LAPORAN PRATIKUM GRAFIK KOMPUTER

Diajukan untuk memenuhi Tugas mata kuliah Pratikum Grafik Komputer

PIRAMIDA SUKU MAYA

Dosen Pengampu : Sri Rahayu, M.Kom

Instruktur Pratikum : Arul Budi Kalimat, S.Kom

Disusun oleh

Kelompok : 2

Karina Hoirun Nisa

2306114

Rifaldi Al Khausaeri Dwi Putra

2306127

wildan syaeful millah albantani

2306118

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 cara membuat objek piramida suku maya.

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, 11 Desember 2024

Karina

DAFTAR ISI

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Peradaban suku Maya merupakan salah satu peradaban kuno yang terkenal dengan warisan arsitektur yang luar biasa, termasuk piramida-piramidanya yang megah. Piramida suku Maya tidak hanya menjadi simbol kekayaan budaya dan sejarah, tetapi juga menyimpan nilai-nilai estetika dan ilmiah yang menarik untuk dipelajari. Dalam era digital saat ini, visualisasi dan simulasi berbasis komputer menjadi salah satu metode efektif untuk merekonstruksi struktur bersejarah, seperti piramida suku Maya, guna mempelajari desain dan fungsinya secara lebih mendalam.[1]

OpenGL (Open Graphics Library) adalah standar lintas platform untuk pemrograman grafis 2D dan 3D yang pertama kali dikembangkan oleh Silicon Graphics Inc. pada tahun 1992. Teknologi ini banyak digunakan dalam berbagai bidang, mulai dari pengembangan game, simulasi, hingga aplikasi edukasi.[2] Dengan fitur-fiturnya yang fleksibel dan performa tinggi, OpenGL memungkinkan pengguna untuk membuat model grafis dengan detail yang kompleks dan realistis. Selain itu, OpenGL juga didukung oleh berbagai bahasa pemrograman seperti C, C++, dan Python, sehingga mempermudah pengembang dalam mengimplementasikan grafis sesuai kebutuhan.[3]

Laporan ini berfokus pada proses pembuatan piramida suku Maya menggunakan OpenGL. Proyek ini diharapkan dapat menjadi referensi bagi pembelajaran lebih lanjut dalam bidang grafik komputer serta menjadi kontribusi untuk meningkatkan kesadaran akan pentingnya melestarikan kebudayaan dunia.

1.2 Rumusan Masalah

1. Apa yang dimkasud dengan OpenGL ?
2. Bagaimana membuat Objek 3D Piramida Suku Maya dalam OpenGL
3. Bagaimana membuat Objek 3D Pohon dalam OpenGL
4. Bagaimana membuat Objek 3D Matahari dalam OpenGL?

1.3 Tujuan

1. Mengethaui apa itu OpenGL
2. Mengetahui cara pembuatan Objek 3D Piramida Suku Maya dalam OpenGL.
3. Mengetahui cara pembuatan Objek 3D Pohon dalam OpenGL
4. Mengetahui cara pembuatan Objek 3D Matahari dalam OpenGL

BAB II
TINJAUAN PUSTAKA

2.1 OpenGL

OpenGL adalah suatu graphic library yang sebagian bersifat open source, dipakai pada banyak platform (windows, linux) dan dapat digunakan pada berbagai jenis compiler seperti C++ atau Delphi.

OpenGL bukanlah bahasa pemrograman tetapi merupakan suatu Application Programming Interface (API).[4]

2.2 Langkah-langkah dalam Pembuatan Objek 3D

2.2.1 Membuat Piramida Suku Maya

1. Menentukan Parameter Piramida
    - Tetapkan parameter dasar, lebar *(width)*, kedalaman *(depth)*, tinggi setiap tingkat *(height)*, faktor pengurangan ukuran tiap tingkat *(shrinkFactor)*, dan jumlah tingkat *(levels)*.

```
float baseWidth = 2.0f;    // Lebar dasar
float baseDepth = 2.0f;    // Kedalaman dasar
float height = 0.2f;       // Tinggi setiap tingkat
float shrinkFactor = 0.9f; // Pengurangan ukuran untuk setiap tingkat
int levels = 10;           // Jumlah tingkatan
```

Gambar 2.2. 1 Menentukan Paramter Piramida Suku Maya

2. Menggambar Setiap Tingkat Piramida
    - Gunakan perulangan untuk menggambar setiap tingkat piramida.
    - Pada setiap iterasi, posisi model ditranslasikan ke atas sesuai tinggi tingkat sebelumnya menggunakan *glTranslatef()*.
    - Gambar kotak untuk tingkat tersebut dengan fungsi *drawBox()* menggunakan parameter ukuran saat ini.
    - Kurangi ukuran dasar piramida *(width dan depth)* dengan faktor pengurangan *(shrinkFactor)*.

```
for (int i = 0; i < levels; i++) {
    glPushMatrix();

    // Naikkan posisi ke atas berdasarkan tingkatan
    glTranslatef(0.0f, i * height, 0.0f);

    // Gambar kotak dengan ukuran yang sesuai
    drawBox(baseWidth, height, baseDepth);

    // Kurangi ukuran dasar untuk tingkat berikutnya
    baseWidth *= shrinkFactor;
    baseDepth *= shrinkFactor;

    glPopMatrix();
}
```

Gambar 2.2. 2 Menentukan setiap tingkat piramida

3. Menggambar Balok Dekorasi pada Piramida
   - Tambahkan balok dekoratif di bagian depan, belakang, kiri, dan kanan piramida menggunakan fungsi *drawSingleBlock()*.
   - Tentukan posisi dan rotasi balok dengan kombinasi *glTranslatef()* dan *glRotatef()*.

```
//depan
glPushMatrix();
glTranslatef(0.0f, 0.80f, 0.35f);
glRotatef(70.0f, 1.0f, 0.0f, 0.0f);
drawSingleBlock();
glPopMatrix();
```

Gambar 2.2. 3 Menggambar Balok

4. Fungsi *drawBox()* untuk Membuat Kotak
   - Fungsi *drawBox()* digunakan untuk menggambar kotak dengan parameter lebar, tinggi, dan kedalaman.
   - Gunakan *glBegin(GL_QUADS)* untuk menggambar setiap sisi kotak.
   - Tetapkan warna untuk kotak menggunakan *glColor3ub()*.

```
void drawBox(float width, float height, float depth) {
    glBegin(GL_QUADS);
    glColor3ub(204, 200, 197);

    // Depan
    glVertex3f(-width / 2, 0, depth / 2);
    glVertex3f(width / 2, 0, depth / 2);
    glVertex3f(width / 2, height, depth / 2);
    glVertex3f(-width / 2, height, depth / 2);
```

Gambar 2.2. 4 Membuat Kotak

5. Fungsi *drawSingleBlock()* untuk Membuat Balok Dekorasi
   - Fungsi ini digunakan untuk menggambar balok dengan parameter dimensi tertentu.
   - Sebelum menggambar, translasi balok ke posisi yang sesuai menggunakan *glTranslatef().*

```
// membuat balok miring di tengah tembok piramida
void drawSingleBlock() {
    float blockWidth = 0.2f;  // Lebar balok
    float blockHeight = 0.3f; // Tinggi balok
    float blockDepth = 2.2f;  // Kedalaman balok

    // Posisikan balok di tempat tangga sebelumnya
    glTranslatef(0.0f, blockHeight / 2.0f, 0.0f);

    // Gambar balok
    drawBox(blockWidth, blockHeight, blockDepth);
}
```

Gambar 2.2. 5 Membuat balok dekorasi

6. Mengelola Transformasi dengan *glPushMatrix()* dan *glPopMatrix()*
   - Gunakan *glPushMatrix()* untuk menyimpan status transformasi saat ini sebelum menerapkan perubahan posisi atau rotasi.
   - Setelah menggambar, kembalikan ke status transformasi sebelumnya dengan *glPopMatrix().*
7. Memastikan Skalabilitas dengan Parameter scale
   - Skala keseluruhan piramida diatur dengan *glScalef(scale, scale, scale)* untuk memungkinkan pengguna mengatur ukuran akhir piramida sesuai kebutuhan.

```
glScalef(scale,scale,scale);
```

Gambar 2.2. 6 Menentukan skalabilitas

8. Integrasi dan Visualisasi Akhir
   - Semua langkah digabungkan ke dalam fungsi utama *drawPyramid().* Fungsi ini memanggil *drawBox()* untuk tiap tingkat dan *drawSingleBlock()* untuk menambahkan dekorasi.

2.2.2 Membuat Pohon

1. Membuat Fungsi drawTree untuk Menggambar Satu Pohon

   Menentukan Batang Pohon
   - Gunakan fungsi glPushMatrix() untuk menyimpan status transformasi.
   - Posisikan batang pohon dengan glTranslatef(x, y, z).
   - Tetapkan warna batang pohon menjadi cokelat menggunakan glColor3f(0.5f, 0.25f, 0.1f).

- Gunakan fungsi gluCylinder() untuk menggambar silinder sebagai batang pohon.
- Akhiri bagian batang pohon dengan glPopMatrix() untuk mengembalikan status transformasi awal.

Menentukan Daun Pohon

- Gunakan fungsi glPushMatrix() untuk menambahkan status transformasi baru.
- Posisikan bola daun di atas batang pohon dengan glTranslatef(x, y + 0.0f, z).
- Tetapkan warna bola daun menjadi hijau menggunakan glColor3f(0.0f, 1.0f, 0.0f).
- Gunakan fungsi glutSolidSphere() untuk menggambar bola sebagai daun pohon.
- Akhiri bagian daun pohon dengan glPopMatrix().

```
void drawTree(float x, float y, float z) {
    // Batang pohon
    glPushMatrix();
    glTranslatef(x, y, z);  // Posisi pohon
    glColor3f(0.5f, 0.25f, 0.1f);  // Warna batang pohon (coklat)

    // Rotasi batang agar tegak lurus
    glRotatef(90, 1.0f, 0.0f, 0.0f);  // Rotasi 90 derajat di sumbu X untuk membuat batang tegak

    GLUquadric *quadratic = gluNewQuadric();
    gluCylinder(quadratic, 0.1f, 0.1f, 1.0f, 32, 32);  // Membuat silinder untuk batang pohon
    glPopMatrix();

    // Daun pohon (bola hijau)
    glPushMatrix();
    glTranslatef(x, y + 0.0f, z);  // Posisikan bola daun di atas batang pohon
    glColor3f(0.0f, 1.0f, 0.0f);  // Warna daun pohon (hijau)
    glutSolidSphere(0.5f, 20, 20);  // Membuat bola sebagai daun pohon
    glPopMatrix();
}
```

Gambar 2.2. 7 Menggambar satu pohon

2. Membuat Fungsi drawForest untuk Menggambar Banyak Pohon
    - Fungsi *drawForest()* memanggil fungsi drawTree() beberapa kali dengan posisi yang berbeda.
    - Posisikan pohon-pohon secara manual dengan argumen posisi (x, y, z) yang bervariasi.

```
void drawForest() {
    // Posisi pohon-pohon yang berbeda
    drawTree(10.0f, 1.0f, 2.0f);  // Pohon pertama
    drawTree(5.0f, 2.0f, 2.0f);
    drawTree(3.0f, 3.0f, 2.0f);
    drawTree(-1.0f, 3.0f, 3.0f);
    drawTree(-2.0f, 2.0f, 3.0f);
    drawTree(-3.0f, 1.0f, 3.0f);
    drawTree(5.0f, 1.0f, 3.0f);
}
```

Gambar 2.2. 8 Menggambar banyak pohon

1.2.3 Membuat Matahari

1. Membuat Fungsi *updateLightingBasedOnSunPosition()*, fungsi ini mengatur pencahayaan berdasarkan posisi matahari
    - Posisi cahaya diatur mengikuti posisi matahari pada sumbu Y menggunakan array GLfloat Intensitas cahaya ditentukan oleh sunPositionY
    - Atur Komponen Cahaya

- Cahaya difus *(lightDiffuse)* : Memberikan efek warna keemasan pada cahaya.
- Cahaya spekular *(lightSpecular)* : Cahaya yang lebih terang untuk efek pantulan.
- Cahaya ambient *(lightAmbient)* : Menambah pencahayaan redup saat matahari lebih rendah.

- Gunakan glLightfv() untuk mengatur posisi dan intensitas cahaya.

```
// Menambahkan pengaruh matahari terhadap pencahayaan
void updateLightingBasedOnSunPosition() {
    GLfloat lightPosition[] = {5.0f, sunPositionY, 0.0f, 1.0f};  // Posisi cahaya mengikuti matahari
    GLfloat lightIntensity = sunPositionY / 5.0f; // Intensitas cahaya bergantung pada posisi matahari

    // Atur intensitas cahaya sesuai dengan ketinggian matahari
    GLfloat lightDiffuse[] = {lightIntensity, lightIntensity, 0.0f, 1.0f};  // Efek keemasan untuk matahari
    GLfloat lightSpecular[] = {lightIntensity, lightIntensity, 1.0f}; // Warna spekular lebih terang
    GLfloat lightAmbient[] = {lightIntensity * 0.2f, lightIntensity * 0.2f, lightIntensity * 0.2f, 1.0f};

    glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);  // Posisi cahaya
    glLightfv(GL_LIGHT0, GL_DIFFUSE, lightDiffuse);    // Intensitas cahaya difus
    glLightfv(GL_LIGHT0, GL_SPECULAR, lightSpecular);  // Intensitas cahaya spekular
    glLightfv(GL_LIGHT0, GL_AMBIENT, lightAmbient);    // Cahaya ambient
}
```

Gambar 2.2. 9 Mengatur pencahayaan matahari

2. Membuat *Fungsi updateSunPosition()*, fungsi ini bertanggung jawab untuk mengupdate posisi matahari secara dinamis
   - Matahari bergerak ke atas secara bertahap hingga mencapai posisi maksimal 5.0f
   - Panggil *updateLightingBasedOnSunPosition()* untuk memperbarui pencahayaan berdasarkan posisi matahari.
   - Gunakan *glutTimerFunc()* untuk mengatur fungsi dijalankan secara berulang setiap 16 milidetik

```
// Memepengaruhi posisi matahari
void updateSunPosition(int value) {
    if (sunPositionY < 5.0f) {
        // Gerakan matahari ke atas
        sunPositionY += 0.01f;
    }

    updateLightingBasedOnSunPosition();

    glutTimerFunc(16, updateSunPosition, 0);
}
```

Gambar 2.2. 10 Mengupdate posisi matahari secara dinamis

3. Membuat Fungsi *drawSun()*, fungsi ini bertanggung jawab untuk menggambar matahari di layar
   - Gunakan *glPushMatrix()* untuk menyimpan status transformasi.
   - Nonaktifkan pencahayaan agar matahari bersinar secara alami
   - Gunakan *glTranslatef()* untuk memindahkan matahari ke koordinat tertentu:
   - Menggambar Matahari

- Atur warna matahari menjadi kuning dengan *glColor3f(1.0, 1.0, 0.0).*
- Gunakan *glutSolidSphere()* untuk menggambar bola sebagai matahari:
- Tambahkan posisi matahari pada sumbu Y jika sunPositionY > 0.5:
- Gunakan *glEnable(GL_LIGHTING*) untuk mengaktifkan kembali pencahayaan global.
- Gunakan *glPopMatrix()* untuk mengembalikan status transformasi awal.

```
// Mmebuat Matahari
void drawSun() {
    glPushMatrix();
    glDisable(GL_LIGHTING); // Nonaktifkan pencahayaan agar matahari bersinar
    glTranslatef(2.2f, 1.5f, 3.0f);
    glColor3f(1.0, 1.0, 0.0);
    if (sunPositionY > 0.5) {
        glTranslatef(0.0f, sunPositionY, 0.0f);
        glutSolidSphere(0.5, 20, 20);
    } else if (sunPositionY < 0.0f) {
        sunPositionY = 0.0f;
    }
    glEnable(GL_LIGHTING); // Aktifkan kembali pencahayaan
    glPopMatrix();
}
```

Gambar 2.2. 11 Menggambar Matahari

2.3 Cara Kerja OpenGL

OpenGL bekerja sebagai sebuah Application Programming Interface (API) yang menyediakan fungsi-fungsi untuk menggambar grafik 2D dan 3D. Prosesnya dimulai dengan inisialisasi lingkungan grafik, seperti pengaturan viewport, buffer, dan matriks awal. Setelah itu, OpenGL memanfaatkan pipeline grafis untuk mengolah data objek menjadi tampilan visual.

Pertama, data geometri dari objek didefinisikan menggunakan fungsi-fungsi OpenGL, seperti *glBegin()* dan *glEnd(),* yang mencakup koordinat vertikal, warna, tekstur, atau properti material lainnya. Data ini diproses oleh matriks transformasi, termasuk model-view matrix untuk pengaturan posisi, rotasi, dan skala objek, serta projection matrix untuk menentukan perspektif tampilan.

Pencahayaan diterapkan menggunakan model pencahayaan Phong yang mencakup komponen ambient, diffuse, dan specular. OpenGL juga mengatur properti material objek, sehingga objek dapat berinteraksi dengan cahaya untuk menciptakan efek yang realistis. Setelah itu, data diolah lebih lanjut dalam buffer, di mana OpenGL menggunakan teknik double buffering. Teknik ini menggambar objek terlebih dahulu di buffer belakang sebelum menampilkannya ke layar, untuk memastikan hasil gambar terlihat halus tanpa flickering.

Interaksi pengguna, seperti input dari mouse atau keyboard, diintegrasikan melalui library pendukung seperti GLUT atau GLFW. Dengan fungsi seperti *glutIdleFunc()* dan *glutDisplayFunc(),* program dapat merespons input dan terus memperbarui tampilan secara dinamis.

OpenGL juga menerapkan optimasi, seperti frustum culling, untuk meningkatkan performa dengan hanya menggambar objek yang berada dalam area pandangan kamera. Setelah semua langkah selesai, pipeline grafis menghasilkan gambar akhir yang ditampilkan di layar.

2.4 Piramida Suku Maya Di OpenGL

Piramida Suku Maya di OpenGL dibuat melalui serangkaian langkah yang melibatkan transformasi geometri dan pengaturan visualisasi. Pertama, parameter dasar piramida ditentukan, seperti lebar, kedalaman, tinggi tiap tingkat, jumlah tingkat, dan faktor pengurangan ukuran. Fungsi *drawBox()* digunakan untuk menggambar setiap tingkat piramida, dengan koordinat yang disesuaikan untuk membentuk bentuk kotak.

Tiap tingkat piramida digambar menggunakan perulangan. Setiap iterasi memindahkan posisi piramida ke atas dengan *glTranslatef(),* menggambar kotak, dan mengurangi ukuran lebar serta kedalaman piramida dengan faktor pengurangan yang telah ditentukan. Untuk menambah estetika, balok dekorasi ditempatkan di keempat sisi piramida menggunakan fungsi *drawSingleBlock(),* yang mengatur posisi dan rotasi balok dengan *glTranslatef()* dan *glRotatef().*

Pengelolaan transformasi dilakukan dengan *glPushMatrix()* dan *glPopMatrix()* untuk memastikan bahwa setiap perubahan posisi atau rotasi tidak memengaruhi bagian lain dari objek. Selanjutnya, fungsi *drawPyramid()* mengintegrasikan semua langkah tersebut, termasuk penambahan skala dengan glScalef() agar ukuran piramida dapat disesuaikan sesuai kebutuhan.

Selain piramida, objek lain seperti pohon dan matahari juga dapat ditambahkan untuk melengkapi pemandangan. Pohon dibuat menggunakan fungsi *drawTree(),* yang terdiri dari batang berbentuk silinder *(dengan gluCylinder())* dan bola daun *(dengan glutSolidSphere()).* Posisi pohon diatur sedemikian rupa untuk memberikan nuansa alami di sekitar piramida.

Matahari digambar menggunakan fungsi *drawSun(),* yang memanfaatkan *glutSolidSphere()* untuk membentuk bola bercahaya. Posisi matahari pada sumbu Y diatur secara dinamis oleh fungsi *updateSunPosition(),* menciptakan efek pencahayaan realistis. Fungsi pencahayaan seperti *glLightfv()* digunakan untuk menentukan intensitas dan warna cahaya, memberikan kesan waktu siang atau sore yang dramatis di sekitar piramida.

Dengan menggabungkan piramida, pohon, dan matahari, visualisasi di OpenGL menjadi lebih hidup dan memberikan kesan lingkungan yang lebih realistis dan menarik.

BAB III
HASIL

3.1 Source Code

```
4   #include <GL/glut.h>
5   #include <math.h>
6   #include <cstdlib>
7   #include <ctime>
8
9   // Camera position and orientation
10  float camX = -3.0f, camY = 3.5f, camZ = -5.5f;
11  float yaw = 28.0f, pitch = -40.0f;
12  float lastX = 400, lastY = 300;
13  bool firstMouse = true;
14
15  // Camera direction
16  float dirX = 0.0f, dirY = 0.0f, dirZ = 0.0f;
17
18  int windowWidth = 800;
19  int windowHeight = 600;
20  int savedWindowWidth = 800;
21  int savedWindowHeight = 600;
22  int savedWindowPosX = 100;
23  int savedWindowPosY = 100;
24  bool isFullscreen = false;
25
26  const int NUM_STEPS = 20;
27  const float STEP_WIDTH = 1.0f;
28  const float STEP_HEIGHT = 0.2f;
29  const float STEP_DEPTH = 2.0f;
30  float rotateX = 0.0f; // Rotasi pada sumbu X
31  float rotateY = 0.0f; // Rotasi pada sumbu Y
32  float cloudPosition = -10.0f;
33  void  drawSingleBlock();
34
35  bool showAxis = false;
```

```
36  void hiddenCarte();
37
38  void drawCoordinate() {
39      if (!showAxis) return;
40      glLineWidth(2.0f);
41
42      // Sumbu X (Merah)
43      glColor3f(1.0f, 0.0f, 0.0f);
44      glBegin(GL_LINES);
45      glVertex3f(-10000.0f, 0.0f, 0.0f);
46      glVertex3f(10000.0f, 0.0f, 0.0f);
47      glEnd();
48
49      // Sumbu Y (Hijau)
50      glColor3f(0.0f, 1.0f, 0.0f);
51      glBegin(GL_LINES);
52      glVertex3f(0.0f, -10000.0f, 0.0f);
53      glVertex3f(0.0f, 10000.0f, 0.0f);
54      glEnd();
55
56      // Sumbu Z (Biru)
57      glColor3f(0.0f, 0.0f, 1.0f);
58      glBegin(GL_LINES);
59      glVertex3f(0.0f, 0.0f, -10000.0f);
60      glVertex3f(0.0f, 0.0f, 10000.0f);
61      glEnd();
62  }
63
64  // variabel untuk posisi matahari
65  // posisi awal matahari di bawah layar
66  float sunPositionY = -1.0f;
67
68  float scale = 1.0f;
69
70  void toggleFullscreen() {
71      isFullscreen = !isFullscreen;
72      if (isFullscreen) {
73          // Save current window position and size
```

```
74          savedWindowPosX = glutGet(GLUT_WINDOW_X);
75          savedWindowPosY = glutGet(GLUT_WINDOW_Y);
76          savedWindowWidth = glutGet(GLUT_WINDOW_WIDTH);
77          savedWindowHeight = glutGet(GLUT_WINDOW_HEIGHT);
78
79          // Switch to fullscreen
80          glutFullScreen();
81      } else {
82          // Restore window position and size
83          glutReshapeWindow(savedWindowWidth,
    savedWindowHeight);
84          glutPositionWindow(savedWindowPosX, savedWindowPosY);
85      }
86  }
87
88  void init() {
89      glEnable(GL_DEPTH_TEST);
90      glEnable(GL_COLOR_MATERIAL);
91      glutSetCursor(GLUT_CURSOR_NONE);
92      glEnable(GL_LIGHTING);
93      glEnable(GL_LIGHT0);
94
95      glClearColor(0.5, 0.7, 1.0, 1.0); // Warna langit
96      glMatrixMode(GL_PROJECTION);
97      glLoadIdentity();
98      gluPerspective(45.0, 1.33, 0.1, 10000.0);
99      glMatrixMode(GL_MODELVIEW);
100 }
101
102 // Membuat Objek Piramida Oleh Karina
103 void drawBox(float width, float height, float depth) {
104     glBegin(GL_QUADS);
105     glColor3ub(204, 200, 197);
106
107     // Depan
108     glVertex3f(-width / 2, 0, depth / 2);
109     glVertex3f(width / 2, 0, depth / 2);
110     glVertex3f(width / 2, height, depth / 2);
```

```
111     glVertex3f(-width / 2, height, depth / 2);
112
113     // Belakang
114     glVertex3f(-width / 2, 0, -depth / 2);
115     glVertex3f(width / 2, 0, -depth / 2);
116     glVertex3f(width / 2, height, -depth / 2);
117     glVertex3f(-width / 2, height, -depth / 2);
118
119     // Kiri
120     glVertex3f(-width / 2, 0, -depth / 2);
121     glVertex3f(-width / 2, 0, depth / 2);
122     glVertex3f(-width / 2, height, depth / 2);
123     glVertex3f(-width / 2, height, -depth / 2);
124
125     // Kanan
126     glVertex3f(width / 2, 0, -depth / 2);
127     glVertex3f(width / 2, 0, depth / 2);
128     glVertex3f(width / 2, height, depth / 2);
129     glVertex3f(width / 2, height, -depth / 2);
130
131     glColor3f(0.6f, 0.4f, 0.2);  // Warna merah
132  // Atas
133     glVertex3f(-width / 2, height, -depth / 2);
134     glVertex3f(width / 2, height, -depth / 2);
135     glVertex3f(width / 2, height, depth / 2);
136     glVertex3f(-width / 2, height, depth / 2);
137
138     // Bawah
139     glVertex3f(-width / 2, 0, -depth / 2);
140     glVertex3f(width / 2, 0, -depth / 2);
141     glVertex3f(width / 2, 0, depth / 2);
142     glVertex3f(-width / 2, 0, depth / 2);
143
144     glEnd();
145 }
146
147 void drawPyramid(float scale) {
148     float baseWidth = 2.0f;   // Lebar dasar
```

```
149     float baseDepth = 2.0f;   // Kedalaman dasar
150     float height = 0.2f;      // Tinggi setiap tingkat
151     float shrinkFactor = 0.9f; // Pengurangan ukuran untuk
    setiap tingkat
152     int levels = 10;          // Jumlah tingkatan
153
154
155  glPushMatrix();
156  glScalef(scale,scale,scale);
157     for (int i = 0; i < levels; i++) {
158         glPushMatrix();
159
160         // Naikkan posisi ke atas berdasarkan tingkatan
161         glTranslatef(0.0f, i * height, 0.0f);
162
163         // Gambar kotak dengan ukuran yang sesuai
164         drawBox(baseWidth, height, baseDepth);
165
166         // Kurangi ukuran dasar untuk tingkat berikutnya
167         baseWidth *= shrinkFactor;
168         baseDepth *= shrinkFactor;
169
170         glPopMatrix();
171     }
172
173     //depan
174  glPushMatrix();
175     glTranslatef(0.0f, 0.80f, 0.35f);
176     glRotatef(70.0f, 1.0f, 0.0f, 0.0f);
177     drawSingleBlock();
178     glPopMatrix();
179
180     // Belakang
181     glPushMatrix();
182     glTranslatef(0.0f, 0.80f, -0.35f);
183     glRotatef(-70.0f, 1.0f, 0.0f, 0.0f);
184     drawSingleBlock();
185     glPopMatrix();
```

```
186
187  // Kiri
188      glPushMatrix();
189      glTranslatef(0.35f, 0.80f, 0.0f);
190      glRotatef(90.0f, 0.0f, 1.0f, 0.0f);
191      glRotatef(70.0f, 1.0f, 0.0f, 0.0f);
192      drawSingleBlock();
193      glPopMatrix();
194
195  // Kanan
196      glPushMatrix();
197      glTranslatef(-0.35f, 0.80f, 0.0f);
198      glRotatef(-90.0f, 0.0f, 1.0f, 0.0f);
199      glRotatef(70.0f, 1.0f, 0.0f, 0.0f);
200      drawSingleBlock();
201      glPopMatrix();
202
203      glPopMatrix();
204 }
205
206 // membuat balok miring di tengah tembok piramida
207 void drawSingleBlock() {
208      float blockWidth = 0.2f;  // Lebar balok
209      float blockHeight = 0.3f; // Tinggi balok
210      float blockDepth = 2.2f;  // Kedalaman balok
211
212      // Posisikan balok di tempat tangga sebelumnya
213      glTranslatef(0.0f, blockHeight / 2.0f, 0.0f);
214
215      // Gambar balok
216      drawBox(blockWidth, blockHeight, blockDepth);
217 }
218
219 void lantai(){
220  glPushMatrix();
221  glEnable(GL_DEPTH_TEST);
222  glColor3f(0.0f, 0.5f, 0.0f);
223  glTranslated(0,-1.0,0);
```

```
224  glScaled(1000, 0.5, 1000);
225  glutSolidSphere(1,30,30);
226  glPopMatrix();
227 }
228
229 // Membuat Matahari oleh Rivaldi
230 // Menambahkan pengaruh matahari terhadap pencahayaan
231 void updateLightingBasedOnSunPosition() {
232     GLfloat lightPosition[] = {5.0f, sunPositionY, 0.0f,
    1.0f};  // Posisi cahaya mengikuti matahari
233     GLfloat lightIntensity = sunPositionY / 5.0f; //
    Intensitas cahaya bergantung pada posisi matahari
234
235     // Atur intensitas cahaya sesuai dengan ketinggian
    matahari
236     GLfloat lightDiffuse[] = {lightIntensity, lightIntensity,
    0.0f, 1.0f};  // Efek keemasan untuk matahari
237     GLfloat lightSpecular[] = {lightIntensity, lightIntensity,
    1.0f}; // Warna spekular lebih terang
238     GLfloat lightAmbient[] = {lightIntensity * 0.2f,
    lightIntensity * 0.2f, lightIntensity * 0.2f, 1.0f};  // Efek
    reduksi cahaya saat matahari di bawah
239
240     glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);  //
    Posisi cahaya
241     glLightfv(GL_LIGHT0, GL_DIFFUSE, lightDiffuse);    //
    Intensitas cahaya difus
242     glLightfv(GL_LIGHT0, GL_SPECULAR, lightSpecular);  //
    Intensitas cahaya spekular
243     glLightfv(GL_LIGHT0, GL_AMBIENT, lightAmbient);    //
    Cahaya ambient
244 }
245
246 // Memepengaruhi posisi matahari
247 void updateSunPosition(int value) {
248  if (sunPositionY < 5.0f) {
249        // Gerakan matahari ke atas
250        sunPositionY += 0.01f;
```

```
251   }
252
253   updateLightingBasedOnSunPosition();
254
255   glutTimerFunc(16, updateSunPosition, 0);
256  }
257
258  // Mmebuat Matahari
259  void drawSun() {
260   glPushMatrix();
261   glDisable(GL_LIGHTING); // Nonaktifkan pencahayaan agar
     matahari bersinar
262   glTranslatef(2.2f, 1.5f, 3.0f);
263   glColor3f(1.0, 1.0, 0.0);
264   if (sunPositionY > 0.5) {
265         glTranslatef(0.0f, sunPositionY, 0.0f);
266         glutSolidSphere(0.5, 20, 20);
267   } else if (sunPositionY < 0.0f) {
268         sunPositionY = 0.0f;
269   }
270   glEnable(GL_LIGHTING); // Aktifkan kembali pencahayaan
271   glPopMatrix();
272  }
273
274
275  // Mmebuat Pohon oleh Wildan
276  void drawTree(float x, float y, float z) {
277      // Batang pohon
278      glPushMatrix();
279      glTranslatef(x, y, z);  // Posisi pohon
280      glColor3f(0.5f, 0.25f, 0.1f);  // Warna batang pohon
     (coklat)
281
282      // Rotasi batang agar tegak lurus
283      glRotatef(90, 1.0f, 0.0f, 0.0f);  // Rotasi 90 derajat di
     sumbu X untuk membuat batang tegak
284
285      GLUquadric *quadratic = gluNewQuadric();
```

```
286     gluCylinder(quadratic, 0.1f, 0.1f, 1.0f, 32, 32);  //
    Membuat silinder untuk batang pohon
287     glPopMatrix();
288
289     // Daun pohon (bola hijau)
290     glPushMatrix();
291     glTranslatef(x, y + 0.0f, z);  // Posisikan bola daun di
    atas batang pohon
292     glColor3f(0.0f, 1.0f, 0.0f);  // Warna daun pohon (hijau)
293     glutSolidSphere(0.5f, 20, 20);  // Membuat bola sebagai
    daun pohon
294     glPopMatrix();
295 }
296
297 void drawForest() {
298     // Posisi pohon-pohon yang berbeda
299     drawTree(10.0f, 1.0f, 2.0f);  // Pohon pertama
300     drawTree(5.0f, 2.0f, 2.0f);
301     drawTree(3.0f, 3.0f, 2.0f);
302     drawTree(-1.0f, 3.0f, 3.0f);
303     drawTree(-2.0f, 2.0f, 3.0f);
304     drawTree(-3.0f, 1.0f, 3.0f);
305     drawTree(5.0f, 1.0f, 3.0f);
306 }
307
308
309 void reshape(int w, int h) {
310      windowWidth = w;
311      windowHeight = h;
312      glViewport(0, 0, w, h);
313 }
314
315 void display() {
316     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
317  glMatrixMode(GL_PROJECTION);
318     glLoadIdentity();
319     gluPerspective(70.0f, windowWidth / windowHeight, 0.1f,
    10000000.0f);
```

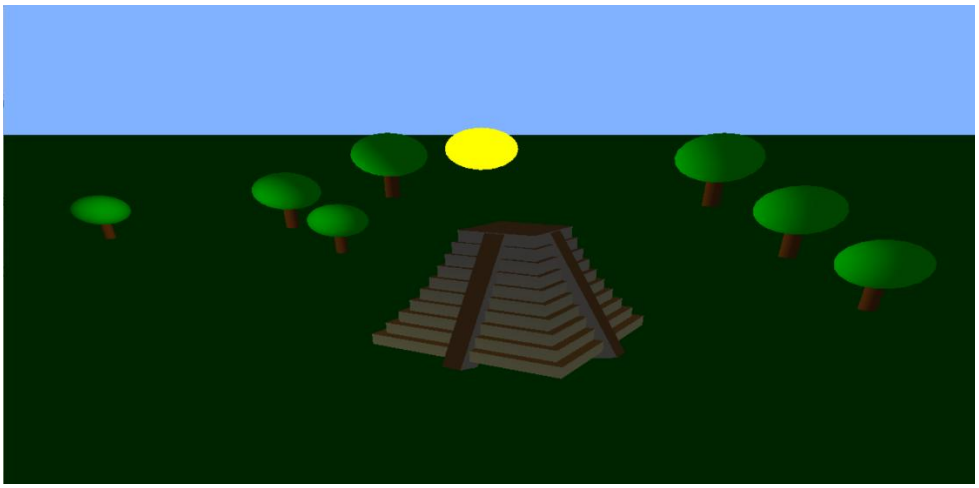
```
320     gluLookAt(camX, camY, camZ,
321               camX + dirX, camY + dirY, camZ + dirZ,
322               0.0f, 1.0f, 0.0f);
323     glMatrixMode(GL_MODELVIEW);
324 
325     // Terapkan rotasi berdasarkan input mouse
326     glRotatef(rotateX, 1.0f, 0.0f, 0.0f); // Rotasi pada sumbu
    X
327     glRotatef(rotateY, 0.0f, 1.0f, 0.0f); // Rotasi pada sumbu
    Y
328 
329 
330  drawCoordinate();
331 
332     // Gambar alas piramida lebih besar dari piramida
333  lantai();
334 
335     // matahari
336     drawSun();
337 
338     // piramida
339     drawPyramid(1.0f);
340 
341  // Pohon
342     drawForest();
343 
344 glPopMatrix();
345 glutSwapBuffers();
346 
347 }
348 
349 void keyboard(unsigned char key, int x, int y) {
350     float speed = 0.1f;
351 
352     switch (key) {
353         case 'w':
354             camX += dirX * speed;
355             camZ += dirZ * speed;
```

```
356             break;
357         case 's':
358             camX -= dirX * speed;
359             camZ -= dirZ * speed;
360             break;
361         case 'a':
362             camX += dirZ * speed;
363             camZ -= dirX * speed;
364             break;
365         case 'd':
366             camX -= dirZ * speed;
367             camZ += dirX * speed;
368             break;
369         case 'f': // Tombol F untuk toggle fullscreen
370             toggleFullscreen();
371             break;
372         case 'c':
373             showAxis = !showAxis;
374             break;
375         case 27:  // ESC key
376             if (isFullscreen) {
377                 toggleFullscreen(); // Keluar dari fullscreen
    dulu
378             } else {
379                 exit(0); // Keluar program jika sudah dalam
    mode window
380             }
381             break;
382     }
383     glutPostRedisplay();
384 }
385
386 void hiddenCarte() {
387  if (showAxis) {
388        void drawCoordinate();
389  }
390 }
391
```

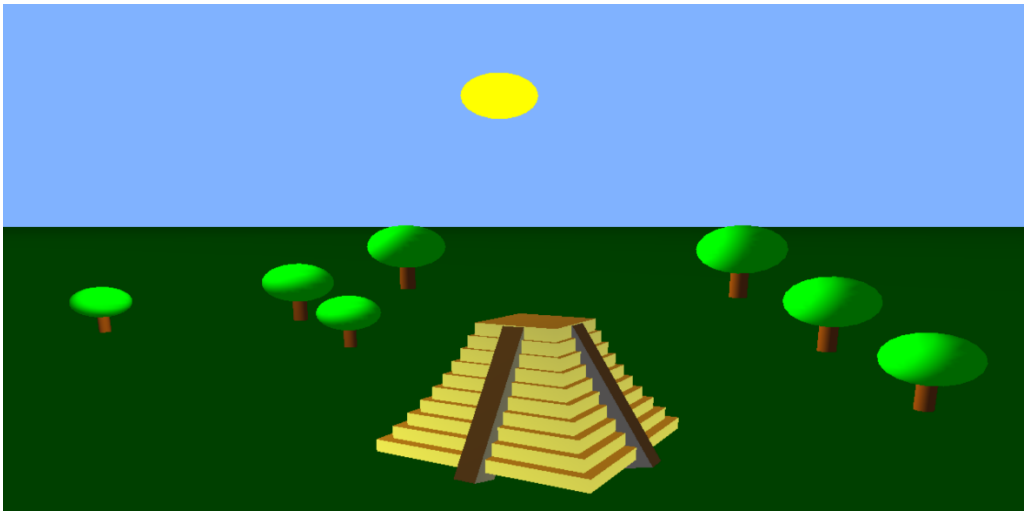
```
392 void mouseMove(int x, int y) {
393     float xoffset = x - windowWidth/2;
394     float yoffset = windowHeight/2 - y;
395 
396     float sensitivity = 0.1f;
397     xoffset *= sensitivity;
398     yoffset *= sensitivity;
399 
400     yaw += xoffset;
401     pitch += yoffset;
402 
403     if (pitch > 89.0f)
404         pitch = 89.0f;
405     if (pitch < -89.0f)
406         pitch = -89.0f;
407 
408     dirX = cos(yaw * M_PI / 180.0f) * cos(pitch * M_PI /
    180.0f);
409     dirY = sin(pitch * M_PI / 180.0f);
410     dirZ = sin(yaw * M_PI / 180.0f) * cos(pitch * M_PI /
    180.0f);
411 
412     // Reset mouse position to center of window
413     glutWarpPointer(windowWidth/2, windowHeight/2);
414 
415     glutPostRedisplay();
416 }
417 
418 int main(int argc, char** argv) {
419     glutInit(&argc, argv);
420     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
421 
422      // Get screen resolution
423     windowWidth = glutGet(GLUT_SCREEN_WIDTH);
424     windowHeight = glutGet(GLUT_SCREEN_HEIGHT);
425 
426     glutInitWindowSize(windowWidth, windowHeight);
427     glutInitWindowPosition(0, 0);
```

```
428     glutCreateWindow("Tangga OpenGL");
429
430     init();
431
432     // Mulai timer untuk update posisi matahari
433     glutTimerFunc(25, updateSunPosition, 0);
434
435     glutDisplayFunc(display);
436     glutReshapeFunc(reshape);
437     glutKeyboardFunc(keyboard);
438     glutPassiveMotionFunc(mouseMove);
439
440     glutMainLoop();
441     return 0;
442 }
```

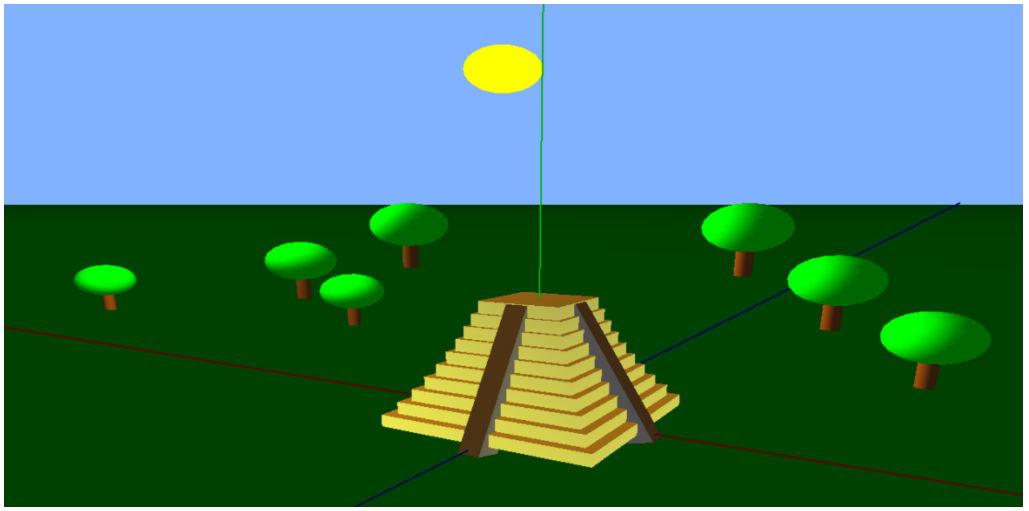
3.2 Output



Gambar 3.2 1 Output sebelum matahari terbit



Gambar 3.2 2 Output sesudah matahari terbit



Gambar 3.2 3 Output keitka menampilkan koordinat kartesius

3.3 Penjelasan

1. Kontrol Kamera
   - Posisi dan orientasi kamera dikontrol oleh variabel camX, camY, camZ, serta sudut yaw dan pitch.
   - Logika untuk mengontrol kamera dapat disesuaikan untuk navigasi dalam dunia 3D.
2. Fitur Piramida
   - Fungsi drawPyramid digunakan untuk menggambar piramida bertingkat.
   - Piramida dibangun dengan beberapa tingkat, di mana setiap tingkat ukurannya menyusut sesuai dengan shrinkFactor.
   - Tambahan "balok miring" di sisi piramida memperkaya detail visual.

3. Lantai
    - Fungsi lantai menggambar lantai hijau besar menggunakan glutSolidSphere yang diratakan.
    - Memberikan dasar untuk struktur piramida dan menciptakan suasana lingkungan.
4. Koordinat Sumbu
    - Fitur untuk menampilkan sumbu koordinat X, Y, dan Z dapat diaktifkan menggunakan boolean showAxis.
    - Berguna untuk debugging dan orientasi objek dalam ruang 3D.
5. Simulasi Matahari
    - Posisi matahari dikendalikan oleh variabel sunPositionY.
    - Fungsi updateLightingBasedOnSunPosition mengatur pencahayaan berdasarkan posisi matahari, termasuk efek difus dan ambient.
    - Memberikan efek pencahayaan realistis yang berubah sesuai waktu.
6. Transisi Layar Penuh
    - Fungsi toggleFullscreen memungkinkan beralih antara mode layar penuh dan mode jendela.
    - Menyimpan dan mengembalikan posisi serta ukuran jendela secara otomatis.
7. Pencahayaan OpenGL
    - Cahaya global diaktifkan menggunakan GL_LIGHTING dan GL_LIGHT0.
    - Intensitas dan posisi cahaya berubah sesuai dengan posisi matahari, menciptakan efek pencahayaan dinamis.
8. Fungsi drawBox
    - Membuat kotak (balok) 3D dengan warna berbeda untuk setiap sisi.
    - Digunakan sebagai komponen utama dalam membangun struktur piramida dan balok tambahan.
9. Interaktivitas
    - Kamera, matahari, dan objek dapat dipindahkan atau dirotasi untuk eksplorasi lebih interaktif.
10. Optimalisasi
    - Menggunakan glPushMatrix dan glPopMatrix untuk memastikan transformasi tidak saling memengaruhi antara objek.
    - Penerapan glEnable(GL_DEPTH_TEST) memastikan objek lebih jauh tidak menimpa objek lebih dekat.

BAB IV

4.1. Kesimpulan

Pada proyek ini, OpenGL berhasil digunakan sebagai library grafis untuk membuat visualisasi objek 3D seperti piramida Suku Maya, pohon, dan matahari. Implementasi ini menunjukkan fleksibilitas dan kekuatan OpenGL dalam menciptakan elemen grafis yang kompleks dan realistis.

Pembuatan piramida Suku Maya mencerminkan kemampuan OpenGL dalam mengelola transformasi objek, seperti translasi, rotasi, dan skala, yang digabungkan untuk menghasilkan struktur bertingkat. Selain itu, elemen dekoratif yang ditambahkan pada piramida memperkaya visualisasi dan meningkatkan daya tarik estetika objek.

Objek pendukung seperti pohon dan matahari berhasil melengkapi pemandangan. Pohon memberikan elemen natural dengan kombinasi batang dan daun yang disusun menggunakan fungsi OpenGL seperti gluCylinder() dan glutSolidSphere(). Matahari, dengan pencahayaan dinamis yang diatur oleh fungsi seperti glLightfv(), menambah kesan realistis dengan efek pencahayaan yang berubah berdasarkan posisi.

Dengan pendekatan ini, proyek mampu menunjukkan cara kerja OpenGL yang efisien, mulai dari pengelolaan geometri hingga pencahayaan dinamis. Hal ini membuktikan bahwa OpenGL adalah pilihan yang ideal untuk membangun aplikasi grafis interaktif dan edukatif. Di masa mendatang, penggunaan OpenGL dapat diperluas ke proyek lain yang lebih kompleks, seperti simulasi lingkungan atau aplikasi berbasis virtual reality.

DAFTAR PUSTAKA

[1] E. A. P.A *et al.*, “Jejak Peradaban Kuno di Amerika: Mengenal Suku Maya, Aztec, dan Inca,” *Buana J. Geogr. Ekol. dan Kebencanaan*, vol. 1, no. 2, pp. 73–78, 2024, doi: 10.56211/buana.v1i2.520.

[2] J. Widadi and A. Dahlan Jl ProfDrSoepomo, “Media Pembelajaran Materi Pengenalan Opengl Pada Mata Kuliah Grafika Komputer,” *J. Sarj. Tek. Inform.*, vol. 6, no. 1, pp. 47–53, 2018, [Online]. Available: http://journal.uad.ac.id/index.php/JSTIF

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

[4] H. Karisma, “Pengenalan OpenGL,” *Unikom Repos.*, 2013, [Online]. Available: https://repository.unikom.ac.id/41687/1/Pengenalan OpenGL di DEVC%2B%2B.pdf