINJAUAN PUSTAKA.....	2
2.1 OpenGL.....	2
2.2 Konfigurasi OpenGL pada VS Code.....	2
2.3 Cara Kerja OpenGL	4
2.4 Rumah Di Musim Salju Di OpenGL.....	4
BAB III HASIL.....	6
3.1 Source Code.....	6
3.2 Output	17
3.3 Penjelasan.....	18
BAB IV	21
4.1. Kesimpulan.....	21
DAFTAR PUSTAKA	22

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 pencapaian yang cukup signifikan dalam dekade terakhir. Grafika komputer merupakan teknik dalam ilmu komputer dan matematika untuk merepresentasikan dan memanipulasikan gambar menggunakan komputer. Salah satu aplikasi yang komplit dari grafika komputer adalah untuk visualisasi data dalam bentuk grafika dua dimensi (2D) atau tiga dimensi (3D) dilengkapi dengan animasi. Perlu dikembangkan sebuah software yang mampu merepresentasikan visual dalam format tiga dimensi (3D) secara real time dengan memasukan data atau parameter-parameter yang diperoleh dari suatu alat eksternal. Tanpa perlu proses rendering sebuah file, melainkan langsung bisa di displaykan pada layar monitor. (Muhammad Adnani & Achmad Zakki Falani, 2021).

Grafika Komputer (Computer Graphic) adalah seperangkat alat yang terdiri dari hardware dan software untuk memproduksi suatu gambar, grafik atau citra realistic untuk seni, game computer, foto dan film animasi. Grafika Komputer merupakan bagian yang paling sulit di bidang computer, karena selain harus mengerti bahasa dan logika pemrograman juga dibutuhkan kemampuan analisis serta pemahaman matematik. Processing merupakan perangkat lunak open source yang menyediakan bahasa pemrograman dan lingkungan pemrograman bagi orang-orang yang ingin membuat program pengolahan citra, animasi dan suara. Processing digunakan dalam berbagai tahap seperti: belajar, membuat prototype sampai pada tahap produksi. Processing dibuat dengan tujuan untuk memberikan fasilitas belajar pemrograman computer dalam konteks visual. (Isi, 2016).

1.2 Rumusan Masalah

Di sini, Anda menyatakan masalah atau pertanyaan yang ingin dijawab melalui praktikum. Misalkan :

1. Apa yang dimaksud dengan OpenGL ?
2. Bagaimana cara mengkonfigurasi OpenGL pada Dev C++ atau VSCode?
3. Bagaimana cara kerja dari OpenGL ?
4. Bagaimana membuat Rumah di musim salju dalam OpenGL

1.3 Tujuan

Nyatakan tujuan dari laporan praktikum. Misalkan :

1. Mengetahi apa itu OpenGL
2. Mengetahui cara mengkonfigurasi OpenGL pada Dev C++ atau VSCode.
3. Mengetahui cara kerja dari OpenGL
4. Mengetahui cara pembuatan Rumah di musim salju dalam OpenGL

BAB II TINJAUAN PUSTAKA

2.1 OpenGL

Menggunakan OpenGL (Open Graphic Library) merupakan library yang terdiri dari berbagai macam fungsi dan digunakan untuk menggambar beberapa objek 2D dan 3D. library –library ini mendefinisikan sebuah Bahasa, cross platfrom API (antar muka pemrograman aplikasi) untuk menulis aplikasi yang menghasilkan objek 2D dan 3D grafis. Bahasa pemrograman yang digunakan pada umumnya adalah Bahasa C++.(Muhammad Adnani & Achmad Zakki Falani, 2021).

OpenGL dikembangkan oleh Silicon Graphics Inc (SGI) pada tahun 1992 dan secara luas digunakan dalam CAD, realitas maya, visualisasi ilmiah, visualisasi informasi, dan simulasi penerbangan (Priyantono & Rachmawan, 2020).

2.2 Konfigurasi OpenGL pada VS Code

Uraikan dan Jelaskan langkah-langkah untuk konfigurasi OpenGL pada VS Code. Setiap langkah harus menyertakan Screenshot. Semua gambar ukurannya harus sama dan posisi gambar berada ditengah (center) dan berikan keterangan dari setiap gambar. Contoh :

Pastikan VS Code sudah terinstal, Kemudian ikuti Langkah-langkah berikut:

Berikut adalah cara konfigurasi OpenGL pada VSCode:

1. Instalasi MinGW (GCC Compiler)
 - 1)Unduh dan instal MinGW: Anda dapat mengunduh MinGW dari situs resmi MinGW. Pastikan untuk menginstal gcc, g++, dan make.
 - 2)Tambahkan MinGW ke PATH:
 - Buka System Properties → Advanced → Environment Variables.
 - Edit variabel Path dan tambahkan path ke direktori bin dari instalasi MinGW Anda, seperti C:\MinGW\bin.
2. Instalasi FREEGLUT dan GLEW
 - 1)Unduh dan instal FREEGLUT: Dapatkan precompiled binaries dari situs FREEGLUT. Pilih versi yang sesuai dengan Windows 32-bit atau 64-bit.
 - 2)Unduh dan instal GLEW: Dapatkan versi terbaru dari situs GLEW.
3. Konfigurasi VSCode
 - 1)Buka VSCode dan buat folder proyek baru.
 - 2)Buat file konfigurasi build (tasks.json) di folder .vscode di root direktori proyek Anda:

```
{
  "version": "2.0.0",
  "tasks": [
    {
      "label": "build",
      "type": "shell",
      "command": "g++",
      "args": [
        "-g",
        "${file}",

```

```

        "-o",
        "${fileDirname}\\${fileBasenameNoExtension}.exe",
        "-Ipath_to_GLFW_include",
        "-Ipath_to_GLEW_include",
        "-Lpath_to_GLFW_lib",
        "-Lpath_to_GLEW_lib",
        "-lfreeglut",
        "-lglew32",
        "-lglu32",
        "-lopengl32",
        "-lgdi32"
    ],
    "group": {
        "kind": "build",
        "isDefault": true
    },
    "problemMatcher": [
        "$gcc"
    ],
    "detail": "Compiler: g++"
}
]
}

```

Ganti path_to_GLFW_include, path_to_GLFW_lib, path_to_GLEW_include, dan path_to_GLEW_lib dengan path yang sesuai di sistem Anda.

4. Build dan Run

- 1) Build: Tekan Ctrl+Shift+B untuk build program Anda.
- 2) Run: Buka terminal di VSCode dan jalankan executable yang telah dibuild, seperti ./main.exe.

2.3 Cara Kerja OpenGL

OpenGL (Open Graphics Library) adalah sebuah API (Application Programming Interface) yang digunakan untuk menggambar grafik 2D dan 3D dalam aplikasi komputer. Dalam konteks C++, OpenGL memungkinkan Anda untuk mengakses fungsi-fungsi grafis tingkat rendah melalui library yang dapat diintegrasikan ke dalam program C++.

Berikut adalah gambaran singkat cara kerja OpenGL dalam C++:

1. **Inisialisasi:** Program C++ yang menggunakan OpenGL biasanya dimulai dengan menginisialisasi konteks OpenGL. Ini melibatkan pembuatan jendela menggunakan library seperti GLFW atau SDL, yang memungkinkan program untuk berinteraksi dengan perangkat keras grafis (GPU).
2. **Mendefinisikan Data Grafis:** Anda mendefinisikan objek grafis menggunakan data seperti koordinat vertex (titik-titik yang membentuk objek 3D atau 2D). Data ini disimpan dalam buffer objek yang dikelola oleh OpenGL.
3. **Shader:** OpenGL menggunakan shader untuk menangani pemrosesan grafis, seperti pewarnaan dan pencahayaan. Anda menulis shader dalam GLSL (OpenGL Shading Language) dan mengompilasinya ke dalam program shader yang digunakan dalam pipeline grafis.
4. **Render Loop:** Setelah data dan shader disiapkan, Anda memasuki loop render. Dalam loop ini, program memanggil fungsi-fungsi OpenGL untuk menggambar objek ke layar, seperti `glDrawArrays` atau `glDrawElements`.
5. **Pencahayaan dan Tekstur:** OpenGL menyediakan berbagai fungsi untuk mengaplikasikan pencahayaan, tekstur, dan efek lainnya ke objek. Shader yang telah didefinisikan menangani sebagian besar perhitungan ini.
6. **Manajemen Buffer dan Framebuffer:** OpenGL menggunakan buffer untuk menyimpan data vertex dan gambar sementara. Penggunaan framebuffer memungkinkan pengolahan gambar sebelum ditampilkan ke layar, misalnya untuk efek pasca-pemrosesan.
7. **Menampilkan Hasil:** Setelah menggambar objek dan memproses efek grafis, hasilnya ditampilkan di layar dengan cara mengubah buffer yang digunakan oleh GPU untuk menampilkan gambar.

Secara keseluruhan, OpenGL dalam C++ memungkinkan pembuatan aplikasi grafis yang kompleks dengan menggunakan serangkaian fungsi dan shader yang efisien.

2.4 Rumah Di Musim Salju Di OpenGL

Kode di atas adalah program dalam bahasa C++ yang menggunakan pustaka OpenGL dan FreeImage untuk menggambar pemandangan 3D sederhana dengan elemen-elemen seperti bulan, rumah, pohon, dan boneka salju. Program ini mencakup berbagai fungsi untuk menggambar setiap elemen dengan detail tekstur dan pencahayaan.

Fungsi utama seperti `setupLighting()` digunakan untuk mengatur pencahayaan dalam adegan 3D. Pencahayaan melibatkan konfigurasi cahaya ambient, difus, dan spekulat untuk menciptakan efek yang realistis. Fungsi `updateLightPosition()` memperbarui posisi cahaya untuk memberikan efek rotasi cahaya secara dinamis, sehingga menciptakan ilusi cahaya bergerak.

Elemen utama pemandangan digambar menggunakan fungsi tertentu. Misalnya, `drawMoon()` bertanggung jawab untuk menggambar bulan yang bertekstur, sedangkan `drawCube()` digunakan untuk membuat kubus sederhana yang dapat diimplementasikan dalam elemen lain, seperti cerobong rumah. Fungsi `drawHouse()` mendetailkan pembuatan rumah dengan dinding bertekstur, atap berbentuk segitiga, cerobong asap, pintu, dan jendela.

Boneka salju digambar menggunakan fungsi `drawSnowman()` yang terdiri dari bola-bola putih berukuran berbeda untuk membentuk tubuhnya. Fungsi `drawTree()` digunakan untuk membuat pohon yang terdiri dari batang berbentuk kubus dan daun bertekstur berbentuk segitiga bertumpuk.

Untuk latar belakang, fungsi `drawBackground()` menggambar berbagai bidang dengan tekstur untuk menciptakan suasana pemandangan yang lebih menarik. Program ini juga menangani input pengguna untuk mengubah elemen-elemen tertentu, seperti skala dan posisi boneka salju, serta memungkinkan rotasi menggunakan mouse.

Dengan pustaka `FreeImage`, tekstur dapat dimuat dan diaplikasikan ke berbagai elemen dalam adegan. Kode ini mencakup elemen-elemen yang dioptimalkan untuk menciptakan tampilan yang mendetail, seperti tekstur pada dinding rumah dan daun pohon, serta efek pencahayaan pada bulan dan latar belakang. Secara keseluruhan, program ini menawarkan kombinasi elemen-elemen dasar OpenGL untuk menghasilkan pemandangan 3D interaktif yang estetik.

BAB III

HASIL

3.1 Source Code

```
#include <GL/glew.h>
#include <GL/glut.h>
#include <FreeImage.h>
#include <cmath>
#include <cstdlib>
#include <ctime>

// Variable untuk boneka salju
float bonekaScale = 1.0f; // Skala awal boneka
float bonekaTranslateX = 0.0f; // Posisi awal translasi X boneka
float bonekaTranslateY = 0.0f; // Posisi awal translasi Y boneka

// Variabel untuk rotasi dengan mouse
float xrot = 0.0f;
float yrot = 0.0f;
float xdiff = 0.0f;
float ydiff = 0.0f;
bool mouseDown = false;

void hiddenCarte();
bool hidden = false;

// Variabel ID tekstur
GLuint textureBackgroundID;
GLuint textureBackgroundMoon;
GLuint textureBackgroundDaun;
GLuint textureBackgroundHouse;
GLuint textureBackgroundLangit;
GLuint textureBackgroundTanah;

// Variabel pencahayaan
GLfloat light_position[] = { 1.0f, 1.0f, 1.0f, 0.0f }; // Posisi cahaya
float light_angle = 0.0f; // Sudut rotasi cahaya

// Variabel posisi kamera
float cameraDistance = 8.0f; // Jarak kamera awal dari bulan

//-----Fungsi untuk Pencahayaan-----
void setupLighting()
{
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);

    GLfloat ambientLight[] = { 0.2f, 0.2f, 0.2f, 1.0f };
    GLfloat diffuseLight[] = { 0.8f, 0.8f, 0.8f, 1.0f };
    GLfloat specularLight[] = { 1.0f, 1.0f, 1.0f, 1.0f };

    glLightfv(GL_LIGHT0, GL_AMBIENT, ambientLight);
```

```

    glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuseLight);
    glLightfv(GL_LIGHT0, GL_SPECULAR, specularLight);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);

    glEnable(GL_COLOR_MATERIAL);
    glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);
}

void updateLightPosition() // Memperbarui posisi cahaya()
{
    light_angle += 0.1f; // Kecepatan rotasi cahaya
    if (light_angle > 360.0f)
        light_angle -= 360.0f;

    light_position[0] = 5.0f * cos(light_angle * M_PI / 180.0f);
    light_position[2] = 5.0f * sin(light_angle * M_PI / 180.0f);

    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glutPostRedisplay();
}

void idle() {
    updateLightPosition(); // Perbarui posisi cahaya secara
    kontinu
    glutPostRedisplay(); // Meminta redisplay untuk menggambar
    ulang
}

//-----Fungsi untuk Bulan-----
void drawMoon() {
    glEnable(GL_TEXTURE_2D); // Pastikan tekstur diaktifkan
    glPushMatrix();

    // Gambar bola dengan tekstur
    glTranslatef(7.0f, 4.0f, -5.0f); // Posisi bulan

    // Aktifkan tekstur dan tentukan koordinat tekstur
    glBindTexture(GL_TEXTURE_2D, textureBackgroundMoon);
    glColor3f(1.0f, 1.0f, 1.0f); // Warna putih untuk bulan

    GLUquadric* quad1 = gluNewQuadric();
    gluQuadricTexture(quad1, GL_TRUE);
    gluSphere(quad1, 1.0, 50, 50);
    gluDeleteQuadric(quad1);
    glPopMatrix();
    glDisable(GL_TEXTURE_2D);
}

//-----Fungsi untuk menggambar kubus-----
void drawCube(float x, float y, float z, float size) {
    float half = size / 2.0f;
    glBegin(GL_QUADS);
    // Sisi depan
    glVertex3f(x - half, y - half, z + half);
    glVertex3f(x + half, y - half, z + half);

```

```

glVertex3f(x + half, y + half, z + half);
glVertex3f(x - half, y + half, z + half);

// Sisi belakang
glVertex3f(x - half, y - half, z - half);
glVertex3f(x + half, y - half, z - half);
glVertex3f(x + half, y + half, z - half);
glVertex3f(x - half, y + half, z - half);

// Sisi kiri
glVertex3f(x - half, y - half, z - half);
glVertex3f(x - half, y - half, z + half);
glVertex3f(x - half, y + half, z + half);
glVertex3f(x - half, y + half, z - half);

// Sisi kanan
glVertex3f(x + half, y - half, z - half);
glVertex3f(x + half, y - half, z + half);
glVertex3f(x + half, y + half, z + half);
glVertex3f(x + half, y + half, z - half);

// Sisi atas
glVertex3f(x - half, y + half, z - half);
glVertex3f(x + half, y + half, z - half);
glVertex3f(x + half, y + half, z + half);
glVertex3f(x - half, y + half, z + half);

// Sisi bawah
glVertex3f(x - half, y - half, z - half);
glVertex3f(x + half, y - half, z - half);
glVertex3f(x + half, y - half, z + half);
glVertex3f(x - half, y - half, z + half);
glEnd();
}

//-----Fungsi menggambar rumah-----
void drawHouse() {
    glEnable(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D, textureBackgroundHouse); //
Gunakan tekstur dinding
    glColor3f(1.0f, 1.0f, 1.0f); // warna putih
    // Depan
    glBegin(GL_QUADS);
    glTexCoord2f(0.0f, 0.0f); glVertex3f(-0.5f, -0.75f, 0.5f); //
Kiri bawah
    glTexCoord2f(1.0f, 0.0f); glVertex3f(0.5f, -0.75f, 0.5f); //
Kanan bawah
    glTexCoord2f(1.0f, 1.0f); glVertex3f(0.5f, 0.25f, 0.5f); //
Kanan atas
    glTexCoord2f(0.0f, 1.0f); glVertex3f(-0.5f, 0.25f, 0.5f); //
Kiri atas

    // Belakang
    glTexCoord2f(0.0f, 0.0f); glVertex3f(-0.5f, -0.75f, -0.5f);
    glTexCoord2f(1.0f, 0.0f); glVertex3f(0.5f, -0.75f, -0.5f);
    glTexCoord2f(1.0f, 1.0f); glVertex3f(0.5f, 0.25f, -0.5f);

```

```

glTexCoord2f(0.0f, 1.0f); glVertex3f(-0.5f, 0.25f, -0.5f);

// Samping Kanan
glTexCoord2f(0.0f, 0.0f); glVertex3f(0.5f, -0.75f, -0.5f);
glTexCoord2f(1.0f, 0.0f); glVertex3f(0.5f, -0.75f, 0.5f);
glTexCoord2f(1.0f, 1.0f); glVertex3f(0.5f, 0.25f, 0.5f);
glTexCoord2f(0.0f, 1.0f); glVertex3f(0.5f, 0.25f, -0.5f);

// Samping Kiri
glTexCoord2f(0.0f, 0.0f); glVertex3f(-0.5f, -0.75f, -0.5f);
glTexCoord2f(1.0f, 0.0f); glVertex3f(-0.5f, -0.75f, 0.5f);
glTexCoord2f(1.0f, 1.0f); glVertex3f(-0.5f, 0.25f, 0.5f);
glTexCoord2f(0.0f, 1.0f); glVertex3f(-0.5f, 0.25f, -0.5f);
glEnd();

glDisable(GL_TEXTURE_2D);

// Atap rumah (dinaikkan ke atas)
glColor3f(1.0f, 1.0f, 1.0f);
glBegin(GL_TRIANGLES);
// Segitiga depan
glVertex3f(-0.7f, 0.1f, 0.7f);
glVertex3f(0.7f, 0.1f, 0.7f);
glVertex3f(0.0f, 0.8f, 0.0f);
// Segitiga belakang
glVertex3f(-0.7f, 0.1f, -0.7f);
glVertex3f(0.7f, 0.1f, -0.7f);
glVertex3f(0.0f, 0.8f, 0.0f);
// Segitiga kiri
glVertex3f(-0.7f, 0.1f, -0.7f);
glVertex3f(-0.7f, 0.1f, 0.7f);
glVertex3f(0.0f, 0.8f, 0.0f);
// Segitiga kanan
glVertex3f(0.7f, 0.1f, -0.7f);
glVertex3f(0.7f, 0.1f, 0.7f);
glVertex3f(0.0f, 0.8f, 0.0f);
glEnd();

// Cerobong asap (warna coklat gelap)
glColor3f(0.5f, 0.3f, 0.2f);
glPushMatrix();
glTranslatef(0.3f, 0.45f, -0.3f); // Posisi cerobong lebih
tinggi
drawCube(0.0f, 0.0f, 0.0f, 0.2f); // Perbesar ukuran cerobong
glPopMatrix();

// Salju di atas cerobong asap
glColor3f(1.0f, 1.0f, 1.0f); // Warna putih untuk salju
glPushMatrix();
glTranslatef(0.3f, 0.55f, -0.3f); // Posisi di atas cerobong
glutSolidSphere(0.1, 16, 16); // Bentuk salju sebagai bola
kecil
glPopMatrix();

// Pintu rumah
glDisable(GL_LIGHTING);
glColor3f(0.4f, 0.2f, 0.1f);

```

```

glPushMatrix();
glTranslatef(0.0f, -0.5f, 0.51f);
glScalef(0.2f, 0.4f, 0.01f);
glutSolidCube(1.0);
glPopMatrix();
glEnable(GL_LIGHTING);

// Jendela kiri
glColor3f(0.0f, 0.5f, 1.0f);
glPushMatrix();
glTranslatef(-0.3f, -0.25f, 0.51f);
glScalef(0.2f, 0.2f, 0.01f);
glutSolidCube(1.0);
glPopMatrix();

// Jendela kanan
glColor3f(0.0f, 0.5f, 1.0f);
glPushMatrix();
glTranslatef(0.3f, -0.25f, 0.51f);
glScalef(0.2f, 0.2f, 0.01f);
glutSolidCube(1.0);
glPopMatrix();
}

//-----Fungsi menggambar boneka
salju-----
void drawSnowman(float x, float y, float z) {
    glColor3f(1.0f, 1.0f, 1.0f);
    glPushMatrix();
    glTranslatef(bonekaTranslateX, bonekaTranslateY, 0.0f);
    glScalef(bonekaScale, bonekaScale, bonekaScale);
    glutSolidSphere(0.4, 18, 18);
    glTranslatef(0.0f, 0.5f, 0.0f);
    glutSolidSphere(0.3, 18, 18);
    glTranslatef(0.0f, 0.4f, 0.0f);
    glutSolidSphere(0.2, 18, 18);
    glPopMatrix();
}

//-----Fungsi menggambar pohon-----
void drawTree(float x, float y, float z, float size) {
    float trunkHeight = size * 0.3f; // Tinggi batang
    float trunkWidth = size * 0.2f; // Lebar batang
    float foliageHeight = size * 0.4f; // Tinggi setiap daun
    float foliageBase = size * 0.6f; // Lebar dasar daun

    // Gambar batang pohon
    glColor3f(0.55f, 0.27f, 0.07f); // Warna coklat untuk batang
    glPushMatrix();
    glTranslatef(x, y + trunkHeight / 2.0f, z);
    glScalef(trunkWidth, trunkHeight, trunkWidth);
    glutSolidCube(1.0);
    glPopMatrix();

    // Gambar daun pohon (5 segitiga bertumpuk dalam 3D)
    glEnable(GL_TEXTURE_2D);

```

```

        glBindTexture(GL_TEXTURE_2D, textureBackgroundDaun);
        glColor3f(1.0f, 1.0f, 1.0f); // Warna putih untuk mengaktifkan
        tekstur
        for (int i = 0; i < 5; ++i) {
            float offsetY = trunkHeight + i * foliageHeight * 0.5f;
            float currentBase = foliageBase * (1.0f - 0.2f * i);
            glPushMatrix();
            glTranslatef(x, y + offsetY, z);

            glBegin(GL_TRIANGLES);
            // Segitiga depan
            glTexCoord2f(0.0f, 0.0f); glVertex3f(-currentBase, 0.0f,
currentBase);
            glTexCoord2f(1.0f, 0.0f); glVertex3f(currentBase, 0.0f,
currentBase);
            glTexCoord2f(0.5f, 1.0f); glVertex3f(0.0f, foliageHeight,
0.0f);

            // Segitiga belakang
            glTexCoord2f(0.0f, 0.0f); glVertex3f(-currentBase, 0.0f, -
currentBase);
            glTexCoord2f(1.0f, 0.0f); glVertex3f(currentBase, 0.0f, -
currentBase);
            glTexCoord2f(0.5f, 1.0f); glVertex3f(0.0f, foliageHeight,
0.0f);

            // Segitiga kiri
            glTexCoord2f(0.0f, 0.0f); glVertex3f(-currentBase, 0.0f, -
currentBase);
            glTexCoord2f(1.0f, 0.0f); glVertex3f(-currentBase, 0.0f,
currentBase);
            glTexCoord2f(0.5f, 1.0f); glVertex3f(0.0f, foliageHeight,
0.0f);

            // Segitiga kanan
            glTexCoord2f(0.0f, 0.0f); glVertex3f(currentBase, 0.0f, -
currentBase);
            glTexCoord2f(1.0f, 0.0f); glVertex3f(currentBase, 0.0f,
currentBase);
            glTexCoord2f(0.5f, 1.0f); glVertex3f(0.0f, foliageHeight,
0.0f);
            glEnd();

            glPopMatrix();
        }
        glDisable(GL_TEXTURE_2D);
    }

//-----Fungsi untuk
Background-----
void drawBackground() {
    glEnable(GL_TEXTURE_2D);

    // Background belakang
    glBindTexture(GL_TEXTURE_2D, textureBackgroundID);
    glColor3f(1.0f, 1.0f, 1.0f); // Warna putih
    glBegin(GL_QUADS);

```

```

    glTexCoord2f(0.0f, 0.0f); glVertex3f(-13.5f, -5.0f, -10.0f);
// Kiri bawah
    glTexCoord2f(1.0f, 0.0f); glVertex3f(13.5f, -5.0f, -10.0f);
// Kanan bawah
    glTexCoord2f(1.0f, 1.0f); glVertex3f(13.5f, 8.0f, -10.0f);
// Kanan atas
    glTexCoord2f(0.0f, 1.0f); glVertex3f(-13.5f, 8.0f, -10.0f);
// Kiri atas
    glEnd();

    // Background depan
    glBindTexture(GL_TEXTURE_2D, textureBackgroundID);
    glBegin(GL_QUADS);
    glTexCoord2f(0.0f, 0.0f); glVertex3f(-13.5f, -5.0f, 10.0f); //
Kiri bawah
    glTexCoord2f(1.0f, 0.0f); glVertex3f(13.5f, -5.0f, 10.0f); //
Kanan bawah
    glTexCoord2f(1.0f, 1.0f); glVertex3f(13.5f, 8.0f, 10.0f); //
Kanan atas
    glTexCoord2f(0.0f, 1.0f); glVertex3f(-13.5f, 8.0f, 10.0f); //
Kiri atas
    glEnd();

    // Background kiri
    glBindTexture(GL_TEXTURE_2D, textureBackgroundID);
    glBegin(GL_QUADS);
    glTexCoord2f(0.0f, 0.0f); glVertex3f(-13.5f, -5.0f, -10.0f);
// Kiri bawah
    glTexCoord2f(1.0f, 0.0f); glVertex3f(-13.5f, -5.0f, 10.0f);
// Kanan bawah
    glTexCoord2f(1.0f, 1.0f); glVertex3f(-13.5f, 8.0f, 10.0f);
// Kanan atas
    glTexCoord2f(0.0f, 1.0f); glVertex3f(-13.5f, 8.0f, -10.0f);
// Kiri atas
    glEnd();

    // Background kanan
    glBindTexture(GL_TEXTURE_2D, textureBackgroundID);
    glBegin(GL_QUADS);
    glTexCoord2f(0.0f, 0.0f); glVertex3f(13.5f, -5.0f, -10.0f); //
Kiri bawah
    glTexCoord2f(1.0f, 0.0f); glVertex3f(13.5f, -5.0f, 10.0f); //
Kanan bawah
    glTexCoord2f(1.0f, 1.0f); glVertex3f(13.5f, 8.0f, 10.0f); //
Kanan atas
    glTexCoord2f(0.0f, 1.0f); glVertex3f(13.5f, 8.0f, -10.0f); //
Kiri atas
    glEnd();

    // Background atas
    glBindTexture(GL_TEXTURE_2D, textureBackgroundLangit);
    glBegin(GL_QUADS);
    glTexCoord2f(0.0f, 0.0f); glVertex3f(-13.5f, 8.0f, -10.0f); //
Kiri belakang
    glTexCoord2f(1.0f, 0.0f); glVertex3f(13.5f, 8.0f, -10.0f); //
Kanan belakang
    glTexCoord2f(1.0f, 1.0f); glVertex3f(13.5f, 8.0f, 10.0f); //

```

```

Kanan depan
    glTexCoord2f(0.0f, 1.0f); glVertex3f(-13.5f, 8.0f, 10.0f); //
Kiri depan
    glEnd();

    // Background alas (tanah)
    glEnable(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D, textureBackgroundTanah);
    glColor3f(1.0f, 1.0f, 1.0f); // Warna putih untuk tanah
    glBegin(GL_QUADS);
    glTexCoord2f(0.0f, 0.0f); glVertex3f(-13.5f, -5.0f, 10.0f);
    glTexCoord2f(1.0f, 0.0f); glVertex3f(13.5f, -5.0f, 10.0f);
    glTexCoord2f(1.0f, 1.0f); glVertex3f(13.5f, -5.0f, -10.0f);
    glTexCoord2f(0.0f, 1.0f); glVertex3f(-13.5f, -5.0f, -10.0f);
    glEnd();

    glDisable(GL_TEXTURE_2D);
}

//-----Fungsi untuk
Kartesius-----
void drawCartesius()
{
    glColor3f(0.0, 0.0, 0.0);
    glBegin(GL_LINES);
    // x Line
    glVertex3f(-10.0, 0.0, 0.0);
    glVertex3f(10.0, 0.0, 0.0);

    // y Line
    glVertex3f(0.0, -10.0, 0.0);
    glVertex3f(0.0, 10.0, 0.0);

    // z Line
    glVertex3f(0.0, 0.0, -10.0);
    glVertex3f(0.0, 0.0, 10.0);
    glEnd();
}

//-----Fungsi untuk memuat
tekstur-----
GLuint loadTexture(const char* path) {
    GLuint textureID;
    glGenTextures(1, &textureID);

    FIBITMAP* bitmap = FreeImage_Load(FreeImage_GetFileType(path,
0), path);
    FIBITMAP* pImage = FreeImage_ConvertTo24Bits(bitmap);
    int width = FreeImage_GetWidth(pImage);
    int height = FreeImage_GetHeight(pImage);
    void* data = FreeImage_GetBits(pImage);

    glBindTexture(GL_TEXTURE_2D, textureID);
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0,
GL_BGR, GL_UNSIGNED_BYTE, data);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR);
}

```



```

    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
GL_LINEAR);

    FreeImage_Unload(pImage);
    FreeImage_Unload(bitmap);
    return textureID;
}

void display() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    // Posisi kamera tetap berdasarkan cameraDistance
    gluLookAt(0, 0, cameraDistance, 0, 0, 0, 0, 1, 0);
    // Perputaran objek otomatis
    glRotatef(xrot, 1, 0, 0);
    glRotatef(yrot, 0, 1, 0); // Perputaran berkelanjutan pada
sumbu Y
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // Warna latar belakang

    hiddenCarte();
    drawBackground(); // Gambar latar belakang

    // Tambahkan bulan dengan pencahayaan
    glPushMatrix();
    setupLighting();
    drawMoon();
    glPopMatrix();

    // Gambar rumah
    glPushMatrix();
    glTranslatef(-4.0f, -2.3f, -6.0f);
    glScalef(3.5f, 3.5f, 3.5f); // Ukuran rumah diperbesar
    drawHouse();
    glPopMatrix();

    // Gambar boneka salju
    glPushMatrix();
    glTranslatef(0.0f, -4.6f, -6.0f);
    drawSnowman(0.0f, 0.0f, 0.0f);
    glPopMatrix();

    // Gambar pohon
    glPushMatrix();
    drawTree(4.0f, -5.0f, -7.0f, 3.5f);
    glPopMatrix();

    glutSwapBuffers();
}

//-----Fungsi reshape-----
void reshape(int w, int h) {
    if (h == 0) h = 1;
    float aspect = (float)w / (float)h;
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

```

```

    gluPerspective(45.0, aspect, 1.0, 200.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

void keyboard(unsigned char key, int x, int y)
{
    switch (key)
    {
        case 27: // ESC untuk keluar
            exit(0);
            break;
        case 'r': // Tambahan untuk Zoom in
            cameraDistance -= 0.5f;
            if (cameraDistance < 2.0f) cameraDistance = 2.0f; //
Jangan terlalu dekat
            break;
        case 't': // Tambahan untuk Zoom out
            cameraDistance += 0.5f;
            break;
        case 'v': // Memperbesar skala boneka
            bonekaScale += 0.1f;
            break;
        case 'b': // Memperkecil skala boneka
            if (bonekaScale > 0.1f)
                bonekaScale -= 0.1f;
            break;
        case 'w': // Pindah boneka ke atas
            bonekaTranslateY += 0.1f;
            break;
        case 's': // Pindah boneka ke bawah
            bonekaTranslateY -= 0.1f;
            break;
        case 'a': // Pindah boneka ke kiri
            bonekaTranslateX -= 0.1f;
            break;
        case 'd': // Pindah boneka ke kanan
            bonekaTranslateX += 0.1f;
            break;
        case 'c':
            hidden = !hidden;
            break;
    }
    // Update posisi cahaya di OpenGL
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glutPostRedisplay();
}

void hiddenCarte()
{
    if (hidden)
    {
        drawCartesius();
    }
}

// Fungsi untuk input mouse

```

```

void mouse(int button, int state, int x, int y) {
    if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
        mouseDown = true;
        xdiff = x - yrot;
        ydiff = -y + xrot;
    } else {
        mouseDown = false;
    }
}

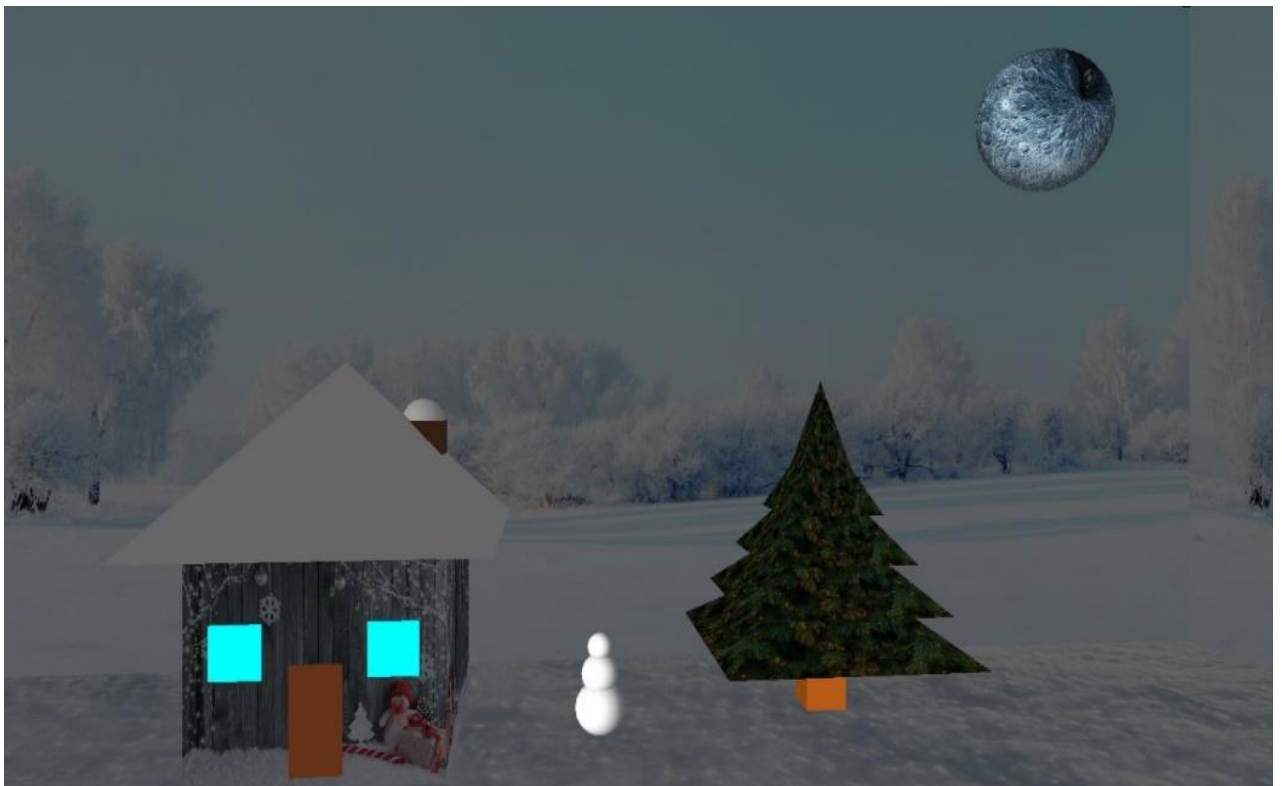
void mouseMotion(int x, int y) {
    if (mouseDown) {
        yrot = x - xdiff;
        xrot = y + ydiff;
        glutPostRedisplay();
    }
}

//-----Fungsi inisialisasi-----
void init() {
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_TEXTURE_2D);
    textureBackgroundID = loadTexture("salju.jpg");
    textureBackgroundMoon = loadTexture("moon.jpg");
    textureBackgroundDaun = loadTexture("christmastree.jpg");
    textureBackgroundHouse = loadTexture("BadanRumah.jpg");
    textureBackgroundLangit = loadTexture("langit.jpg");
    textureBackgroundTanah = loadTexture("TanahSalju.jpg");
}

// Fungsi utama
int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(800, 600);
    glutCreateWindow("Rumah di Musim Salju");
    glutFullScreen(); // tampilan fullscreen
    glewInit();
    init();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMouseFunc(mouse);
    glutMotionFunc(mouseMotion);
    glutKeyboardFunc(keyboard); // Daftarkan fungsi keyboard
    glutIdleFunc(idle); // Mengatur fungsi idle
    glutMainLoop();
    return 0;
}

```

3.2 Output



3.3 Penjelasan

penjelasan tentang bagian-bagian utama dari kode

1. Inisialisasi Variabel dan Library

- GL/glew.h dan GL/glut.h adalah library untuk menggunakan OpenGL dalam membuat aplikasi grafis 3D.
- FreeImage.h adalah library untuk menangani format gambar.
- cmath, cstdlib, dan ctime adalah library C++ standar untuk operasi matematika, fungsi acak, dan pengaturan waktu.

2. Variabel Boneka Salju

- bonekaScale, bonekaTranslateX, dan bonekaTranslateY mengatur skala dan posisi boneka salju dalam ruang 3D.
- bonekaScale digunakan untuk mengubah ukuran objek boneka.
- bonekaTranslateX dan bonekaTranslateY digunakan untuk menggeser posisi boneka pada sumbu X dan Y.

3. Rotasi dengan Mouse

- Variabel xrot, yrot, xdiff, ydiff, dan mouseDown mengontrol rotasi objek berdasarkan input dari mouse. Variabel-variabel ini digunakan untuk memanipulasi sudut rotasi objek.

4. Pencahayaan

- Fungsi setupLighting() digunakan untuk mengonfigurasi pencahayaan dalam ruang 3D. Posisi dan intensitas cahaya diatur di sini.
- Variabel light_position menentukan posisi cahaya di ruang 3D, sementara light_angle digunakan untuk memutar posisi cahaya.
- Fungsi updateLightPosition() memperbarui posisi cahaya secara dinamis berdasarkan rotasi cahaya.

5. Fungsi untuk Menggambar Objek

- **Bulan (drawMoon):** Fungsi ini menggambar bulan dalam bentuk bola 3D dengan tekstur menggunakan gluSphere(). Tekstur untuk bulan diambil dari textureBackgroundMoon.
- **Kubus (drawCube):** Fungsi ini menggambar sebuah kubus di 3D. Fungsi ini menerima parameter x, y, z, dan size untuk menentukan posisi dan ukuran kubus. Setiap sisi kubus digambar dengan menggunakan koordinat vertikal dan tekstur.
- **Rumah (drawHouse):** Fungsi ini menggambar sebuah rumah kecil menggunakan beberapa sisi kotak dengan tekstur dinding. Fungsi ini menggunakan glBegin(GL_QUADS) dan glEnd() untuk menggambar sisi rumah menggunakan koordinat vertikal dan tekstur yang telah ditentukan.

6. Tekstur

- Kode ini memanfaatkan tekstur untuk memberikan detail pada objek 3D. Fungsi `glBindTexture()` digunakan untuk mengikat tekstur pada objek-objek tertentu (misalnya bulan atau rumah).
- Tekstur yang digunakan termasuk `textureBackgroundMoon`, `textureBackgroundHouse`, dan lainnya.

7. Rotasi Objek

- Untuk memberikan efek dinamis pada objek, terutama pencahayaan, objek seperti bulan dan rumah dapat diputar. Fungsi-fungsi rotasi ini menggunakan variabel sudut rotasi untuk memutar objek di dalam ruang 3D.

8. Main Loop

- Fungsi `idle()` adalah fungsi utama yang secara berkala memperbarui posisi cahaya dan menggambar ulang scene, agar objek bergerak atau berubah berdasarkan input pengguna.

9. Proses Penggambaran

- Penggambaran objek dilakukan dalam loop render dengan GLUT (`glutPostRedisplay()`) untuk memastikan objek-objek seperti bulan dan rumah terus-menerus digambar ulang dengan setiap frame.

Penjelasan Fungsi Utama:

- **`setupLighting()`**: Menyusun pencahayaan untuk scene 3D.
- **`drawMoon()`**: Menggambar objek bulan menggunakan tekstur dan fungsi `gluSphere()`.
- **`drawCube()`**: Menggambar objek kubus dengan sisi-sisi yang terdefinisi menggunakan koordinat 3D.
- **`drawHouse()`**: Menggambar objek rumah menggunakan teknik tekstur dan koordinat 3D.
- **`updateLightPosition()`**: Mengubah posisi cahaya dalam ruang 3D.
- **`idle()`**: Fungsi yang dipanggil berulang kali untuk memperbarui objek-objek dan memberi efek gerakan.

Kelebihan dan Potensi Pengembangan:

- **Interaktivitas**: Kode ini memungkinkan pengaturan pencahayaan dan posisi objek dengan cara yang dinamis (misalnya, rotasi cahaya, pergerakan objek).
- **Tekstur**: Penggunaan tekstur pada objek membuat hasil visual lebih realistis.
- **Fleksibilitas**: Fungsi-fungsi seperti `drawCube()` dan `drawHouse()` bisa dikembangkan lebih lanjut untuk menggambar objek-objek lebih kompleks.

Output yang dihasilkan oleh kode tersebut adalah tampilan grafis 3D yang interaktif, di mana objek-objek seperti rumah dan bulan muncul dalam scene dengan rotasi dinamis. Rumah yang berbentuk kubus dan bulan yang berbentuk bola berputar secara terus-menerus, dengan gerakan yang dipengaruhi oleh input mouse pengguna, yang memungkinkan objek-objek tersebut berputar mengikuti gerakan mouse. Selain itu, pencahayaan dinamis ditambahkan untuk memberikan efek bayangan dan cahaya yang bervariasi di sekitar objek, menciptakan tampilan yang lebih realistis. Cahaya tersebut dapat bergerak sesuai dengan pergerakan kamera, yang memberikan efek perubahan pencahayaan saat objek-objek bergerak. Fungsi `idle()` memastikan pembaruan terus-menerus setiap frame, menjaga agar rotasi objek dan pergerakan pencahayaan tetap berlangsung tanpa henti, menciptakan animasi yang halus. Secara keseluruhan, output ini menciptakan pengalaman visual 3D yang interaktif, di mana pengguna dapat mengubah sudut pandang dan melihat objek dengan pencahayaan yang dinamis, memberikan kesan seolah-olah objek tersebut berputar dan bergerak di ruang 3D.

Output yang dihasilkan oleh kode tersebut dipengaruhi oleh beberapa faktor, seperti pencahayaan, rotasi objek, tekstur, dan pembaruan frame. Pencahayaan yang dinamis, yang diatur melalui pergerakan cahaya dalam ruang 3D, memberikan efek kedalaman pada objek dan menambahkan dimensi visual. Hal ini memungkinkan objek, seperti bulan dan rumah, untuk terlihat lebih realistis dengan cahaya yang berbeda di setiap sudutnya. Rotasi objek, yang dipengaruhi oleh input mouse, juga memberikan kesan dinamis, di mana objek tampak berputar dan berubah perspektif seiring dengan pergerakan kamera. Penggunaan tekstur pada objek, seperti pada bulan dan rumah, memperkaya detail visual, membuat objek tampak lebih nyata dan tidak datar. Pembaruan frame yang dilakukan dalam fungsi `idle()` memungkinkan objek bergerak secara dinamis, memperbarui rotasi dan posisi cahaya secara berkelanjutan. Semua elemen ini bekerja bersama-sama untuk menciptakan efek visual yang menarik, dengan objek yang terus bergerak, berubah pencahayaannya, dan bereaksi terhadap input pengguna, memberikan pengalaman tampilan 3D yang lebih hidup dan dinamis.

BAB IV

4.1. Kesimpulan

Program kami ini adalah simulasi grafis berbasis OpenGL yang menggabungkan elemen tiga dimensi dengan berbagai fitur, seperti tekstur, pencahayaan dinamis, dan interaksi pengguna. Simulasi memiliki semua objek, seperti pohon, rumah, boneka salju, bulan, dan lainnya, yang dibangun secara modular menggunakan fungsi terpisah. Karena itu, sangat mudah untuk mengembangkan lebih lanjut, karena setiap objek dapat diubah atau diperbarui tanpa mempengaruhi bagian lain dari kode.

Tampilan yang lebih realistis diperoleh melalui penerapan tekstur pada berbagai objek. Daun pohon, tekstur dinding rumah, dan efek langit yang memanfaatkan gambar latar belakang yang cocok adalah contohnya. Dengan menggunakan tekstur ini, pengalaman grafis menjadi lebih baik. Pencahayaan dinamis juga digunakan untuk membuat efek bayangan yang lebih realistis dan membuat suasana lebih hidup. Cahaya berputar di sekitar scene dengan menggunakan `GL_LIGHTING`, memberikan kesan pencahayaan yang bergerak dan meningkatkan kesan interaktivitas.

Selain itu, program ini memungkinkan pengguna berinteraksi melalui kontrol kamera yang dapat diputar menggunakan mouse untuk memberikan sudut pandang yang berbeda pada objek. Dengan cara ini, pengguna dapat meningkatkan pengalaman visual dengan mengubah secara interaktif skala dan posisi objek.

Program ini dapat berkembang menjadi aplikasi grafis yang lebih kompleks, seperti game sederhana, visualisasi 3D untuk edukasi, atau demo grafis yang lebih interaktif dan mendalam. Ini dapat dilakukan dengan menambahkan animasi, efek suara, atau elemen interaktif lainnya.

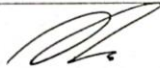


DAFTAR PUSTAKA

1. Isi, D. (2016). ii, *I*, 1–40.
2. Muhammad Adnani, & Achmad Zakki Falani. (2021). Implementasi Open Gl Untuk Pembuatan Objek 3d. *Journal Zetroem*, 3(1), 1–6.
<https://doi.org/10.36526/ztr.v3i1.1249>
3. Priyantono, M. B., & Rachmawan, A. A. (2020). Implementasi Sistem Simulasi Penampilan Tata Surya Berbasis 3D Menggunakan Opengl. *Jurnal Teknologi Informasi*, 4(1), 91–95. <https://doi.org/10.36294/jurti.v4i1.1231>

LAMPIRAN

ASISTENSI TUGAS PRAKTIKUM GRAFIK KOMPUTER

Nama Pembimbing : M. Miskun Amhuslam
Kelompok & Kelas : Kelompok 8, kelas B
Anggota Kelompok : - Restu Bagja Maulud
- Tsani hisni Amala
- Faren Anjani
Tema : Rumah musim Salju

No	Tanggal	Uraian Asistensi	Tanda Tangan
1	16/12-2024	Konsep awal	
2	23/12-2024	Texture, Pencahayaan	
3	6/1-2025	Background belakang Balas	
4			
5			

Instruktur Praktikum



Arul Budi Kalimat, S.Kom