PRAKTIKUM GRAFIKA KOMPUTER

Pertemuan ke-5 (*Pemodelan 3D dengan Jaring Poligon*)

Ertansyah Rizal Priadi Sumarna
2006008

Jurusan Ilmu Komputer
Program Studi Teknik Informatika
Institut Teknologi Garut
Jl. Mayor Syamsu No. 1 Jayaraga Garut 44151 Indonesia

I. PENDAHULUAN

Pemodelan objek tiga dimensi (3D) merupakan aspek yang kunci dalam pengembangan grafika komputer. Pemodelan 3D memungkinkan kita untuk membuat objek yang kompleks dan realistis, seperti kubus, bola, kendaraan, bahkan manusia. Salah satu pendekatan yang digunakan dalam pemodelan 3D adalah dengan menggunakan jaring poligon (polygon meshes). Dalam bab ini, kita akan menjelaskan konsep dasar tentang pembentukan objek 3D dengan jaring poligon, dan bagaimana metode ini dapat digunakan untuk menciptakan objek yang kompleks.

Membuat Objek Mudah dan Kompleks dengan Jaring Poligon:** Kami akan menggali teknik untuk membentuk objek 3D menggunakan jaring poligon. Ini mencakup cara mendefinisikan titik-titik dan permukaan objek dengan cara yang memungkinkan kita menghasilkan objek yang kompleks dan detail.

Merender Jaring Poligon sebagai Wireframe dan/atau Pejal dengan OpenGL:** Selain pembentukan objek, kami juga akan membahas bagaimana Merender jaring poligon ini dengan bantuan OpenGL. Ini termasuk cara menggambarkan objek sebagai wireframe (kawat) atau objek pejal dengan pencahayaan.

Penting untuk memahami konsep jaring poligon. Jaring poligon adalah representasi objek 3D dengan menggunakan segitiga atau banyak segi lebih. Anda dapat membayangkan jaring ini sebagai rangkaian titik yang dihubungkan satu sama lain untuk membentuk objek tertentu. Sebagai contoh, untuk menggambarkan kubus, kita dapat membuat satu daftar yang berisi daftar masing-masing poligon, lokasi titik, dan normal untuk masing-masing titik. Namun, ada pendekatan yang lebih efisien yang melibatkan tiga daftar terpisah: daftar titik, daftar permukaan, dan daftar normal.

Daftar Titik: Daftar ini berisi koordinat titik-titik yang membentuk objek. Sebagai contoh, untuk kubus, kita memiliki 8 titik yang mewakili sudut-sudut kubus.

Daftar Permukaan: Daftar ini berisi indeks titik-titik yang membentuk permukaan objek. Dengan menggunakan daftar ini, kita dapat menggambarkan bagaimana titik-titik di daftar titik dihubungkan untuk membentuk permukaan objek.

Daftar Normal: Daftar ini berisi vektor normal untuk setiap permukaan objek. Vektor normal digunakan untuk menghitung pencahayaan dan shading pada permukaan objek.

Selain itu, kita juga perlu memahami konsep lintasan muka. Lintasan muka digunakan untuk menentukan bagian dalam dan luar objek serta arah normal. Ini adalah langkah penting dalam perhitungan normal permukaan.

Mengapa perhitungan normal penting dalam grafika komputer? Vektor normal memberi tahu kita letak permukaan bagian luar objek, digunakan untuk menghitung seberapa banyak sinar cahaya yang mengenai permukaan tersebut, dan menentukan seberapa halus atau kasar tampilan permukaan saat dirender.

Dengan pemahaman tentang konsep dasar ini, kita dapat lebih lanjut menjelajahi implementasi dan penerapan jaring poligon dalam pemodelan objek 3D. Melalui laporan ini, kami akan membahas langkah-langkah praktis, mengenai bagaimana membuat dan Merender objek 3D dengan jaring poligon menggunakan OpenGL, serta implikasi pentingnya dalam pengembangan grafika komputer.

II. PEMBAHASAN

A. Source Code

1. (1) titik (2) normal (3) permukaan:
a) Prisma;
b) Antiprisma;
c) Tetrahedron;
d) Octahedron;
e) Dodecahedron;

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

GLfloat light_diffuse[] = {1.0, 0.0, 0.0, 1.0};
GLfloat light_position[] = {1.0, 1.0, 1.0, 0.0};

void hitungNormal(float *normal, float *p1, float *p2, float *p3) {
    float v1[3], v2[3];

    // Hitung vektor dari p1 ke p2
    for (int i = 0; i < 3; i++) {
        v1[i] = p2[i] - p1[i];
    }

    // Hitung vektor dari p1 ke p3
    for (int i = 0; i < 3; i++) {
        v2[i] = p3[i] - p1[i];
    }

    // Hitung vektor normal sebagai hasil cross product v1 x v2
    normal[0] = v1[1] * v2[2] - v1[2] * v2[1];
    normal[1] = v1[2] * v2[0] - v1[0] * v2[2];
    normal[2] = v1[0] * v2[1] - v1[1] * v2[0];

    // Normalisasi vektor normal
    float length = std::sqrt(normal[0] * normal[0] + normal[1] *
normal[1] + normal[2] * normal[2]);
    for (int i = 0; i < 3; i++) {
        normal[i] /= length;
    }
}

void gambarPrisma(void)
{
    // Koordinat titik-titik prisma
    GLfloat vertices[8][3] = {
        {0.0, 0.0, 0.0},
        {1.0, 0.0, 0.0},
        {1.0, 1.0, 0.0},
        {0.0, 1.0, 0.0},
```

```
        {0.0, 0.0, 1.0},
        {1.0, 0.0, 1.0},
        {1.0, 1.0, 1.0},
        {0.0, 1.0, 1.0}
    };

    // Indeks permukaan prisma
    int surfaces[6][4] = {
        {0, 1, 2, 3},  // Permukaan alas
        {4, 5, 6, 7},  // Permukaan atas
        {0, 1, 5, 4},
        {1, 2, 6, 5},
        {2, 3, 7, 6},
        {3, 0, 4, 7}
    };

    // Menggambar prisma
    glBegin(GL_QUADS);
    for (int i = 0; i < 6; i++) {
        float normal[3];
        hitungNormal(normal, vertices[surfaces[i][0]],
vertices[surfaces[i][1]], vertices[surfaces[i][2]]);
        glNormal3fv(normal);

        for (int j = 0; j < 4; j++) {
            glVertex3fv(vertices[surfaces[i][j]]);
        }
    }
    glEnd();
}

void gambarAntiprisma(void)
{
    // Koordinat titik-titik antiprisma
    GLfloat vertices[10][3] = {
        {0.0, 0.0, 1.0},
        {1.0, 0.0, 0.0},
        {1.0, 0.0, 1.0},
        {0.0, 0.0, 0.0},
        {0.5, 1.0, 0.5},
        {0.0, 0.0, 1.0},
        {0.0, 0.0, 0.0},
        {1.0, 0.0, 1.0},
        {1.0, 0.0, 0.0},
        {0.5, 1.0, 0.5}
    };

    // Indeks permukaan antiprisma
    int surfaces[7][3] = {
        {0, 1, 2},
        {0, 2, 3},
        {4, 5, 6},
        {4, 7, 8},
        {4, 5, 7},
```

```
        {5, 6, 8},
        {6, 7, 8}
    };

    // Menggambar antiprisma
    glBegin(GL_TRIANGLES);
    for (int i = 0; i < 7; i++) {
        float normal[3];
        hitungNormal(normal, vertices[surfaces[i][0]],
vertices[surfaces[i][1]], vertices[surfaces[i][2]]);
        glNormal3fv(normal);

        for (int j = 0; j < 3; j++) {
            glVertex3fv(vertices[surfaces[i][j]]);
        }
    }
    glEnd();
}

void gambar3D(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    GLfloat intensitasCahaya[] = {0.9f, 0.9f, 0.9f, 1.0f};
    GLfloat posisiCahaya[] = {2.0f, 2.0f, 2.0f, 0.0f};

    glLightfv(GL_LIGHT0, GL_POSITION, posisiCahaya);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, intensitasCahaya);

    GLfloat bahan_ambient[] = {0.0f, 0.5f, 0.6f, 1.0f};
    GLfloat bahan_diffuse[] = {0.6f, 0.6f, 0.6f, 1.0f};
    GLfloat bahan_specular[] = {1.0f, 1.0f, 1.0f, 1.0f};
    GLfloat bahan_shininess[] = {90.0f};

    glMaterialfv(GL_FRONT, GL_AMBIENT, bahan_ambient);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, bahan_diffuse);
    glMaterialfv(GL_FRONT, GL_SPECULAR, bahan_specular);
    glMaterialfv(GL_FRONT, GL_SHININESS, bahan_shininess);

    glPushMatrix();
    glTranslatef(0.0, 1.0, -8.0);
    glutSolidTetrahedron(); // Menambahkan objek Tetrahedron
    glPopMatrix();

    glPushMatrix();
    glTranslatef(2.0, 1.0, -8.0);
    glutSolidOctahedron(); // Menambahkan objek Octahedron
    glPopMatrix();

    glPushMatrix();
    glTranslatef(2.5f, -2.5f, -8.0);
    glutSolidDodecahedron(); // Menambahkan objek Dodecahedron
    glPopMatrix();
```

```
    glPushMatrix();
    glTranslatef(-4, -2.5f, -8.0);
    glutSolidIcosahedron(); // Menambahkan objek Icosahedron
    glPopMatrix();

    // Menggambar prisma
    glPushMatrix();
    glTranslatef(-2.0, 1.0, -8.0);
    gambarPrisma();
    glPopMatrix();

    // Menggambar antiprisma
    glPushMatrix();
    glTranslatef(4.0, 1.0, -8.0);
    gambarAntiprisma();
    glPopMatrix();

    glFlush();
}

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    gambar3D();
    glutSwapBuffers();
}

void inisialisasi(void)
{
    int w = 800, h = 600;
    glShadeModel(GL_FLAT);
    glClearColor(1.0, 1.0, 1.0, 0.0);
    glClearAccum(0.0, 0.0, 0.0, 0.0);
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60.0, (GLfloat)w / (GLfloat)h, 1.0, 20.0);
    glMatrixMode(GL_MODELVIEW);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_SMOOTH);
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_NORMALIZE);
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutCreateWindow("Pemodelan 3D dengan Jaring Poligon");
    glutDisplayFunc(display);
    inisialisasi();
    glutMainLoop();
    return 0;
```

```
}
```

2. Modifikasi program-3 dengan mengubah warna masing-masing polihendra dengan warna yang berbeda

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

GLfloat light_diffuse[] = {1.0, 0.0, 0.0, 1.0};
GLfloat light_position[] = {1.0, 1.0, 1.0, 0.0};

void hitungNormal(float *normal, float *p1, float *p2, float *p3) {
    float v1[3], v2[3];

    // Hitung vektor dari p1 ke p2
    for (int i = 0; i < 3; i++) {
        v1[i] = p2[i] - p1[i];
    }

    // Hitung vektor dari p1 ke p3
    for (int i = 0; i < 3; i++) {
        v2[i] = p3[i] - p1[i];
    }

    // Hitung vektor normal sebagai hasil cross product v1 x v2
    normal[0] = v1[1] * v2[2] - v1[2] * v2[1];
    normal[1] = v1[2] * v2[0] - v1[0] * v2[2];
    normal[2] = v1[0] * v2[1] - v1[1] * v2[0];

    // Normalisasi vektor normal
    float length = std::sqrt(normal[0] * normal[0] + normal[1] *
normal[1] + normal[2] * normal[2]);
    for (int i = 0; i < 3; i++) {
        normal[i] /= length;
    }
}

void gambarPrisma(void)
{
    // Koordinat titik-titik prisma
    GLfloat vertices[8][3] = {
        {0.0, 0.0, 0.0},
        {1.0, 0.0, 0.0},
        {1.0, 1.0, 0.0},
        {0.0, 1.0, 0.0},
        {0.0, 0.0, 1.0},
        {1.0, 0.0, 1.0},
        {1.0, 1.0, 1.0},
        {0.0, 1.0, 1.0}
    };

    // Indeks permukaan prisma
    int surfaces[6][4] = {
        {0, 1, 2, 3},  // Permukaan alas
```

```
        {4, 5, 6, 7},  // Permukaan atas
        {0, 1, 5, 4},
        {1, 2, 6, 5},
        {2, 3, 7, 6},
        {3, 0, 4, 7}
    };

    // Material prisma
    GLfloat bahan_ambient[] = {1.0f, 0.0f, 0.0f, 1.0f}; // Warna merah
    glMaterialfv(GL_FRONT, GL_AMBIENT, bahan_ambient);

    // Menggambar prisma
    glBegin(GL_QUADS);
    for (int i = 0; i < 6; i++) {
        float normal[3];
        hitungNormal(normal, vertices[surfaces[i][0]],
vertices[surfaces[i][1]], vertices[surfaces[i][2]]);
        glNormal3fv(normal);

        for (int j = 0; j < 4; j++) {
            glVertex3fv(vertices[surfaces[i][j]]);
        }
    }
    glEnd();
}

void gambarAntiprisma(void)
{
    // Koordinat titik-titik antiprisma
    GLfloat vertices[10][3] = {
        {0.0, 0.0, 1.0},
        {1.0, 0.0, 0.0},
        {1.0, 0.0, 1.0},
        {0.0, 0.0, 0.0},
        {0.5, 1.0, 0.5},
        {0.0, 0.0, 1.0},
        {0.0, 0.0, 0.0},
        {1.0, 0.0, 1.0},
        {1.0, 0.0, 0.0},
        {0.5, 1.0, 0.5}
    };

    // Indeks permukaan antiprisma
    int surfaces[7][3] = {
        {0, 1, 2},
        {0, 2, 3},
        {4, 5, 6},
        {4, 7, 8},
        {4, 5, 7},
        {5, 6, 8},
        {6, 7, 8}
    };

    // Material antiprisma
```

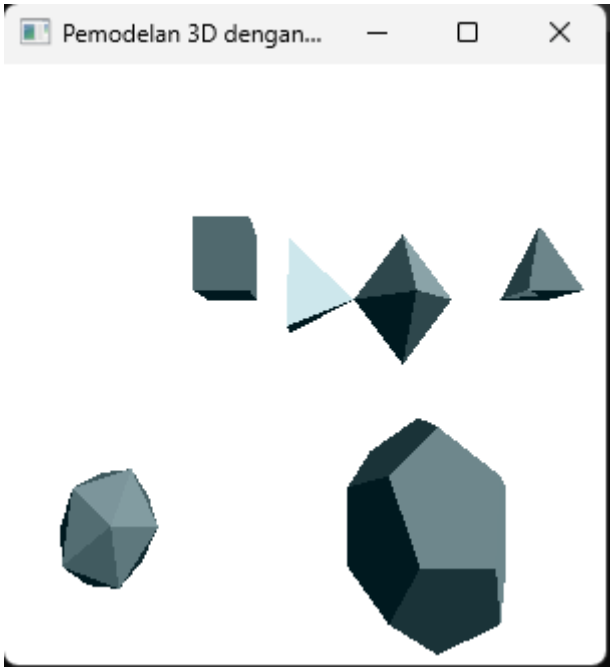
```
    GLfloat bahan_ambient[] = {0.0f, 1.0f, 0.0f, 1.0f}; // Warna hijau
    glMaterialfv(GL_FRONT, GL_AMBIENT, bahan_ambient);

    // Menggambar antiprisma
    glBegin(GL_TRIANGLES);
    for (int i = 0; i < 7; i++) {
        float normal[3];
        hitungNormal(normal, vertices[surfaces[i][0]],
vertices[surfaces[i][1]], vertices[surfaces[i][2]]);
        glNormal3fv(normal);

        for (int j = 0; j < 3; j++) {
            glVertex3fv(vertices[surfaces[i][j]]);
        }
    }
    glEnd();
}

void gambar3D(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    GLfloat intensitasCahaya[] = {0.9f, 0.9f, 0.9f, 1.0f};
    GLfloat posisiCahaya[] = {2.0f, 2.0f, 2.0f, 0.0f};

    glLightfv(GL_LIGHT0, GL_POSITION, posisiCahaya);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, intensitasCahaya);

    // Material objek tetrahedron
    GLfloat tetrahedron_ambient[] = {1.0f, 1.0f, 0.0f, 1.0f}; // Warna
kuning
    glMaterialfv(GL_FRONT, GL_AMBIENT, tetrahedron_ambient);

    glPushMatrix();
    glTranslatef(0.0, 1.0, -8.0);
    glutSolidTetrahedron();
    glPopMatrix();

    // Material objek octahedron
    GLfloat octahedron_ambient[] = {0.0f, 0.0f, 1.0f, 1.0f}; // Warna
biru
    glMaterialfv(GL_FRONT, GL_AMBIENT, octahedron_ambient);

    glPushMatrix();
    glTranslatef(2.0, 1.0, -8.0);
    glutSolidOctahedron();
    glPopMatrix();

    // Material objek dodecahedron
    GLfloat dodecahedron_ambient[] = {1.0f, 0.5f, 0.0f, 1.0f}; // Warna
oranye
    glMaterialfv(GL_FRONT, GL_AMBIENT, dodecahedron_ambient);

    glPushMatrix();
```

```
    glTranslatef(2.5f, -2.5f, -8.0);
    glutSolidDodecahedron();
    glPopMatrix();

    // Material objek icosahedron
    GLfloat icosahedron_ambient[] = {0.0f, 1.0f, 1.0f, 1.0f}; // Warna
biru kehijauan
    glMaterialfv(GL_FRONT, GL_AMBIENT, icosahedron_ambient);

    glPushMatrix();
    glTranslatef(-4, -2.5f, -8.0);
    glutSolidIcosahedron();
    glPopMatrix();

    // Material objek prisma
    GLfloat prisma_ambient[] = {1.0f, 0.0f, 1.0f, 1.0f}; // Warna ungu
    glMaterialfv(GL_FRONT, GL_AMBIENT, prisma_ambient);

    // Menggambar prisma
    glPushMatrix();
    glTranslatef(-2.0, 1.0, -8.0);
    gambarPrisma();
    glPopMatrix();

    // Material objek antiprisma
    GLfloat antiprisma_ambient[] = {0.5f, 0.0f, 0.5f, 1.0f}; // Warna
ungu tua
    glMaterialfv(GL_FRONT, GL_AMBIENT, antiprisma_ambient);

    // Menggambar antiprisma
    glPushMatrix();
    glTranslatef(4.0, 1.0, -8.0);
    gambarAntiprisma();
    glPopMatrix();

    glFlush();
}

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    gambar3D();
    glutSwapBuffers();
}

void inisialisasi(void)
{
    int w = 800, h = 600;
    glShadeModel(GL_FLAT);
    glClearColor(1.0, 1.0, 1.0, 0.0);
    glClearAccum(0.0, 0.0, 0.0, 0.0);
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
```

```
    gluPerspective(60.0, (GLfloat)w / (GLfloat)h, 1.0, 20.0);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_SMOOTH);
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_NORMALIZE);
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutCreateWindow("Pemodelan 3D dengan Jaring Poligon");
    glutDisplayFunc(display);
    inisialisasi();
    glutMainLoop();
    return 0;
}
```

B. Output

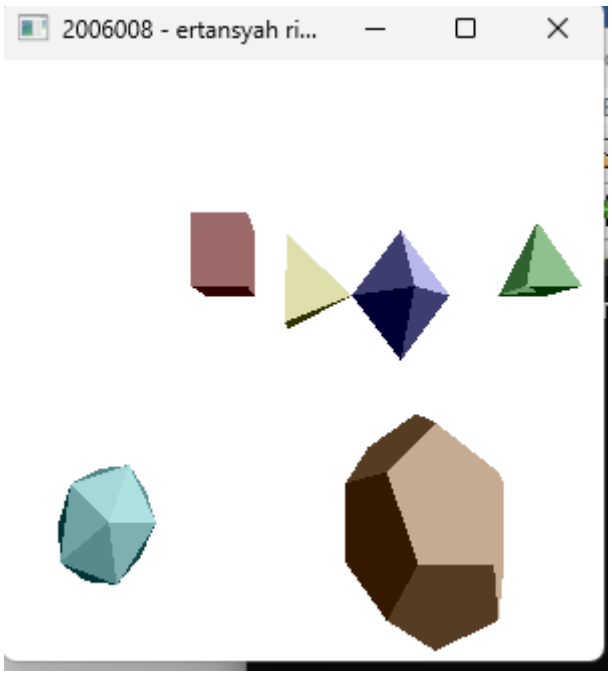
1.



- OpenGL Setup: Kode dimulai dengan mengimpor pustaka OpenGL (#include <GL/glut.h>) dan pustaka matematika (#include <cmath>). Selain itu, beberapa variabel seperti light_diffuse, light_position, dan beberapa fungsi, seperti hitungNormal, didefinisikan untuk pengaturan pencahayaan dan perhitungan vektor normal.
- Hitung Normal: Fungsi hitungNormal digunakan untuk menghitung vektor normal untuk permukaan objek 3D. Vektor normal digunakan untuk perhitungan pencahayaan dan rendering yang realistis.
- Gambar Prisma dan Antiprisma: Dua fungsi gambarPrisma dan gambarAntiprisma digunakan untuk menggambar Prisma dan Antiprisma dengan menggunakan koordinat titik-titik dan indeks permukaan. Vektor normal dihitung dan digunakan untuk mengatur pencahayaan.
- Gambar 3D: Fungsi gambar3D merupakan fungsi utama yang menggambar objek-objek 3D, mengatur cahaya, bahan, dan matriks transformasi untuk objek-objek tersebut.
- Display Function: Fungsi display digunakan untuk membersihkan buffer layar dan menggambar objek-objek 3D. Hasil gambaran disajikan dalam jendela tampilan.
- Inisialisasi: Fungsi inisialisasi digunakan untuk mengatur parameter OpenGL, seperti mode shading, warna latar belakang, perspektif, pencahayaan, dan lain-lain.
- Main Function: Fungsi main adalah fungsi utama yang memulai program. Itu mengatur jendela tampilan, menentukan fungsi display, menginisialisasi OpenGL, dan memulai loop utama program.

Kode ini secara khusus telah dimodifikasi untuk menambahkan objek-objek 3D tambahan seperti Tetrahedron, Octahedron, dan Dodecahedron, selain dari Prisma dan Antiprisma yang sudah ada. Semua objek tersebut diberi bahan, pencahayaan, dan ditransformasikan untuk memposisikan mereka dalam tampilan 3D yang sesuai. Ketika Anda menjalankan program ini, Anda akan melihat jendela OpenGL yang menampilkan semua objek tersebut dengan efek pencahayaan dan material yang telah diatur.

2.



Untuk mengubah warna objek secara terpisah, Anda perlu menetapkan warna pada masing-masing objek yang ingin Anda ganti warnanya. Dalam OpenGL, warna objek dapat diatur dengan mengubah atribut warna tertentu, seperti `GL_AMBIENT`, `GL_DIFFUSE`, atau `GL_SPECULAR`, tergantung pada efek pencahayaan yang Anda inginkan. Di kode yang Anda berikan, warna objek diatur melalui atribut `GL_AMBIENT`. Berikut adalah cara mengubah warna objek secara terpisah:

Pada fungsi gambarPrisma :

Untuk mengubah warna objek prisma, Anda dapat menetapkan nilai atribut `GL_AMBIENT` sebelum Anda memulai menggambar objek tersebut. Sebagai contoh, untuk mengatur warna prisma menjadi merah, Anda dapat menambahkan baris berikut di dalam fungsi `gambarPrisma`:

Pada fungsi`gambarAntiprisma

Demikian pula, untuk mengubah warna objek antiprisma, Anda dapat menetapkan nilai atribut `GL_AMBIENT` sebelum menggambar objek tersebut. Sebagai contoh, untuk mengatur warna antiprisma menjadi hijau, Anda dapat menambahkan baris berikut di dalam fungsi `gambarAntiprisma`:

Dengan mengubah atribut `GL_AMBIENT` seperti ini, Anda dapat mengatur warna objek secara terpisah sesuai dengan preferensi Anda. Pastikan untuk mengatur warna sebelum menggambar objek dan sesuai dengan objek yang sedang digambar. Juga, Anda dapat menggunakan komponen warna merah, hijau, dan biru (RGBA) dalam array sesuai dengan warna yang diinginkan.

C. Penjelasan

1. OpenGL Setup: Pada bagian awal kode, pustaka OpenGL dan pustaka matematika diimpor, dan beberapa variabel seperti `light_diffuse` dan `light_position` dideklarasikan untuk mengatur pencahayaan.

2. Hitung Normal: Fungsi `hitungNormal` digunakan untuk menghitung vektor normal pada permukaan objek 3D. Vektor normal penting untuk perhitungan pencahayaan dan memberikan efek rendering yang realistis.

3. Gambar Prisma dan Antiprisma: Dua fungsi, `gambarPrisma` dan `gambarAntiprisma`, digunakan untuk menggambar objek Prisma dan Antiprisma. Koordinat titik-titik objek ditentukan dalam array `vertices`, dan indeks permukaan ditentukan dalam array `surfaces`. Vektor normal dihitung untuk setiap permukaan dan digunakan untuk mengatur pencahayaan.

4. Gambar 3D: Fungsi `gambar3D` merupakan fungsi utama yang menggambar objek-objek 3D, mengatur cahaya, bahan, dan matriks transformasi untuk objek-objek tersebut. Objek seperti Tetrahedron, Octahedron, Dodecahedron, Prisma, dan Antiprisma ditambahkan dan digambar dalam fungsi ini.

5. Display Function: Fungsi `display` digunakan untuk membersihkan buffer layar dan menggambar objek-objek 3D. Hasil gambaran disajikan dalam jendela tampilan.

6. Inisialisasi: Fungsi `inisialisasi` digunakan untuk mengatur parameter OpenGL, seperti mode shading, warna latar belakang, perspektif, pencahayaan, dan lain-lain.

7. Main Function: Fungsi `main` adalah fungsi utama yang memulai program. Ini mengatur jendela tampilan, menentukan fungsi `display`, menginisialisasi OpenGL, dan memulai loop utama program.

Kode tersebut telah dimodifikasi untuk menambahkan objek-objek 3D seperti Tetrahedron, Octahedron, dan Dodecahedron, selain dari Prisma dan Antiprisma yang sudah ada. Seluruh objek ini diberi bahan, efek pencahayaan, dan ditransformasikan untuk memposisikan mereka dalam tampilan 3D yang sesuai.

Untuk mengubah warna objek secara terpisah, atribut `GL_AMBIENT` digunakan. Ini memungkinkan pengaturan warna untuk setiap objek. Setiap fungsi `gambarPrisma` dan `gambarAntiprisma` diberi atribut warna yang berbeda untuk mengubah warna objek secara individu.

Dengan demikian, kode ini adalah contoh program OpenGL sederhana untuk merender objek-objek 3D dengan warna dan efek pencahayaan yang berbeda.

III. Kesimpulan

Dalam kode program OpenGL yang telah diberikan, kita dapat mengambil beberapa kesimpulan penting:

1. Program ini menunjukkan cara menggunakan OpenGL untuk merender objek-objek 3D dengan efek pencahayaan dan material yang berbeda. Dalam program ini, objek-objek seperti Prisma, Antiprisma, Tetrahedron, Octahedron, dan Dodecahedron ditambahkan dan diberikan efek pencahayaan serta material yang berbeda.

2. Objek-objek 3D ini direpresentasikan dengan menggunakan koordinat titik-titik (vertices) dan indeks permukaan (surfaces). Vektor normal dihitung untuk masing-masing permukaan objek, yang penting untuk perhitungan pencahayaan.

3. Pencahayaan diatur dengan menggunakan atribut `GL_DIFFUSE` dan `GL_LIGHT0`. Selain itu, material objek diatur dengan atribut `GL_AMBIENT`, `GL_DIFFUSE`, `GL_SPECULAR`, dan `GL_SHININESS`.

4. Untuk mengubah warna objek secara terpisah, atribut `GL_AMBIENT` diubah sebelum menggambar objek. Ini memungkinkan penentuan warna individu untuk masing-masing objek dalam tampilan 3D.

5. Program ini memanfaatkan beberapa fungsi OpenGL standar, seperti `glBegin`, `glEnd`, `glVertex`, dan fungsi-fungsi untuk pengaturan tampilan, pencahayaan, dan matriks transformasi.

Kode program ini memberikan pemahaman dasar tentang bagaimana menggunakan OpenGL untuk merender objek-objek 3D dengan pencahayaan dan warna yang berbeda. Dengan pemahaman ini, Anda dapat membuat tampilan 3D yang lebih kompleks dan realistis dalam pengembangan grafika komputer.