LAPORAN PRATIKUM GRAFIK KOMPUTER

Diajukan untuk memenuhi Tugas mata kuliah Pratikum Grafik Komputer

STADION BOLA

Dosen Pengampu : Sri Rahayu, M.Kom

Instruktur Pratikum : Arul Budi Kalimat, S.Kom

Disusun oleh

Kelompok 7 :

Rifki Firmansyah
2306067

Zaki Muhammad
2306094

Mohammad Yasrul
2306097

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN ILMU KOMPUTER

INSTITUT TEKNOLOGI GARUT

2024

KATA PENGANTAR

Puji dan syukur kehadirat Allah SWT atas segala rahmat dan karunia-Nya sehingga kami dapat menyelesaikan Laporan Praktikum Jaringan Komputer ini. Laporan ini dibuat sebagai salah satu tugas dari mata kuliah Jaringan Komputer, dengan tujuan untuk memberikan pemahaman yang lebih baik tentang Pembuatan Lapangan Bola.

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,14 Desember 2024

Kelompok 7

DAFTAR ISI

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Grafik komputer merupakan salah satu cabang ilmu komputer yang mempelajari teknik-teknik untuk menghasilkan, memanipulasi, dan merepresentasikan gambar atau objek secara visual menggunakan komputer. Dalam perkembangan teknologi, grafik komputer memiliki peran penting dalam berbagai bidang seperti hiburan, pendidikan, arsitektur, simulasi, dan pengembangan perangkat lunak. Salah satu implementasi dasar dari grafik komputer adalah pembuatan objek 3D yang dapat divisualisasikan secara interaktif. (Deddy Suhardiman, G. Kaunang, Rizal Sengkey, 2012)

Sebagai salah satu bentuk implementasi, pembuatan objek lapangan bola menggunakan OpenGL memberikan gambaran tentang bagaimana konsep dasar grafik komputer seperti koordinat ruang, transformasi geometris, dan pewarnaan diterapkan dalam membangun sebuah simulasi visual. Dalam konteks ini, objek lapangan bola dipilih karena memiliki karakteristik yang cukup sederhana namun tetap memuat berbagai aspek fundamental grafik komputer, seperti pembuatan objek geometri persegi panjang, pemberian tekstur, dan manipulasi kamera. (Dwi Agus Kurniawan, Muhammad Iqbal, 2021)

Unity Hub adalah platform manajemen proyek yang mempermudah pengelolaan pengembangan aplikasi berbasis Unity Engine. Dalam konteks visualisasi titik koordinat, Unity Hub menyediakan akses ke alat-alat yang memungkinkan pengembang memvisualisasikan data koordinat secara akurat dalam ruang tiga dimensi. Untuk aplikasi seperti stadion sepak bola, titik-titik koordinat dapat digunakan untuk merepresentasikan posisi elemen penting, seperti garis lapangan, posisi pemain, atau objek lainnya. Dengan memanfaatkan kemampuan Unity dalam merender grafik real-time, pengembang dapat mengimpor data koordinat dari sumber eksternal, memetakan titik-titik tersebut ke dalam ruang 3D, dan menghasilkan visualisasi yang interaktif dan informatif. Unity Hub menjadi sarana penting untuk memulai dan mengelola proyek ini, memastikan kompatibilitas versi Unity dan memudahkan integrasi alat pengembangan tambahan.

Dalam era digital yang semakin berkembang pesat, kolaborasi menjadi elemen penting dalam pengembangan perangkat lunak. GitHub, sebagai platform manajemen versi berbasis Git, telah menjadi salah satu alat utama yang digunakan untuk mendukung kolaborasi dalam pengembangan perangkat lunak secara global.

GitHub menyediakan berbagai fitur yang memungkinkan tim bekerja secara efisien, seperti kontrol versi, sistem pull request untuk mereview perubahan kode, serta kemampuan untuk melacak isu dan bug. Selain itu, platform ini juga mendukung kolaborasi lintas disiplin, memfasilitasi komunikasi antara pengembang, desainer, dan manajer proyek.

Dengan integrasi fitur-fitur seperti Continuous Integration/Continuous Deployment (CI/CD) dan kemampuan untuk berinteraksi melalui dokumentasi dan Wiki, GitHub memungkinkan tim untuk menjaga kualitas kode, mempercepat pengembangan, dan mendukung kerja tim yang terdesentralisasi. Oleh karena itu, pemahaman dan penggunaan GitHub secara optimal sangat penting untuk memastikan keberhasilan kolaborasi dalam proyek pengembangan perangkat lunak, terutama dalam lingkungan kerja yang dinamis dan berbasis remote.

.

1.2 Rumusan Masalah

1. Apa itu Grafis Komputer?
2. Apa itu OpenGL?
3. Bagaimana cara kerja dari OpenGL ?
4. Apa itu Unity Hub?
5. Bagaimana cara kerja Unity Hub?
6. Bagaimana membuat Visualisasi Stadion Bola dengan Unity Hub?
7. Bagaimana membuat Objek Stadion Bola dalam OpenGL?
8. Apa itu github ?
9. Bagaimana cara kerja github?

1.3 Tujuan

1. Mengethaui apa itu Grafis Komputer.
2. Mengetahui apa itu OpenGL.
3. Mengetahui apa itu Unity Hub.
4. Mengetahui cara kerja dari OpenGL.
5. Mengetahhui cara kerja dari Unity Hub.
6. Mengetahui cara pembuatan Objek Stadion Bola dalam Unity Hub.
7. Mengetahui cara pembuatan Objek Stadion Bola dalam OpenGL.
8. Mengetahui apa itu github
9. Mengetahui cara kerja github

BAB II
TINJAUAN PUSTAKA

2.1 Grafis Komputer

Grafis komputer adalah bidang ilmu yang mempelajari pembuatan, manipulasi, dan representasi visual data menggunakan komputer. Bidang ini mencakup berbagai teknik dan metode untuk menghasilkan gambar dan animasi, baik dalam dua dimensi (2D) maupun tiga dimensi (3D). (Universitas Sains dan Teknologi Komputer (STEKOM))

Perkembangan grafis komputer dimulai pada tahun 1960-an dengan pembuatan gambar sederhana menggunakan plotter. Seiring kemajuan teknologi, grafis komputer berkembang pesat, memungkinkan pembuatan visual yang lebih kompleks dan realistis. Saat ini, grafis komputer digunakan dalam berbagai industri, termasuk film, permainan video, desain arsitektur, dan simulasi ilmiah.

2.2 Open GL

OpenGL (Open Graphics Library) adalah salah satu pustaka grafik yang paling banyak digunakan untuk pengembangan aplikasi berbasis grafik. OpenGL menyediakan berbagai fungsi untuk menggambar objek 2D dan 3D, manipulasi transformasi, pengaturan pencahayaan, serta efek visual lainnya. Pengenalan dan pemahaman terhadap OpenGL menjadi langkah awal yang penting dalam memahami dasar-dasar grafik komputer. (Muhammad Adnani, 2021)

2.3 Cara Kerja OpenGL

OpenGL (Open Graphics Library) adalah API grafis lintas platform yang digunakan untuk membuat grafik 2D dan 3D. Cara kerja OpenGL dapat dijelaskan melalui beberapa langkah utama, dari input data hingga rendering akhir pada layar.

1. Inisialisasi

   OpenGL memerlukan konteks grafik, yang disiapkan oleh sistem jendela atau pustaka seperti GLFW atau SDL. Konteks ini adalah lingkungan yang memungkinkan komunikasi antara aplikasi dan perangkat keras grafis (GPU).
2. Input

   OpenGL bekerja dengan data geometris (seperti titik, garis, dan poligon). Data ini dikirimkan ke OpenGL melalui buffer. **Vertex Buffer Objects (VBOs)**: Menyimpan data vertex (koordinat, warna, normal, dll.). **Element Buffer Objects (EBOs)**: Menyimpan indeks yang menentukan bagaimana vertex digabungkan menjadi primitif grafis (segitiga, garis, dll.).
3. Shader dan PipeLine

   Shader adalah program kecil yang berjalan di GPU untuk memproses data grafis. OpenGL menggunakan pipeline grafis berbasis shader yang terdiri dari beberapa tahap.
4. Resterasi

   Data geometris yang telah diproses shader diubah menjadi fragmen melalui proses rasterisasi. Fragmen adalah kandidat piksel yang akan ditampilkan di layar.

5. Pengujian dan Blending

   Pada tahap pengujian dan penggabungan warna (blending), OpenGL menjalankan beberapa tes untuk memastikan fragmen yang dihasilkan layak ditampilkan. Tes kedalaman (**depth test**) memastikan hanya fragmen yang berada paling dekat dengan kamera yang akan ditampilkan, sehingga objek yang lebih jauh tidak menimpa objek di depan. Selain itu, tes stencil (**stencil test**) dapat digunakan untuk membatasi area tertentu dari layar tempat fragmen diizinkan untuk dirender. Setelah tes ini selesai, OpenGL menerapkan operasi penggabungan warna (**blending**) untuk menciptakan efek visual seperti transparansi dengan menggabungkan warna fragmen baru dengan warna yang sudah ada di framebuffer.

6. Rendering ke Layar

   Tahap akhir adalah rendering hasil ke layar. Setelah semua fragmen melewati pengujian dan operasi lainnya, OpenGL mengirimkan data grafis ke framebuffer. Framebuffer ini kemudian diteruskan ke layar untuk ditampilkan kepada pengguna. Untuk memastikan rendering yang mulus dan menghindari flickering, OpenGL biasanya menggunakan teknik **double buffering**. Dalam teknik ini, satu buffer digunakan untuk menggambar frame berikutnya, sementara buffer lainnya menampilkan frame sebelumnya. Setelah proses menggambar selesai, buffer tersebut ditukar, memastikan tampilan grafis yang halus dan bebas gangguan.

2.4 Unity Hub

Unity Hub adalah alat bantu yang dikembangkan oleh Unity Technologies untuk mempermudah pengelolaan berbagai versi Unity Editor, proyek game, aset, serta akun pengguna. Dengan Unity Hub, pengguna dapat:

1. Mengelola berbagai versi Unity Editor tanpa harus menginstal ulang.
2. Membuat dan mengelola proyek Unity dengan berbagai pengaturan.
3. Mengunduh paket dan add-ons sesuai kebutuhan pengembangan.
4. Mengakses tutorial dan dokumentasi Unity secara langsung.

2.5 Cara Kerja Unity Hub

Unity Hub bekerja sebagai pengelola proyek dan versi Unity Editor, mempermudah pengembang dalam mengelola berbagai proyek game atau aplikasi berbasis Unity.

Instalasi & Manajemen Unity Editor

1. Unity Hub memungkinkan pengguna untuk menginstal, memperbarui, dan menghapus berbagai versi Unity Editor. Ini berguna jika proyek memerlukan versi Unity yang berbeda.
2. Pembuatan & Pengelolaan Proyek

   Pengguna dapat membuat proyek baru dengan memilih versi Unity, template, dan lokasi penyimpanan.

   Semua proyek yang sudah ada dapat diakses langsung dari Unity Hub.
3. Manajemen Akun & Lisensi

   Pengguna harus masuk dengan akun Unity untuk mengakses fitur Unity dan mengelola lisensi (Personal, Plus, atau Pro).
4. Instalasi Modul Tambahan

   Unity Hub memungkinkan instalasi paket tambahan seperti Android Build

Support, WebGL, dan iOS Support untuk pengembangan lintas platform.

5. Akses Tutorial & Dokumentasi

   Unity Hub menyediakan akses langsung ke dokumentasi, tutorial, dan Unity Learn untuk membantu pengguna memahami Unity lebih lanjut.

2.6 GitHub

GitHub merupakan platform berbasis web yang berfungsi sebagai alat kolaborasi dan pengelolaan kode sumber dengan menggunakan sistem kontrol versi Git. Platform ini menyediakan berbagai fitur seperti repositori untuk menyimpan proyek, branching untuk pengembangan fitur baru, dan merging untuk mengintegrasikan perubahan ke dalam kode utama. Selain itu, GitHub mendukung kolaborasi antar anggota tim melalui fitur seperti pull request dan issue tracker. Dengan menggunakan GitHub, pengembang dapat bekerja secara efisien, memantau perubahan kode, serta memastikan integritas dan stabilitas proyek perangkat lunak yang dikelola.

2.7 Cara Kerja GitHub

GitHub adalah platform yang menggunakan sistem kontrol versi Git untuk mengelola dan berbagi proyek perangkat lunak. Dengan GitHub, pengembang dapat berkolaborasi secara efisien, melacak perubahan kode, dan mengelola versi yang berbeda dari proyek. GitHub menyediakan berbagai fitur seperti repositori untuk menyimpan kode, serta alat untuk mengelola perubahan melalui commit, push, dan branching. Berikut adalah cara kerja GitHub dalam mengelola proyek perangkat lunak.

1. Pembuatan Repositori:

   Pengguna membuat repositori di GitHub yang berfungsi sebagai tempat penyimpanan kode sumber.

2. Clone Repositori:

   Pengguna meng-clone repositori ke komputer lokal untuk mulai bekerja dengan proyek tersebut. Proses ini dilakukan dengan perintah `git clone <URL-repositori>`.

3. Melakukan Perubahan:

   Pengguna mengedit atau menambah file di repositori lokal sesuai dengan kebutuhan pengembangan.

4. Commit Perubahan:

   Setelah melakukan perubahan, pengguna menyimpan perubahan tersebut secara lokal dengan perintah `git commit -m "deskripsi perubahan"`.

5. Push ke GitHub:

   Perubahan yang sudah di-commit dikirim kembali ke repositori GitHub dengan perintah `git push`.

6. Branching:

   Pengguna dapat membuat branch untuk mengembangkan fitur baru atau memperbaiki bug tanpa memengaruhi kode utama. Proses ini dilakukan dengan perintah git branch `<nama-branch>`.

7. Merging:

8. Setelah fitur atau perbaikan selesai, branch tersebut digabungkan kembali ke cabang utama (main branch) dengan perintah `git merge <nama-branch>`.

2.8 Pembuatan Visualisasi Stadion Bola Dengan Unity Hub

1. Buat Proyek Baru
2. Pilih template 3D dan beri nama proyek.
3. Membuat Lapangan Bola
4. Buat Cube dan ubah skala
5. Membuat Tribun

2.9 Pembuatan Stadion Bola Dengan OpenGL

1. Membuat Folder TUGAS_BESAR
2. Membuat File main.exe
3. Membuat struktuk 3D Open GL
4. Membuat Lahan dengan cube
5. Membuat Lapang dengan cube
6. Membuat Tribun dengan cube
7. Membuat Banch dengan cube
8. Membuat Garis dengan Vertex
9. Membuat Matahari dan awang menggunakan spehere

BAB III
HASIL

3.1 Source Code

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

bool showAxes = true;

float bolaRotasi = 0.0f;
float bolaPosisiX = 0.0f;
float bolaPosisiZ = 0.0f;

float cameraAngleX = 0.0f;
float cameraAngleY = 0.0f;
float cameraDistance = 15.0f;
int isDragging = 0;
int lastX, lastY;

float scaleAwan = 1.0f;

void init()
{
    glClearColor(0.5, 0.5, 0.5, 1.0);
    glEnable(GL_DEPTH_TEST);


    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);

    glEnable(GL_COLOR_MATERIAL);
    glColorMaterial(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE);

    GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
    GLfloat mat_shininess[] = { 50.0 };
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
}

void drawAxes()
{
    if (!showAxes)
    return;

    glLineWidth(2.0);

    glColor3f(1.0, 0.0, 0.0);
    glBegin(GL_LINES);
    glVertex3f(-100.0, 0.0, 0.0);
    glVertex3f(100.0, 0.0, 0.0);
    glEnd();

    glColor3f(0.0, 1.0, 0.0);
    glBegin(GL_LINES);
```

```
    glVertex3f(0.0, -100.0, 0.0);
    glVertex3f(0.0, 100.0, 0.0);
    glEnd();

    glColor3f(0.0, 0.0, 1.0);
    glBegin(GL_LINES);
    glVertex3f(0.0, 0.0, -100.0);
    glVertex3f(0.0, 0.0, 100.0);
    glEnd();
}

void lahan()
{
    glPushMatrix();
    glColor3f(0.6, 0.6, 0.6);
    glScalef(25, 0.2, 20);
    glutSolidCube(1);
    glPopMatrix();
}

void lapang1()
{
    glPushMatrix();
    glColor3f(0, 0.6, 0);
    glScalef(1.1, 0.15, 7.5);
    glutSolidCube(1);
    glPopMatrix();
}

void lapang2()
{
    glPushMatrix();
    glColor3f(0, 0.7, 0);
    glScalef(1.1, 0.15, 7.5);
    glutSolidCube(1);
    glPopMatrix();
}

void lapang()
{
    float x1 = 4.95;
    float x2 = 3.85;
    for (int i = 0; i < 5; i++)
    {
        glPushMatrix();
        glTranslatef(x1, 0.2, 0);
        lapang1();
        glPopMatrix();
        x1 -= 2.2;
    }
    for (int i = 0; i < 5; i++)
    {
        glPushMatrix();
        glTranslatef(x2, 0.2, 0);
        lapang2();
        glPopMatrix();
        x2 -= 2.2;
```

```
    }
}

void lapangOut()
{
    glPushMatrix();
    glColor3f(0,0.5,0);
    glTranslatef(0, 0.2, 4);
    glScalef(12, 0.15, 0.5);
    glutSolidCube(1);
    glPopMatrix();
    glPushMatrix();
    glColor3f(0,0.5,0);
    glTranslatef(0, 0.2, -4);
    glScalef(12, 0.15, 0.5);
    glutSolidCube(1);
    glPopMatrix();
    glPushMatrix();
    glColor3f(0,0.5,0);
    glTranslatef(-5.75,0.2,0);
    glScalef(0.5, 0.15, 7.5);
    glutSolidCube(1);
    glPopMatrix();
    glPushMatrix();
    glColor3f(0,0.5,0);
    glTranslatef(5.75, 0.2, 0);
    glScalef(0.5, 0.15, 7.5);
    glutSolidCube(1);
    glPopMatrix();
}

void papanIklan()
{
    glPushMatrix();
    glColor3f(0.7,0,0);
    glTranslatef(0,0.335,-4.255);
    glScalef(12.02,0.12,0.01);
    glutSolidCube(1);
    glPopMatrix();
    glPushMatrix();
    glColor3f(0.7,0,0);
    glTranslatef(-6.005,0.335,0);
    glScalef(0.01,0.12,8.5);
    glutSolidCube(1);
    glPopMatrix();
    glPushMatrix();
    glColor3f(0.7,0,0);
    glTranslatef(6.005,0.335,0);
    glScalef(0.01,0.12,8.5);
    glutSolidCube(1);
    glPopMatrix();
}

void tribunTimur()
{
    float yTranslasi = 0.465f;
    float zTranslasi = -6.31f;
```

```
    float zSkala = 2.9;
    float color = 0.95f;
    glPushMatrix();
    glColor3f(0,0,0.4);
    glTranslatef(0,0.23,-6.01);
    glScalef(12.02,0.2,3.5);
    glutSolidCube(1);
    glPopMatrix();
    glPushMatrix();
    glColor3f(0.25,0.25,0.25);
    glTranslatef(0,0.38,-6.26);
    glScalef(12.02,0.1,3);
    glutSolidCube(1);
    glPopMatrix();
    for(int i=0;i<25;i++){
        glPushMatrix();
        glColor3f(0, 0, color);
        glTranslatef(0,yTranslasi,zTranslasi);
        glScalef(12.02,0.07,zSkala);
        glutSolidCube(1);
        glPopMatrix();
        yTranslasi += 0.07;
        zTranslasi -= 0.05;
        zSkala -= 0.1;
        color -= 0.016;
    }
}
void tribunBarat()
{
    float yTranslasi = 0.465f;
    float zTranslasi = 6.31f;
    float zSkala = 2.9;
    float color = 0.95f;
    glPushMatrix();
    glColor3f(0,0,0.4);
    glTranslatef(0,0.23,6.01);
    glScalef(12.02,0.2,3.5);
    glutSolidCube(1);
    glPopMatrix();
    glPushMatrix();
    glColor3f(0.25,0.25,0.25);
    glTranslatef(0,0.38,6.26);
    glScalef(12.02,0.1,3);
    glutSolidCube(1);
    glPopMatrix();
    for(int i=0;i<25;i++){
        glPushMatrix();
        glColor3f(0,0,color);
        glTranslatef(0,yTranslasi,zTranslasi);
        glScalef(12.02,0.07,zSkala);
        glutSolidCube(1);
        glPopMatrix();
        yTranslasi += 0.07;
        zTranslasi += 0.05;
        zSkala -= 0.1;
        color -= 0.016;
    }
```

```
}

void tribunSelatan()
{
    float yTranslasi = 0.465f;
    float xTranslasi = -8.06f;
    float xSkala = 2.9;
    float color = 0.95f;
    glPushMatrix();
    glColor3f(0,0,0.4);
    glTranslatef(-7.76,0.23,0);
    glScalef(3.5,0.2,8.52);
    glutSolidCube(1);
    glPopMatrix();
    glPushMatrix();
    glColor3f(0.25,0.25,0.25);
    glTranslatef(-8.01,0.38,0);
    glScalef(3,0.1,8.52);
    glutSolidCube(1);
    glPopMatrix();
    for(int i=0;i<25;i++){
        glPushMatrix();
        glColor3f(0,0,color);
        glTranslatef(xTranslasi,yTranslasi,0);
        glScalef(xSkala,0.07,8.52);
        glutSolidCube(1);
        glPopMatrix();
        yTranslasi += 0.07;
        xTranslasi -= 0.05;
        xSkala -= 0.1;
        color -= 0.016;
    }
}

void tribunUtara()
{
    float yTranslasi = 0.465f;
    float xTranslasi = 8.06f;
    float xSkala = 2.9;
    float color = 0.95f;
    glPushMatrix();
    glColor3f(0,0,0.4);
    glTranslatef(7.76,0.23,0);
    glScalef(3.5,0.2,8.52);
    glutSolidCube(1);
    glPopMatrix();
    glPushMatrix();
    glColor3f(0.25,0.25,0.25);
    glTranslatef(8.01,0.38,0);
    glScalef(3,0.1,8.52);
    glutSolidCube(1);
    glPopMatrix();
    for(int i=0;i<25;i++){
        glPushMatrix();
        glColor3f(0,0,color);
        glTranslatef(xTranslasi,yTranslasi,0);
        glScalef(xSkala,0.07,8.52);
```

```
        glutSolidCube(1);
        glPopMatrix();
        yTranslasi += 0.07;
        xTranslasi += 0.05;
        xSkala -= 0.1;
        color -= 0.016;
    }
}

void tribun(){
    glPushMatrix();
    tribunTimur();
    tribunBarat();
    tribunSelatan();
    tribunUtara();
    glPopMatrix();
    glPushMatrix();
    glColor3f(0,0,0.4);
    glTranslatef(-6.51,0.23,-4.76);
    glScalef(1,0.2,1);
    glutSolidCube(1);
    glPopMatrix();
    glPushMatrix();
    glColor3f(0,0,0.4);
    glTranslatef(6.51,0.23,-4.76);
    glScalef(1,0.2,1);
    glutSolidCube(1);
    glPopMatrix();
    glPushMatrix();
    glColor3f(0,0,0.4);
    glTranslatef(-6.51,0.23,4.76);
    glScalef(1,0.2,1);
    glutSolidCube(1);
    glPopMatrix();
    glPushMatrix();
    glColor3f(0,0,0.4);
    glTranslatef(6.51,0.23,4.76);
    glScalef(1,0.2,1);
    glutSolidCube(1);
    glPopMatrix();
}

void atapTribun(){
    glPushMatrix();
    glPopMatrix();
}

void banchPemain(){
    glPushMatrix();
    glColor3f(0.8,0.8,0.8);
    glTranslatef(3.92,0.44,4.656);
    glScalef(2,0.3,0.2);
    glutSolidCube(1);
    glPopMatrix();
    glPushMatrix();
    glColor3f(0.8,0.8,0.8);
    glTranslatef(-3.92,0.44,4.656);
```

```
    glScalef(2,0.3,0.2);
    glutSolidCube(1);
    glPopMatrix();
}

void gedung(){
    glPushMatrix();
    glColor3f(0.2,0.2,0.2);
    glTranslatef(-7.76,0.455,-5.51);
    glScalef(1.5,0.65,2.5);
    glutSolidCube(1);
    glPopMatrix();
    glPushMatrix();
    glColor3f(0.2,0.2,0.2);
    glTranslatef(-6.51,0.455,-6.01);
    glScalef(1,0.65,1.5);
    glutSolidCube(1);
    glPopMatrix();
    glPushMatrix();
    glColor3f(0.2,0.2,0.2);
    glTranslatef(7.76,0.455,-5.51);
    glScalef(1.5,0.65,2.5);
    glutSolidCube(1);
    glPopMatrix();
    glPushMatrix();
    glColor3f(0.2,0.2,0.2);
    glTranslatef(6.51,0.455,-6.01);
    glScalef(1,0.65,1.5);
    glutSolidCube(1);
    glPopMatrix();
    glPushMatrix();
    glColor3f(0.2,0.2,0.2);
    glTranslatef(-7.76,0.455,5.51);
    glScalef(1.5,0.65,2.5);
    glutSolidCube(1);
    glPopMatrix();
    glPushMatrix();
    glColor3f(0.2,0.2,0.2);
    glTranslatef(-6.51,0.455,6.01);
    glScalef(1,0.65,1.5);
    glutSolidCube(1);
    glPopMatrix();
    glPushMatrix();
    glColor3f(0.2,0.2,0.2);
    glTranslatef(7.76,0.455,5.51);
    glScalef(1.5,0.65,2.5);
    glutSolidCube(1);
    glPopMatrix();
    glPushMatrix();
    glColor3f(0.2,0.2,0.2);
    glTranslatef(6.51,0.455,6.01);
    glScalef(1,0.65,1.5);
    glutSolidCube(1);
    glPopMatrix();
}

void bola()
```

```
{
    glPushMatrix();
    glTranslatef(bolaPosisiX, 0.35f, bolaPosisiZ);
    glRotatef(bolaRotasi, 0.0f, 0.0f, 1.0f);

    glColor3f(1.0f, 1.0f, 1.0f);
    glutSolidSphere(0.08, 20, 20);

    glColor3f(0.0f, 0.0f, 0.0f);
    glutWireSphere(0.07, 10, 10);

    glPopMatrix();
}

void matahari() {
    GLfloat light_position[] = { 0.0f, 18.0f, 0.0f, 1.0f };

    GLfloat light_ambient[] = { 0.2f, 0.2f, 0.2f, 1.0f };
    GLfloat light_diffuse[] = { 1.0f, 1.0f, 1.0f, 1.0f };
    GLfloat light_specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };

    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);

    glPushMatrix();
    glDisable(GL_LIGHTING);
    glColor3f(1.0f, 1.0f, 0.0f);
    glTranslatef(0.0f, 10.0f, 0.0f);
    glutSolidSphere(1.0, 20, 20);
    glEnable(GL_LIGHTING);
    glPopMatrix();
}

void awanSatu()
{
    glPushMatrix(); // Awan 1
    glColor3f(0.9f, 0.9f, 0.9f);
    glTranslatef(-7.0f, 6.5f, 3.0f);
    glScalef(scaleAwan, scaleAwan, scaleAwan);
    glutSolidSphere(1.0, 10, 10);
    glPopMatrix();

    glPushMatrix(); // Awan 2
    glColor3f(0.9f, 0.9f, 0.9f);
    glTranslatef(-6.5f, 6.9f, 2.4f);
    glScalef(scaleAwan, scaleAwan, scaleAwan);
    glutSolidSphere(1.15, 10, 11);
    glPopMatrix();

    glPushMatrix(); // Awan 3
    glColor3f(0.9f, 0.9f, 0.9f);
    glTranslatef(-6.5f, 6.5f, 2.0f);
    glScalef(scaleAwan, scaleAwan, scaleAwan);
    glutSolidSphere(1.1, 10, 11);
    glPopMatrix();
```

```
    glPushMatrix(); // Awan 4
    glColor3f(0.9f, 0.9f, 0.9f);
    glTranslatef(-5.2f, 6.5f, 2.7f);
    glScalef(scaleAwan, scaleAwan, scaleAwan);
    glutSolidSphere(1.5, 12, 10);
    glPopMatrix();
}

void awanDua()
{
    glPushMatrix(); // Awan 5
    glColor3f(0.9f, 0.9f, 0.9f);
    glTranslatef(5.3f, 6.5f, -3.2f);
    glScalef(scaleAwan, scaleAwan, scaleAwan);
    glutSolidSphere(1.4, 12, 9.0);
    glPopMatrix();

    glPushMatrix(); // Awan 6
    glColor3f(0.9f, 0.9f, 0.9f);
    glTranslatef(4.0f, 6.5f, -3.0f);
    glScalef(scaleAwan, scaleAwan, scaleAwan);
    glutSolidSphere(1.13, 12, 9.0);
    glPopMatrix();

    glPushMatrix(); // Awan 7
    glColor3f(0.9f, 0.9f, 0.9f);
    glTranslatef(6.5f, 6.5f, -3.0f);
    glScalef(scaleAwan, scaleAwan, scaleAwan);
    glutSolidSphere(1.0, 12, 9.0);
    glPopMatrix();

    glPushMatrix(); // Awan 8
    glColor3f(0.9f, 0.9f, 0.9f);
    glTranslatef(4.0f, 6.8f, -3.0f);
    glScalef(scaleAwan, scaleAwan, scaleAwan);
    glutSolidSphere(1.0, 12, 9.0);
    glPopMatrix();
}

void awanTiga()
{
    glPushMatrix(); // Awan 9
    glColor3f(0.9f, 0.9f, 0.9f);
    glTranslatef(-5.0f, 6.5f, -3.2f);
    glScalef(scaleAwan, scaleAwan, scaleAwan);
    glutSolidSphere(1.4, 12, 9.0);
    glPopMatrix();

    glPushMatrix(); // Awan 10
    glColor3f(0.9f, 0.9f, 0.9f);
    glTranslatef(-3.7f, 6.5f, -3.0f);
    glScalef(scaleAwan, scaleAwan, scaleAwan);
    glutSolidSphere(1.13, 12, 9.0);
    glPopMatrix();

    glPushMatrix(); // Awan 11
```

```
    glColor3f(0.9f, 0.9f, 0.9f);
    glTranslatef(-6.2f, 6.5f, -3.0f);
    glScalef(scaleAwan, scaleAwan, scaleAwan);
    glutSolidSphere(1.0, 12, 9.0);
    glPopMatrix();

    glPushMatrix(); // Awan 12
    glColor3f(0.9f, 0.9f, 0.9f);
    glTranslatef(-3.7f, 6.8f, -3.0f);
    glScalef(scaleAwan, scaleAwan, scaleAwan);
    glutSolidSphere(1.0, 12, 9.0);
    glPopMatrix();
}

void awanEmpat()
{
    glPushMatrix(); // Awan 13
    glColor3f(0.9f, 0.9f, 0.9f);
    glTranslatef(5.3f, 6.5f, 3.2f);
    glScalef(scaleAwan, scaleAwan, scaleAwan);
    glutSolidSphere(1.4, 12, 9.0);
    glPopMatrix();

    glPushMatrix(); // Awan 14
    glColor3f(0.9f, 0.9f, 0.9f);
    glTranslatef(4.0f, 6.5f, 3.0f);
    glScalef(scaleAwan, scaleAwan, scaleAwan);
    glutSolidSphere(1.13, 12, 9.0);
    glPopMatrix();

    glPushMatrix(); // Awan 15
    glColor3f(0.9f, 0.9f, 0.9f);
    glTranslatef(6.5f, 6.5f, 3.0f);
    glScalef(scaleAwan, scaleAwan, scaleAwan);
    glutSolidSphere(1.0, 12, 9.0);
    glPopMatrix();

    glPushMatrix(); // Awan 16
    glColor3f(0.9f, 0.9f, 0.9f);
    glTranslatef(4.0f, 6.8f, 3.0f);
    glScalef(scaleAwan, scaleAwan, scaleAwan);
    glutSolidSphere(1.0, 12, 9.0);
    glPopMatrix();
}

void garisLapangan()
{
    glDisable(GL_LIGHTING);
    glLineWidth(2.0);
    glColor3f(1.0f, 1.0f, 1.0f);

    glBegin(GL_LINES);

    // Garis luar lapangan (persegi panjang)
    glVertex3f(-5.5f, 0.275f, -3.75f);
    glVertex3f(5.5f, 0.275f, -3.75f);
    glVertex3f(5.5f, 0.275f, -3.75f);
```

```
    glVertex3f(5.5f, 0.275f, 3.75f);
    glVertex3f(5.5f, 0.275f, 3.75f);
    glVertex3f(-5.5f, 0.275f, 3.75f);
    glVertex3f(-5.5f, 0.275f, 3.75f);
    glVertex3f(-5.5f, 0.275f, -3.75f);

    // Garis tengah
    glVertex3f(0.0f, 0.275f, -3.75f);
    glVertex3f(0.0f, 0.275f, 3.75f);

    // Kotak Penalti Kiri
    glVertex3f(-4.0f, 0.275f, -2.0f);
    glVertex3f(-4.0f, 0.275f, 2.0f);
    glVertex3f(-4.0f, 0.275f, -2.0f);
    glVertex3f(-5.5f, 0.275f, -2.0f);
    glVertex3f(-4.0f, 0.275f, 2.0f);
    glVertex3f(-5.5f, 0.275f, 2.0f);

    // Kotak Penalti Kanan
    glVertex3f(4.0f, 0.275f, -2.0f);
    glVertex3f(4.0f, 0.275f, 2.0f);
    glVertex3f(4.0f, 0.275f, -2.0f);
    glVertex3f(5.5f, 0.275f, -2.0f);
    glVertex3f(4.0f, 0.275f, 2.0f);
    glVertex3f(5.5f, 0.275f, 2.0f);

    // Kotak Gawang Kiri (dekat titik penalti kiri)
    glVertex3f(-5.5f, 0.275f, -1.0f); glVertex3f(-4.8f, 0.275f,
-1.0f);
    glVertex3f(-5.5f, 0.275f, 1.0f);  glVertex3f(-4.8f, 0.275f,
1.0f);
    glVertex3f(-4.8f, 0.275f, -1.0f); glVertex3f(-4.8f, 0.275f,
1.0f);
    glVertex3f(-5.5f, 0.275f, -1.0f); glVertex3f(-5.5f, 0.275f,
1.0f);

    // Kotak Gawang Kanan (dekat titik penalti kanan)
    glVertex3f(5.5f, 0.275f, -1.0f);  glVertex3f(4.8f, 0.275f,
-1.0f);
    glVertex3f(5.5f, 0.275f, 1.0f);   glVertex3f(4.8f,0.275f,
1.0f);
    glVertex3f(4.8f, 0.275f, -1.0f);  glVertex3f(4.8f, 0.275f,
1.0f);
    glVertex3f(5.5f, 0.275f, -1.0f);  glVertex3f(5.5f, 0.275f,
1.0f);

    glEnd();

    // Lingkaran tengah
    glBegin(GL_LINE_LOOP);
    float radius = 1.0f;
    int segments = 100;
    for (int i = 0; i < segments; i++)
    {
        float angle = 2.0f * M_PI * i / segments;
        float x = radius * cos(angle);
        float z = radius * sin(angle);
```

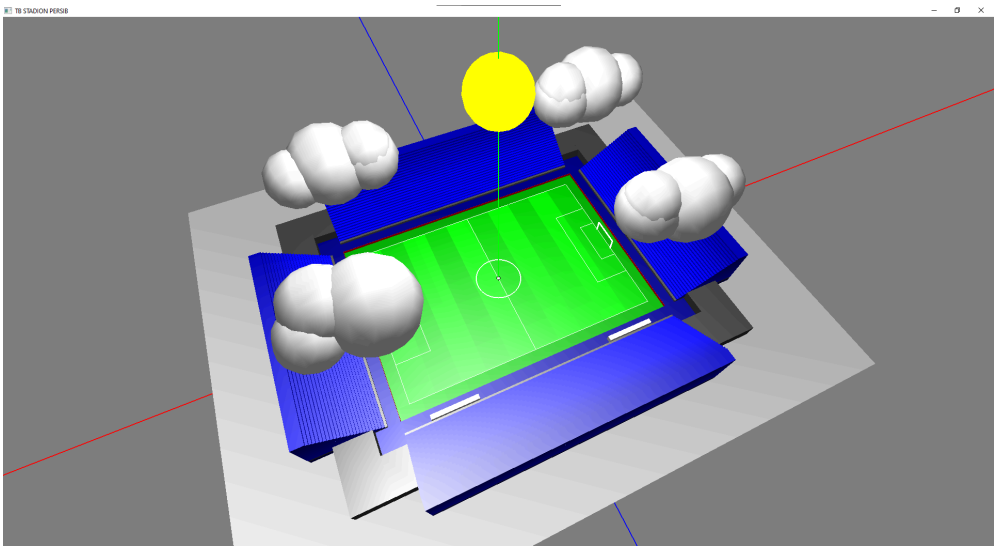
```
        glVertex3f(x, 0.35f, z);
    }
    glEnd();
}
void gawangKiri()
{
    float tiangTebal = 0.05f;
    float tiangTinggi = 0.8f;
    float tiangLebar = 1.2f;

    // Tiang Vertikal Kiri
    glPushMatrix();
    glColor3f(1.0f, 1.0f, 1.0f);
    glTranslatef(-5.5f, tiangTinggi / 2.0f, -tiangLebar / 2.0f);
    glScalef(tiangTebal, tiangTinggi, tiangTebal);
    glutSolidCube(1);
    glPopMatrix();

    // Tiang Vertikal Kanan
    glPushMatrix();
    glColor3f(1.0f, 1.0f, 1.0f);
    glTranslatef(-5.5f, tiangTinggi / 2.0f, tiangLebar / 2.0f);
    glScalef(tiangTebal, tiangTinggi, tiangTebal);
    glutSolidCube(1);
    glPopMatrix();

    // Tiang Horizontal Atas
    glPushMatrix();
    glColor3f(1.0f, 1.0f, 1.0f);
    glTranslatef(-5.5f, tiangTinggi, 0.0f);
    glScalef(tiangTebal, tiangTebal, tiangLebar);
    glutSolidCube(1);
    glPopMatrix();
}

void gawangKanan()
{
    float tiangTebal = 0.05f;
    float tiangTinggi = 0.8f;
    float tiangLebar = 1.2f;

    // Tiang Vertikal Kiri
    glPushMatrix();
    glColor3f(1.0f, 1.0f, 1.0f);
    glTranslatef(5.5f, tiangTinggi / 2.0f, -tiangLebar / 2.0f);
    glScalef(tiangTebal, tiangTinggi, tiangTebal);
    glutSolidCube(1);
    glPopMatrix();

    // Tiang Vertikal Kanan
    glPushMatrix();
    glColor3f(1.0f, 1.0f, 1.0f);
    glTranslatef(5.5f, tiangTinggi / 2.0f, tiangLebar / 2.0f);
    glScalef(tiangTebal, tiangTinggi, tiangTebal);
    glutSolidCube(1);
    glPopMatrix();
```

```
    // Tiang Horizontal Atas
    glPushMatrix();
    glColor3f(1.0f, 1.0f, 1.0f);
    glTranslatef(5.5f, tiangTinggi, 0.0f);
    glScalef(tiangTebal, tiangTebal, tiangLebar);
    glutSolidCube(1);
    glPopMatrix();
}

void keyboard(unsigned char key, int x, int y)
{
    float step = 0.1f;
    float rotStep = 10.0f;

    switch (key)
    {
    case 'x': // Toggle garis sumbu
        showAxes = !showAxes;
        break;
    case 'w': // Bola maju
        bolaPosisiZ -= step;
        bolaRotasi += rotStep;
        break;
    case 's': // Bola mundur
        bolaPosisiZ += step;
        bolaRotasi -= rotStep;
        break;
    case 'a': // Bola ke kiri
        bolaPosisiX -= step;
        bolaRotasi += rotStep;
        break;
    case 'd': // Bola ke kanan
        bolaPosisiX += step;
        bolaRotasi -= rotStep;
        break;
    case '+': // Perbesar skala awan
        scaleAwan += 0.1f;
        break;
    case '-': // Perkecil skala awan
        scaleAwan -= 0.1f;
        if (scaleAwan < 0.1f)
            scaleAwan = 0.1f; // Batas minimum skala
        break;
    case 27: // ESC untuk keluar
        exit(0);
        break;
    }
    glutPostRedisplay();
}

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

    float eyeX = cameraDistance * sin(cameraAngleX) *
cos(cameraAngleY);
```

```
    float eyeY = cameraDistance * sin(cameraAngleY);
    float eyeZ = cameraDistance * cos(cameraAngleX) *
cos(cameraAngleY);

    gluLookAt(eyeX, eyeY, eyeZ,
              0, 0, 0,
              0.0, 1.0, 0.0);

    // Pastikan normal vector dihitung dengan benar
    glEnable(GL_NORMALIZE);

    drawAxes();
    matahari();
    lahan();
    lapang();
    lapangOut();
    papanIklan();
    tribun();
    banchPemain();
    gedung();
    bola();
    awanSatu();
    awanDua();
    awanTiga();
    awanEmpat();
    garisLapangan();
    gawangKiri();
    gawangKanan();

    glutSwapBuffers();
}

void reshape(int w, int h)
{
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60.0, (double)w / (double)h, 1.0, 100.0);
    glMatrixMode(GL_MODELVIEW);
}

void mouseMotion(int x, int y)
{
    if (isDragging)
    {
        float deltaX = (x - lastX) * 0.005f;
        float deltaY = (y - lastY) * 0.005f;

        cameraAngleX += deltaX;
        cameraAngleY += deltaY;

        if (cameraAngleY > 1.5f)
            cameraAngleY = 1.5f;
        if (cameraAngleY < -1.5f)
            cameraAngleY = -1.5f;

        lastX = x;
```

```
        lastY = y;

        glutPostRedisplay();
    }
}

void mouse(int button, int state, int x, int y)
{
    if (button == GLUT_LEFT_BUTTON)
    {
        if (state == GLUT_DOWN)
        {
            isDragging = 1;
            lastX = x;
            lastY = y;
        }
        else if (state == GLUT_UP)
        {
            isDragging = 0;
        }
    }

    // Zoom in dan zoom out menggunakan scroll mouse
    if (button == 3)
    {
        cameraDistance -= 0.5f;
        if (cameraDistance < 2.0f)
            cameraDistance = 2.0f;
        glutPostRedisplay();
    }
    else if (button == 4)
    {
        cameraDistance += 0.5f;
        if (cameraDistance > 50.0f)
            cameraDistance = 50.0f;
        glutPostRedisplay();
    }
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(800, 600);
    glutCreateWindow("TB STADION PERSIB");

    init();

    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMouseFunc(mouse);
    glutMotionFunc(mouseMotion);
    glutKeyboardFunc(keyboard);

    glutMainLoop();
    return 0;
}
```

3.2 Output



Gambar 1 Output Code

Gambar 2 Postingan linkedin

3.3 Penjelasan

1. Include Liblary
    - Kode ini pakai pustaka OpenGL sama GLUT buat bikin aplikasi grafis. `GL/glut.h` itu berfungsi buat nge-handle pembuatan jendela, input dari pengguna, dan menggambar objek 3D. Sedangkan `math.h` dipakai buat perhitungan matematika, misalnya trigonometri, yang sering banget dibutuhin di grafik komputer buat nentuin posisi atau arah objek. Jadi intinya, kode ini bisa dipakai buat bikin aplikasi grafis 3D yang interaktif, di mana kita bisa gambar objek 3D atau animasi di jendela aplikasi.

2. Deklarasi Variable Global

   Kode ini berisi beberapa variabel yang digunakan untuk mengatur posisi, rotasi, dan kontrol kamera pada sebuah aplikasi grafis.

   - bolaRotasi digunakan untuk menyimpan nilai rotasi bola, agar bola bisa diputar.
   - bolaPosisiX dan bolaPosisiZ menyimpan posisi bola pada sumbu X dan Z, yang memungkinkan bola bergerak di ruang 3D.
   - cameraAngleX dan cameraAngleY menyimpan sudut rotasi kamera pada sumbu X dan Y, yang mengatur pandangan kamera terhadap objek dalam scene.
   - cameraDistance mengatur jarak kamera dari objek (misalnya bola), memberi kontrol seberapa dekat atau jauh kamera dengan objek tersebut.
   - isDragging adalah status yang menandakan apakah pengguna sedang menarik atau menggerakkan mouse (digunakan untuk kontrol interaktif).
   - lastX dan lastY menyimpan posisi terakhir mouse di layar, yang digunakan untuk menghitung pergerakan mouse ketika drag.
   - scaleAwan mengatur skala atau ukuran objek awan, sehingga kita bisa mengubah besar kecilnya objek tersebut.

   Secara keseluruhan, kode ini digunakan untuk mengontrol rotasi dan posisi bola, kontrol kamera, serta interaksi pengguna dengan objek dalam aplikasi grafis 3D

3. Fungsi init

   `Fungsi init()` berfungsi untuk mengatur pengaturan dasar dalam OpenGL yang diperlukan untuk rendering 3D. Pertama, fungsi ini mengatur warna latar belakang menjadi abu-abu menggunakan `glClearColor()`. Kemudian, `glEnable(GL_DEPTH_TEST)` diaktifkan agar objek yang lebih dekat dengan kamera tidak tertutup oleh objek yang lebih jauh. Pencahayaan diaktifkan dengan `glEnable(GL_LIGHTING)` dan sumber cahaya pertama diaktifkan menggunakan `glEnable(GL_LIGHT0)`. Selain itu, pengaturan material warna diaktifkan dengan `glEnable(GL_COLOR_MATERIAL)`, yang memungkinkan objek memiliki warna yang dipengaruhi oleh pencahayaan. Material objek juga diatur dengan mendefinisikan nilai pantulan spekular dan kekerasan pantulan menggunakan `glMaterialfv()`. Secara keseluruhan, fungsi ini menyiapkan semua pengaturan dasar agar objek yang digambar dapat terlihat dengan pencahayaan yang realistis dan efek material yang sesuai.

4. Fungsi drawAxes

   Fungsi `drawAxes()` digunakan untuk menggambar sumbu koordinat 3D di ruang tampilan. Fungsi ini pertama-tama memeriksa apakah sumbu harus ditampilkan dengan variabel showAxes. Jika showAxes bernilai false, fungsi ini akan keluar dan tidak melakukan apa-apa. Jika showAxes bernilai true, fungsi akan menggambar tiga sumbu koordinat: 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 menggunakan dua titik dengan `glBegin(GL_LINES)` dan `glEnd()`, yang membentuk garis dari nilai -100

hingga 100 pada masing-masing sumbu. Fungsi ini memberikan panduan visual untuk melihat arah sumbu koordinat dalam ruang 3D.

5. Fungsi lahan

   Fungsi `lahan()` digunakan untuk menggambar sebuah objek 3D berbentuk kubus yang telah diubah ukurannya untuk merepresentasikan lahan atau permukaan. Di dalam fungsi ini, pertama-tama dipanggil `glPushMatrix()` untuk menyimpan matriks transformasi saat ini, sehingga perubahan transformasi yang dilakukan dalam fungsi ini tidak mempengaruhi objek lainnya. Kemudian, dengan `glColor3f(0.6, 0.6, 0.6)`, warna objek diubah menjadi abu-abu. Fungsi `glScalef(25, 0.2, 20)` digunakan untuk melakukan skala objek, mengubah kubus menjadi bentuk lebih datar, menyerupai lahan atau permukaan yang panjang dan lebar. Setelah itu, `glutSolidCube(1)` menggambar kubus dengan sisi panjang 1 unit yang telah diskalakan sesuai transformasi sebelumnya. Terakhir, `glPopMatrix()` mengembalikan matriks transformasi ke keadaan semula, sehingga objek lain yang digambar setelahnya tidak terpengaruh oleh transformasi yang baru saja diterapkan.

6. Fungsi lapang1

   Fungsi `lapang1()` digunakan untuk menggambar objek 3D yang menyerupai lapangan dengan bentuk kubus yang telah diubah ukurannya. Fungsi ini dimulai dengan memanggil `glPushMatrix()` untuk menyimpan status transformasi matriks saat ini, agar transformasi yang dilakukan tidak memengaruhi objek lainnya. Kemudian, warna objek diubah menjadi hijau dengan `glColor3f(0, 0.6, 0)`. Fungsi `glScalef(1.1, 0.15, 7.5)` mengubah ukuran kubus, menjadikannya lebih datar dan lebih panjang pada sumbu X dan Z, menciptakan bentuk yang lebih menyerupai lapangan. Setelah itu, `glutSolidCube(1)` menggambar kubus dengan sisi panjang 1 unit yang telah diterapkan skala. Akhirnya, `glPopMatrix()` digunakan untuk mengembalikan matriks transformasi ke keadaan semula, sehingga objek lain yang digambar setelahnya tidak terpengaruh oleh transformasi yang baru saja diterapkan.

7. Fungsi lapang2

   Fungsi `lapang2()` digunakan untuk menggambar objek 3D yang menyerupai lapangan dengan bentuk kubus yang telah diubah ukurannya. Fungsi ini dimulai dengan memanggil `glPushMatrix()` untuk menyimpan status transformasi matriks saat ini, agar transformasi yang dilakukan tidak memengaruhi objek lainnya. Kemudian, warna objek diubah menjadi hijau dengan `glColor3f(0, 0.7, 0)`. Fungsi `glScalef(1.1, 0.15, 7.5)` mengubah ukuran kubus, menjadikannya lebih datar dan lebih panjang pada sumbu X dan Z, menciptakan bentuk yang lebih menyerupai lapangan. Setelah itu, `glutSolidCube(1)` menggambar kubus dengan sisi panjang 1 unit yang telah diterapkan skala. Akhirnya, `glPopMatrix()` digunakan untuk mengembalikan matriks transformasi ke keadaan semula, sehingga objek lain yang digambar setelahnya tidak terpengaruh oleh transformasi yang baru saja diterapkan.

8. Fungsi lapang

Fungsi `lapang()` menggambar dua set objek lapangan dengan posisi berbeda. Dua loop digunakan untuk menggambar lima objek lapangan menggunakan fungsi `lapang1()` dan `lapang2()`. Posisi setiap objek bergeser 2.2 unit ke kiri, dimulai dari posisi awal x1 dan x2. Setiap gambar lapangan dibungkus dengan `glPushMatrix()` dan `glPopMatrix()` agar transformasi posisi tidak memengaruhi objek lainnya. Fungsi ini menciptakan deretan lapangan di ruang 3D.

9. Fungsi lapangOut

   Fungsi `lapangOut()` menggambar batas luar lapangan dengan empat objek kubus yang diposisikan di sekitar pusat. Dua objek pertama digambar di sepanjang sumbu Z, sementara dua objek lainnya di sepanjang sumbu X. Masing-masing objek memiliki skala panjang yang besar dan tinggi rendah. Setiap objek digambar secara terpisah dengan menggunakan `glPushMatrix()` dan `glPopMatrix()` untuk memastikan transformasi posisi tidak memengaruhi objek lainnya. Fungsi ini membentuk struktur batas luar lapangan di ruang 3D.

10. Fungsi papanIklan

    Fungsi papanIklan() menggambar papan iklan dengan tiga objek kubus yang diatur sedemikian rupa untuk membentuk struktur papan iklan. Pertama, objek utama digambar di posisi tertentu dengan skala panjang lebih besar dan tinggi rendah. Dua objek lainnya digambar di sisi kiri dan kanan objek utama, masing-masing dengan skala kecil di sumbu X dan panjang di sumbu Z. Semua objek diberi warna merah dan posisinya disesuaikan dengan transformasi glTranslatef() dan glScalef(). Fungsi ini menciptakan struktur papan iklan di ruang 3D.

11. Fungsi tribunTimur

    Fungsi tribunTimur() menggambar struktur tribun dengan beberapa lapisan bertingkat. Pertama, dua objek kubus digambar sebagai bagian dasar tribun, dengan warna biru tua untuk bagian bawah dan abu-abu untuk bagian atasnya. Setelah itu, sebuah loop digunakan untuk menggambar 25 lapisan kubus yang lebih kecil secara bertahap. Setiap lapisan digambar dengan warna biru yang semakin terang, posisi dan ukuran yang semakin kecil, serta jarak antar lapisan yang semakin pendek. Transformasi glTranslatef() dan glScalef() digunakan untuk mengatur posisi dan ukuran setiap objek, sementara warna objek berubah setiap iterasi. Fungsi ini menciptakan efek tribun bertingkat yang mengarah ke tengah.

12. Fungsi tribunBarat

    Fungsi tribunBarat() menggambar tribun di sisi barat lapangan dengan struktur bertingkat. Dua bagian utama digambar terlebih dahulu sebagai dasar tribun, dengan warna biru tua untuk bagian bawah dan abu-abu untuk bagian atasnya. Selanjutnya, sebuah loop menggambar 25 lapisan tribun yang semakin kecil. Setiap lapisan memiliki warna biru yang secara bertahap lebih terang, posisi yang lebih tinggi, dan ukuran yang mengecil di sumbu Z. Transformasi glTranslatef() dan glScalef() digunakan untuk mengatur posisi dan ukuran setiap lapisan, dengan perubahan posisi yang mengarah ke sumbu Z positif. Fungsi ini menciptakan tampilan tribun bertingkat yang simetris dengan tribunTimur().

13. Fungsi tribunSelatan

Fungsi tribunSelatan() menggambar tribun bertingkat di sisi selatan lapangan. Bagian dasar tribun digambar terlebih dahulu menggunakan dua objek kubus: bagian bawah dengan warna biru tua dan bagian atas dengan warna abu-abu. Setelah itu, loop menggambar 25 lapisan tribun bertingkat, dengan setiap lapisan memiliki warna biru yang semakin terang, posisi lebih tinggi, dan ukuran lebih kecil di sumbu X. Transformasi posisi dan skala diterapkan di setiap iterasi menggunakan glTranslatef() dan glScalef(). Fungsi ini menghasilkan tampilan tribun yang berlapis dan terletak di sisi selatan lapangan.

14. Fungsi tribunUtara

Fungsi tribunUtara() menggambar tribun bertingkat di sisi utara lapangan. Dua bagian dasar tribun digambar lebih dulu: bagian bawah dengan warna biru tua dan bagian atas dengan warna abu-abu. Selanjutnya, loop menggambar 25 lapisan tribun yang semakin kecil, dengan posisi bergerak ke sumbu X positif dan ketinggian bertambah pada setiap iterasi. Warna setiap lapisan secara bertahap menjadi lebih terang, dan ukurannya di sumbu X mengecil. Fungsi ini menghasilkan tampilan tribun berlapis yang simetris dengan tribunSelatan().

15. Fungsi tribun

Fungsi tribun() menggambar keseluruhan struktur tribun dengan memanggil empat fungsi utama: tribunTimur(), tribunBarat(), tribunSelatan(), dan tribunUtara(). Keempat fungsi tersebut menciptakan tribun bertingkat di masing-masing sisi lapangan. Selain itu, fungsi ini juga menambahkan empat kubus kecil di keempat sudut tribun dengan warna biru tua. Kubus ini berfungsi sebagai elemen tambahan untuk memperkuat tampilan struktur tribun. Semua transformasi posisi dan skala dikendalikan dengan glPushMatrix() dan glPopMatrix() untuk menjaga transformasi tetap terorganisasi.

16. Fungsi banchPemain

Fungsi banchPemain() menggambar dua bangku pemain dengan posisi simetris di sisi tertentu lapangan. Bangku-bangku tersebut digambar sebagai kubus kecil dengan warna abu-abu terang. Bangku pertama ditempatkan di posisi positif sumbu X, sementara bangku kedua ditempatkan di posisi negatif sumbu X dengan sumbu Z yang sama. Transformasi posisi dan skala setiap bangku diatur menggunakan glTranslatef() dan glScalef(). Kedua bangku dibungkus dalam glPushMatrix() dan glPopMatrix() untuk memastikan perubahan transformasi tidak memengaruhi elemen lain di dalam program.

17. Fungsi gedung

Fungsi gedung() menggambar delapan bangunan kecil yang tersebar di sekitar area lapangan. Setiap bangunan digambarkan sebagai kubus dengan warna abu-abu gelap, mewakili tampilan gedung. Posisi dan ukuran masing-masing gedung ditentukan menggunakan kombinasi transformasi glTranslatef() dan glScalef(). Keempat gedung pertama ditempatkan di dekat sudut-sudut bagian selatan lapangan, sementara empat gedung lainnya ditempatkan di sudut-sudut bagian utara. Transformasi setiap gedung dibungkus dengan glPushMatrix()

dan glPopMatrix() untuk menjaga pengaturan transformasi tetap terisolasi. Fungsi ini menambahkan elemen visual tambahan untuk melengkapi area sekitar lapangan.

18. Fungsi bola

Fungsi bola() digunakan untuk menggambar bola pada lapangan. Bola diposisikan berdasarkan nilai variabel global bolaPosisiX dan bolaPosisiZ, sementara rotasinya ditentukan oleh bolaRotasi. Bola memiliki dua elemen utama: bola solid berwarna putih yang digambar menggunakan glutSolidSphere() dan kerangka kawat berwarna hitam yang dibuat dengan glutWireSphere(), memberikan tampilan lebih detail pada bola. Transformasi posisi dan rotasi bola diterapkan menggunakan glTranslatef() dan glRotatef(), dengan semua pengaturan transformasi dijaga tetap terisolasi menggunakan glPushMatrix() dan glPopMatrix().

19. Fungsi matahari

Fungsi matahari() bertujuan untuk mensimulasikan matahari sekaligus sebagai sumber cahaya utama dalam adegan. Sumber cahaya diatur menggunakan GL_LIGHT0, dengan posisi di atas adegan pada koordinat (0.0f, 18.0f, 0.0f). Komponen pencahayaan ambient, diffuse, dan specular diatur untuk menciptakan efek pencahayaan yang realistis. Selain itu, fungsi ini menggambar representasi visual matahari sebagai bola kuning solid di koordinat (0.0f, 10.0f, 0.0f) menggunakan glutSolidSphere(). Untuk memastikan bola tidak terpengaruh oleh pencahayaan, fungsi ini sementara menonaktifkan pencahayaan dengan glDisable(GL_LIGHTING) sebelum menggambar matahari, lalu mengaktifkannya kembali dengan glEnable(GL_LIGHTING). Transformasi dijaga dengan pasangan glPushMatrix() dan glPopMatrix().

20. Fungsi awanSatu

Fungsi awanSatu() menggambar satu kelompok awan menggunakan empat bola putih dengan posisi dan ukuran berbeda. Setiap bola ditransformasi dengan glTranslatef dan glScalef untuk menciptakan tampilan awan yang alami di langit.

21. Fungsi awanDua

Fungsi awanDua() menggambar satu kelompok awan menggunakan empat bola putih dengan posisi dan ukuran berbeda. Setiap bola ditransformasi dengan glTranslatef dan glScalef untuk menciptakan tampilan awan yang alami di langit.

22. Fungsi awanTiga

Fungsi awanTiga() menggambar satu kelompok awan menggunakan empat bola putih dengan posisi dan ukuran berbeda. Setiap bola ditransformasi dengan glTranslatef dan glScalef untuk menciptakan tampilan awan yang alami di langit.

23. Fungsi awanEmpat

Fungsi awanEmpat() menggambar satu kelompok awan menggunakan empat bola putih dengan posisi dan ukuran berbeda. Setiap bola ditransformasi dengan glTranslatef dan glScalef untuk menciptakan tampilan awan yang

alami di langit.

24. Fungsi garisLapangan

Fungsi garisLapangan() menggambar lapangan sepak bola menggunakan beberapa garis. Ini mencakup garis luar lapangan, garis tengah, kotak penalti, kotak gawang, dan lingkaran tengah. Garis digambar dengan glBegin(GL_LINES) untuk setiap bagian, sementara lingkaran tengah digambar menggunakan glBegin(GL_LINE_LOOP) dengan koordinat dihitung berdasarkan sudut.

25. Fungsi gawangKiri

Fungsi gawangKiri() menggambar gawang sepak bola di sisi kiri dengan menggunakan tiga tiang berbentuk kubus. Tiang-tiang tersebut mencakup dua tiang vertikal di kiri dan kanan serta satu tiang horizontal di bagian atas. Tiang-tiang ini digambar menggunakan OpenGL dengan warna putih dan ukuran yang ditentukan oleh variabel tiangTebal, tiangTinggi, dan tiangLebar untuk memberi tampilan gawang yang realistis di lapangan.

26. Fungsi gawangKanan

Fungsi gawangKanan() menggambar gawang sepak bola di sisi kanan lapangan dengan tiga tiang berbentuk kubus. Sama seperti pada gawangKiri(), fungsi ini menggambar dua tiang vertikal di kiri dan kanan serta satu tiang horizontal di bagian atas gawang. Tiang-tiang ini memiliki warna putih dengan ukuran yang ditentukan oleh variabel tiangTebal, tiangTinggi, dan tiangLebar, yang memberikan tampilan visual dari gawang yang terletak di sisi kanan lapangan.

27. Fungsi keyboard

Fungsi keyboard() menangani input keyboard dalam program OpenGL, di mana setiap tombol yang ditekan mengubah posisi, rotasi, atau skala objek dalam scene. Tombol 'w', 'a', 's', 'd' menggerakkan bola maju, mundur, kiri, atau kanan sambil memutar bola. Tombol '+' dan '-' mengubah skala awan, sementara tombol 'x' menampilkan atau menyembunyikan sumbu koordinat. Tombol ESC akan keluar dari program. Fungsi ini memperbarui tampilan dengan glutPostRedisplay() setelah setiap perubahan.

28. Fungsi display

Fungsi display() bertanggung jawab untuk menggambar seluruh scene dalam program OpenGL. Dimulai dengan membersihkan buffer warna dan kedalaman, lalu mengatur posisi kamera menggunakan gluLookAt(). Posisi kamera dihitung berdasarkan parameter sudut dan jarak (cameraAngleX, cameraAngleY, dan cameraDistance). Fungsi ini juga memastikan vektor normal dihitung dengan benar menggunakan glEnable(GL_NORMALIZE). Setelah itu, fungsi menggambar berbagai objek seperti matahari, lapangan, awan, bola, gawang, dan elemen lainnya dengan memanggil fungsi yang sesuai. Akhirnya, tampilan diperbarui dengan glutSwapBuffers() untuk menampilkan hasil render.

29. Fungsi reshape

Fungsi reshape() digunakan untuk mengubah tampilan sesuai dengan ukuran jendela. Ketika ukuran jendela diubah, fungsi ini memanggil glViewport()

untuk menentukan area tampilan yang akan digunakan. Kemudian, mengubah mode matriks ke GL_PROJECTION dan menggunakan gluPerspective() untuk mengatur perspektif proyeksi dengan sudut pandang 60 derajat, rasio aspek yang disesuaikan dengan ukuran jendela, dan jarak dekat serta jauh kamera. Terakhir, fungsi ini mengembalikan mode matriks ke GL_MODELVIEW untuk melanjutkan dengan penggambaran objek-objek 3D.

30. Fungsi mouseMotion

Fungsi mouseMotion() digunakan untuk mengatur gerakan kamera berdasarkan pergerakan mouse. Ketika mouse digerakkan (selama status isDragging aktif), fungsi ini menghitung perubahan posisi mouse pada sumbu X dan Y. Perubahan ini kemudian digunakan untuk memperbarui sudut pandang kamera (cameraAngleX dan cameraAngleY). Pembatasan diterapkan pada cameraAngleY agar tidak melebihi 1.5f atau -1.5f untuk mencegah rotasi yang tidak diinginkan. Setelah pembaruan sudut pandang, fungsi ini memanggil glutPostRedisplay() untuk menggambar ulang tampilan sesuai posisi kamera yang baru.

31. Fungsi mouse

Fungsi mouse() menangani interaksi mouse untuk menggerakkan kamera dan zoom. Ketika tombol kiri mouse ditekan, kamera mulai digerakkan sesuai pergerakan mouse, dengan pembatasan sudut agar tidak terlalu tinggi atau rendah. Untuk zoom, scroll mouse digunakan untuk memperbesar atau memperkecil jarak kamera, dengan batas jarak antara 2.0f dan 50.0f. Setelah perubahan, tampilan diperbarui menggunakan glutPostRedisplay().

32. Fungsi main

Fungsi main() ini digunakan untuk menginisialisasi aplikasi dengan GLUT. Pertama, GLUT diinisialisasi menggunakan glutInit(), kemudian mode tampilan ditetapkan dengan glutInitDisplayMode() untuk menggunakan double buffering, mode warna RGB, dan depth buffering. Ukuran jendela diatur ke 800x600 pixel melalui glutInitWindowSize(), dan jendela dengan judul "TB STADION PERSIB" dibuat menggunakan glutCreateWindow(). Fungsi init() dipanggil untuk menyiapkan pengaturan OpenGL seperti kamera dan pencahayaan.

Setelah itu, berbagai fungsi callback didaftarkan: display untuk rendering, reshape untuk penyesuaian ukuran jendela, mouse dan mouseMotion untuk menangani interaksi mouse, serta keyboard untuk input dari keyboard. Terakhir, glutMainLoop() memulai loop utama aplikasi yang memungkinkan interaksi dan rendering terus berjalan.

BAB IV

4.1. Kesimpulan

Grafis komputer adalah bidang yang sangat penting dalam pengolahan data visual menggunakan komputer, yang telah berkembang pesat sejak awal 1960-an. Teknologi ini kini digunakan dalam banyak sektor, seperti film, game, dan desain arsitektur. Perkembangan grafis komputer memungkinkan penciptaan visual yang lebih kompleks dan realistis.

OpenGL adalah pustaka grafis yang penting untuk pembuatan aplikasi berbasis grafis 2D dan 3D. Dengan menyediakan berbagai fungsi untuk menggambar objek, melakukan manipulasi transformasi, serta pengaturan pencahayaan, OpenGL menjadi dasar yang penting dalam memahami grafis komputer. Pemahaman tentang cara kerja OpenGL, mulai dari inisialisasi hingga rendering, memberikan wawasan tentang proses pembuatan visual yang efisien dan halus.

Unity Hub adalah alat bantu yang sangat berguna bagi pengembang game untuk mengelola berbagai proyek dan versi Unity Editor. Melalui Unity Hub, pengelolaan proyek dan instalasi modul tambahan menjadi lebih mudah, serta memberikan akses ke dokumentasi dan tutorial yang dapat mempercepat proses belajar dan pengembangan.

GitHub berperan sebagai platform kolaborasi yang memungkinkan pengembang untuk bekerja bersama dalam mengelola kode sumber dengan sistem kontrol versi Git. Dengan fitur-fitur seperti repositori, branching, dan merging, GitHub memfasilitasi pengelolaan proyek perangkat lunak secara efisien dan menjaga integritas kode yang dikembangkan.

Pembuatan Visualisasi Stadion Bola Dengan Unity Hub: Proses pembuatan stadion bola di Unity Hub dimulai dengan membuat proyek baru dan memilih template 3D. Kemudian, elemen-elemen seperti lapangan bola dan tribun dibuat menggunakan objek dasar seperti cube. Unity Hub menyediakan alat yang memudahkan pengembangan dan desain grafis secara visual.

Pembuatan Stadion Bola Dengan OpenGL: Pembuatan stadion bola dengan OpenGL dimulai dengan membuat folder dan file utama, lalu dilanjutkan dengan pembuatan objek 3D seperti lapangan, tribun, dan bangku menggunakan cube. Selain itu, penciptaan elemen-elemen visual seperti matahari dan awan menggunakan sphere memperkaya visualisasi tersebut. Penggunaan OpenGL memungkinkan pengembangan grafis yang lebih terperinci meskipun lebih teknis dibandingkan Unity Hub.

DAFTAR PUSTAKA

Deddy Suhardiman, G. Kaunang, Rizal Sengkey, A. M. R. (2012). Pembuatan Simulasi Pergerakan Objek 3D menggunakan OpenGL. *Jurnal Teknik Elektro Dan Komputer*, *1*, 2.

Dwi Agus Kurniawan, Muhammad Iqbal, M. F. (2021). Mekanika Fisika untuk Gerak Objek 3D Berbasis OpenGL sebagai Aplikasi Media Pembelajaran. *Jurnal Ilmiah Pendidikan Fisika Al-BiRuNi*, *10*(1).

Muhammad Adnani, A. Z. F. (2021). Implementasi OpenGL untuk Pembuatan Objek 3D. *Journal Zetroem*, *3*(1).

Universitas Sains dan Teknologi Komputer (STEKOM). (n.d.). *Jurnal Pixel - Teknologi Komputer dan Seni Media*. https://journal.stekom.ac.id/index.php/pixel