PRAKTIKUM GRAFIKA KOMPUTER

Pertemuan ke-6 (Pemodelan 3D dengan Jaring Poligon)

Yoga Agustiansyah
2206050

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

I. PENDAHULUAN

Pemodelan 3D dengan Jaring Poligon adalah salah satu aspek kunci dalam dunia Grafik Komputer yang memungkinkan kita untuk menggambarkan objek tiga dimensi (3D) ke dalam dunia digital. Pemodelan 3D adalah cabang yang sangat penting dalam ilmu komputer yang memungkinkan kita untuk membuat representasi digital yang lebih mendalam dan realistis dari objek nyata atau konseptual. Dalam praktikum mata kuliah Grafik Komputer ini, kita akan menjelajahi konsep dasar Pemodelan 3D dengan Jaring Poligon.

Pemodelan 3D dengan Jaring Poligon adalah teknik yang digunakan untuk merepresentasikan objek tiga dimensi dalam dunia digital. Teknik ini mendasarkan diri pada penggunaan poligon, yang merupakan bentuk geometris dasar yang memiliki sisi-sisi datar. Poligon, seperti segitiga dan segi empat, digunakan sebagai "bata bangunan" dasar untuk membangun objek 3D yang lebih kompleks.

Pemodelan 3D dengan Jaring Poligon adalah teknik yang digunakan untuk merepresentasikan objek tiga dimensi dalam dunia digital. Teknik ini mendasarkan diri pada penggunaan poligon, yang merupakan bentuk geometris dasar yang memiliki sisi-sisi datar. Poligon, seperti segitiga dan segi empat, digunakan sebagai "bata bangunan" dasar untuk membangun objek 3D yang lebih kompleks.

Selain itu, Pemodelan 3D dengan Jaring Poligon juga mencakup konsep materi (material) dan pencahayaan (lighting) yang memengaruhi cara objek berinteraksi dengan cahaya, memberikan kesan kedalaman, bayangan, dan penampakan realistis pada objek 3D.

Dalam praktikum ini, kita akan memahami konsep dasar Pemodelan 3D dengan Jaring Poligon, menggambar objek 3D sederhana, dan memahami bagaimana materi dan pencahayaan dapat memengaruhi tampilan objek. Pengetahuan ini adalah fondasi penting dalam dunia grafik komputer dan berperan dalam pembuatan animasi, permainan komputer, simulasi, dan banyak aplikasi lainnya yang memerlukan representasi 3D yang akurat dan realistis.

II. PEMBAHASAN

SOAL 1

1. Buatlah program dengan OpenGL untuk menampilkan objek 3D dengan metoda 3 daftar yaitu (1) titik (2) normal (3) permukaan:

a) Prisma
b) Antiprisma
c) Tetrahedron
d) Octahedron
e) Dodecahedron
f) Icosahedron

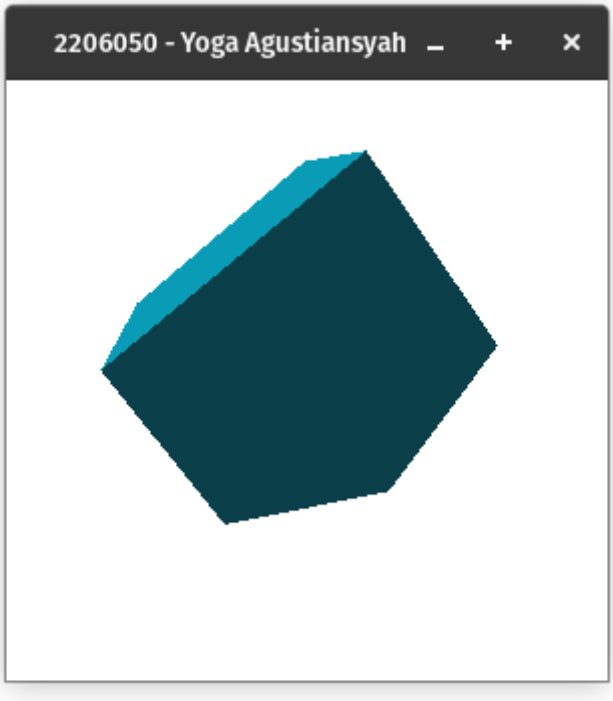
a) Prisma Segi Lima

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

GLfloat light_diffuse[] = {0.0, 0.850, 1.0, 1.0};
GLfloat light_position[] = {1.0, 1.0, 1.0, 0.0};

void pentagonPrism(void)
{
    glRotatef(1.0, 0.5, 1.0, 0.0);

    glBegin(GL_POLYGON);
    glNormal3f(0.0f, 0.0f, 1.0f);  // Normal menghadap ke depan
    glColor3f(1.0, 0.0, 0.0);
    glVertex3f(0.0f, 6.0f, 2.0f);
    glVertex3f(5.0f, 2.0f, 2.0f);
    glVertex3f(3.0f, -4.0f, 2.0f);
    glVertex3f(-3.0f, -4.0f, 2.0f);
    glVertex3f(-5.0f, 2.0f, 2.0f);
    glEnd();

    glBegin(GL_POLYGON);
    glNormal3f(0.0f, 0.0f, -1.0f);  // Normal menghadap ke belakang
    glColor3f(1.0, 0.0, 0.0);
    glVertex3f(0.0f, 6.0f, -2.0f);
    glVertex3f(5.0f, 2.0f, -2.0f);
    glVertex3f(3.0f, -4.0f, -2.0f);
    glVertex3f(-3.0f, -4.0f, -2.0f);
    glVertex3f(-5.0f, 2.0f, -2.0f);
    glEnd();

    glBegin(GL_POLYGON);
    glNormal3f(0.0f, 1.0f, 0.0f);  // Normal menghadap ke atas
    glColor3f(0, 0, 1);
    glVertex3f(0.0f, 6.0f, -2.0f);
    glVertex3f(5.0f, 2.0f, -2.0f);
    glVertex3f(5.0f, 2.0f, 2.0f);
    glVertex3f(0.0f, 6.0f, 2.0f);
    glEnd();
```

```
    glBegin(GL_POLYGON);
    glNormal3f(-1.0f, 0.0f, 0.0f);  // Normal menghadap ke kiri
    glColor3f(0, 0, 1);
    glVertex3f(5.0f, 2.0f, -2.0f);
    glVertex3f(3.0f, -4.0f, -2.0f);
    glVertex3f(3.0f, -4.0f, 2.0f);
    glVertex3f(5.0f, 2.0f, 2.0f);
    glEnd();

    glBegin(GL_POLYGON);
    glNormal3f(1.0f, 0.0f, 0.0f);  // Normal menghadap ke kanan
    glColor3f(0, 0, 1);
    glVertex3f(3.0f, -4.0f, -2.0f);
    glVertex3f(-3.0f, -4.0f, -2.0f);
    glVertex3f(-3.0f, -4.0f, 2.0f);
    glVertex3f(3.0f, -4.0f, 2.0f);
    glEnd();

    glBegin(GL_POLYGON);
    glNormal3f(0.0f, 0.0f, -1.0f);  // Normal menghadap ke belakang
    glColor3f(0, 0, 1);
    glVertex3f(-3.0f, -4.0f, -2.0f);
    glVertex3f(-5.0f, 2.0f, -2.0f);
    glVertex3f(-5.0f, 2.0f, 2.0f);
    glVertex3f(-3.0f, -4.0f, 2.0f);
    glEnd();

    glBegin(GL_POLYGON);
    glNormal3f(0.0f, 1.0f, 0.0f);  // Normal menghadap ke atas
    glColor3f(0, 0, 1);
    glVertex3f(0.0f, 6.0f, -2.0f);
    glVertex3f(-5.0f, 2.0f, -2.0f);
    glVertex3f(-5.0f, 2.0f, 2.0f);
    glVertex3f(0.0f, 6.0f, 2.0f);
    glEnd();

    glBegin(GL_POLYGON);
    glNormal3f(-1.0f, 0.0f, 0.0f);  // Normal menghadap ke kiri
    glColor3f(0, 0, 1);
    glVertex3f(-5.0f, 2.0f, -2.0f);
    glVertex3f(-3.0f, -4.0f, -2.0f);
    glVertex3f(-3.0f, -4.0f, 2.0f);
    glVertex3f(-5.0f, 2.0f, 2.0f);
    glEnd();

    glBegin(GL_POLYGON);
    glNormal3f(1.0f, 0.0f, 0.0f);  // Normal menghadap ke kanan
    glColor3f(0, 0, 1);
    glVertex3f(0.0f, 6.0f, -2.0f);
    glVertex3f(-5.0f, 2.0f, -2.0f);
    glVertex3f(-5.0f, 2.0f, 2.0f);
    glVertex3f(0.0f, 6.0f, 2.0f);
    glEnd();
```

```
    glutPostRedisplay();
}

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

void init(void)
{
    glClearColor(1, 1, 1, 1);
    // PENCAHAYAAN
    glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glEnable(GL_LIGHT0);

    glEnable(GL_LIGHTING);

    // BUFFER
    glEnable(GL_DEPTH_TEST);

    // SETUP CUBE
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(90.0, 1.0, 1.0, 100.0); // 45 derajat FOV, rasio
aspek 1.0, jarak dekat 1.0, jarak jauh 100.0
    // glOrtho(-10.0, 10.0, -10.0, 10.0, -10, 10);

    glMatrixMode(GL_MODELVIEW);
    gluLookAt(0.0, 0.0, 10.0, // Posisi kamera
              0.0, 0.0, 0.0,  // Titik pandang
              0.0, 1.0, 0.0); // Vektor arah atas

    // glRotatef(30.0, 0.0, 1.0, 1.0);
    // glRotatef(15.0, 1.0, 1.0, 0.0);
}


int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutCreateWindow("2206050 - Yoga Agustiansyah");
    glutDisplayFunc(display);
    init();
    glutMainLoop();
    return 0;
}
```

Output



Program ini menghasilkan sebuah prisma berbentuk segi lima. Prisma segi lima adalah bangun ruang tiga dimensi yang memiliki atap dan alas berbentuk segi lima dan memiliki selimut yang berbentuk persegi panjang pada 5 sisinya.

Program dimulai dengan mengimpor library GLUT dan mendefinisikan dua array GLfloat: light_diffuse dan light_position. light_diffuse digunakan untuk menentukan warna cahaya difus, dan light_position menentukan posisi cahaya.

Terdapat fungsi pentagonPrism yang digunakan untuk menggambar prisma segi lima. Di dalamnya, ada beberapa pemanggilan glBegin yang menentukan permukaan prisma dengan poligon berwarna merah dan biru. Setiap permukaan didefinisikan dengan menggunakan glNormal3f untuk menentukan arah normal permukaan, dan glColor3f untuk menentukan warna poligon.

Fungsi display digunakan untuk menggambar objek dan dipanggil dalam loop utama glut. Di sini, kita memanggil pentagonPrism untuk menggambar prisma.

Dalam fungsi init, beberapa pengaturan awal dilakukan, termasuk pengaturan latar belakang berwarna putih (glClearColor), pencahayaan difus (glLightfv), pengaturan buffer (glEnable(GL_DEPTH_TEST)), dan pengaturan proyeksi dan pandangan kamera.

Program utama dimulai dengan menginisialisasi GLUT, mengatur mode tampilan, membuat jendela, dan menjalankan loop utama GLUT. Di dalam loop, fungsi display akan dipanggil untuk menggambar objek pada jendela.

b) AntiPrisma Segi Lima

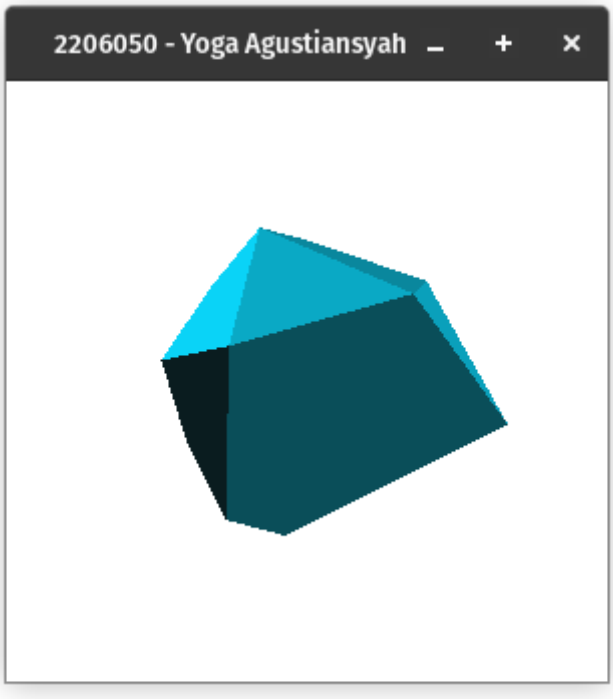
```
#include <GL/glut.h>

GLfloat light_diffuse[] = {0.0, 0.850, 1.0, 1.0};
GLfloat light_position[] = {1.0, 1.0, 1.0, 0.0};

void pentagonPrism(void)
{
    glRotatef(1.0, 0.5, 1.0, 0.0);

    // Sisi depan
    glBegin(GL_POLYGON);
    glNormal3f(0.0f, 0.0f, 1.0f);  // Normal menghadap ke depan
    glColor3f(1.0, 0.0, 0.0);
    glVertex3f(0.0f, 6.0f, 2.0f);
    glVertex3f(5.0f, 2.0f, 2.0f);
    glVertex3f(3.0f, -4.0f, 2.0f);
    glVertex3f(-3.0f, -4.0f, 2.0f);
    glVertex3f(-5.0f, 2.0f, 2.0f);
    glEnd();

    // Sisi bawah
    glBegin(GL_POLYGON);
    glNormal3f(0.0f, 0.0f, -1.0f);  // Normal menghadap ke belakang
    glColor3f(0.0, 0.0, 1.0);
    glVertex3f(0.0f, -4.0f, -2.0f);
    glVertex3f(-5.0f, 0.0f, -2.0f);
    glVertex3f(-3.0f, 6.0f, -2.0f);
    glVertex3f(3.0f, 6.0f, -2.0f);
    glVertex3f(5.0f, 0.0f, -2.0f);
    glEnd();

    // Sisi atas
    glBegin(GL_POLYGON);
    glNormal3f(0.0f, 1.0f, 0.0f);
    glColor3f(0, 1, 0);
    glVertex3f(0.0f, 6.0f, 2.0f);
    glVertex3f(5.0f, 2.0f, 2.0f);
    glVertex3f(3.0f, 6.0f, -2.0f);
    glEnd();

    // Sisi kiri depan
    glBegin(GL_POLYGON);
    glNormal3f(-1.0f, 0.5f, 0.0f);
    glVertex3f(-3.0f, 6.0f, -2.0f);
    glVertex3f(3.0f, 6.0f, -2.0f);
    glVertex3f(0.0f, 6.0f, 2.0f);
    glEnd();

    // Sisi kanan depan
    glBegin(GL_POLYGON);
    glNormal3f(1.0f, 0.5f, 0.0f);
    glVertex3f(3.0f, 6.0f, -2.0f);
```

```
    glVertex3f(5.0f, 0.0f, -2.0f);
    glVertex3f(5.0f, 2.0f, 2.0f);
    glEnd();

    // Sisi kiri agak depan
    glBegin(GL_POLYGON);
    glNormal3f(-1.0f, 0.0f, 0.5f);
    glVertex3f(0.0f, 6.0f, 2.0f);
    glVertex3f(-5.0f, 2.0f, 2.0f);
    glVertex3f(-3.0f, 6.0f, -2.0f);
    glEnd();

    // Sisi kanan agak depan
    glBegin(GL_POLYGON);
    glNormal3f(1.0f, 0.0f, 0.5f);
    glVertex3f(5.0f, 2.0f, 2.0f);
    glVertex3f(3.0f, -4.0f, 2.0f);
    glVertex3f(5.0f, 0.0f, -2.0f);
    glEnd();

    // Sisi kiri
    glBegin(GL_POLYGON);
    glNormal3f(-1.0f, 0.0f, 0.0f);
    glVertex3f(-5.0f, 0.0f, -2.0f);
    glVertex3f(-3.0f, 6.0f, -2.0f);
    glVertex3f(-5.0f, 2.0f, 2.0f);
    glEnd();

    // Sisi kanan
    glBegin(GL_POLYGON);
    glNormal3f(1.0f, 0.0f, 0.0f);
    glVertex3f(0.0f, -4.0f, -2.0f);
    glVertex3f(5.0f, 0.0f, -2.0f);
    glVertex3f(3.0f, -4.0f, 2.0f);
    glEnd();

    // Sisi kiri agak belakang
    glBegin(GL_POLYGON);
    glNormal3f(-1.0f, -0.5f, -0.5f);
    glVertex3f(-3.0f, -4.0f, 2.0f);
    glVertex3f(-5.0f, 2.0f, 2.0f);
    glVertex3f(-5.0f, 0.0f, -2.0f);
    glEnd();

    // Sisi kanan agak belakang
    glBegin(GL_POLYGON);
    glNormal3f(1.0f, -0.5f, -0.5f);
    glVertex3f(3.0f, -4.0f, 2.0f);
    glVertex3f(-3.0f, -4.0f, 2.0f);
    glVertex3f(0.0f, -4.0f, -2.0f);
    glEnd();

    // Sisi belakang
    glBegin(GL_POLYGON);
```

```
    glNormal3f(0.0f, -1.0f, 0.0f);
    // glNormal3f(0.0f, 0.0f, -1.0f);
    glColor3f(1, 0, 1);
    glVertex3f(0.0f, -4.0f, -2.0f);
    glVertex3f(-5.0f, 0.0f, -2.0f);
    glVertex3f(-3.0f, -4.0f, 2.0f);
    glEnd();

    glutPostRedisplay();
}

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

void init(void)
{
    glClearColor(1, 1, 1, 1);
    // PENCAHAYAAN
    glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glEnable(GL_LIGHT0);

    glEnable(GL_LIGHTING);

    // BUFFER
    glEnable(GL_DEPTH_TEST);

    // SETUP CUBE
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    // gluPerspective(120.0, 1.0, 1.0, 100.0); // 45 derajat FOV, rasio
aspek 1.0, jarak dekat 1.0, jarak jauh 100.0
    glOrtho(-10.0, 10.0, -10.0, 10.0, -10, 10);

    glMatrixMode(GL_MODELVIEW);
    gluLookAt(0.0, 0.0, 10.0, // Posisi kamera
              0.0, 0.0, 0.0,  // Titik pandang
              0.0, 1.0, 0.0); // Vektor arah atas

    // glRotatef(30.0, 0.0, 1.0, 1.0);
    // glRotatef(15.0, 1.0, 1.0, 0.0);
}


int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutCreateWindow("2206050 - Yoga Agustiansyah");
    glutDisplayFunc(display);
```

```
    init();
    glutMainLoop();
    return 0;
}
```

Output



Program ini menghasilkan sebuah antiprisma berbentuk segi lima. Antirisma segi lima adalah bangun ruang tiga dimensi yang memiliki sisi atap dan sisi alas berbentuk segi lima, namun pada salah satunya diputar sebesar 180 derajat terhadap sisi yang lain. Antiprisma segi lima memiliki selimut yang berbentuk segitiga sebanyak 5 pasang atas bawah yang saling terhubung dengan sisi alas dan sisi atas.

Fungsi pentagonAntiPrism digunakan untuk menggambar anti-prisma segi lima dengan beberapa sisi yang berbeda. Setiap sisi didefinisikan sebagai poligon menggunakan glBegin dan glEnd, dan masing-masing sisi memiliki warna dan arah normal yang berbeda.

Fungsi display digunakan untuk membersihkan buffer layar, menggambar anti-prisma dengan memanggil pentagonAntiPrism, dan menukar buffer untuk tampilan yang mulus.

Dalam fungsi init, beberapa pengaturan awal dilakukan, termasuk pengaturan latar belakang berwarna putih, pencahayaan difus, pengaturan buffer, dan proyeksi dan pandangan kamera.

Program utama dimulai dengan inisialisasi GLUT, pengaturan mode tampilan, pembuatan jendela, dan menjalankan loop utama GLUT. Program ini dapat digunakan sebagai dasar untuk membuat objek 3D yang lebih kompleks dalam bidang grafik komputer dan memahami konsep dasar rendering 3D dengan OpenGL.

Program utama dimulai dengan menginisialisasi GLUT, mengatur mode tampilan, membuat jendela, dan menjalankan loop utama GLUT. Di dalam loop, fungsi display akan dipanggil untuk menggambar objek pada jendela.

c) Tetrahedron

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

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

void tetraHedron(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);

    // Objek Tetrahedron
    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[] = {0.2f, 0.6f, 0.2f, 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);
    glRotated(1, 1, 1, 1);
    glTranslatef(0, 0.0, -5.0);
    glScaled(3, 3, 3);
    glRotated(90, 1, 1, 1);

    glutSolidTetrahedron();

    glFlush();
}

void Display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    tetraHedron();
    glutSwapBuffers();
}

void init(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, 10.0);
```

```
    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("2206050 - Yoga Agustiansyah");
    glutDisplayFunc(Display);
    init();
    glutMainLoop();
    return 0;
}
```

Output

Program ini menghasilkan sebuah antiprisma tetrahedron. Tetrahedron adalah bangun ruang tiga dimensi yang terdiri dari empat muka segitiga, enam garis rusuk yang lurus, dan empat titik sudut. Tetrahedron juga dikenal sebagai piramida segitiga karena memiliki alas berbentuk segitiga.

Fungsi tetraHedron digunakan untuk menggambar tetrahedron. Pertama, program mengatur intensitas cahaya dan posisi cahaya. Kemudian, program mendefinisikan sifat-sifat material dari tetrahedron, seperti ambient, diffuse, specular, dan shininess. Setelah itu, tetrahedron digambar dengan menggunakan glutSolidTetrahedron. Tetrahedron tersebut dipindahkan, diputar, dan diubah ukurannya sebelum digambar.

Fungsi Display digunakan untuk membersihkan buffer layar, menggambar tetrahedron dengan memanggil tetraHedron, dan menukar buffer untuk tampilan yang mulus.

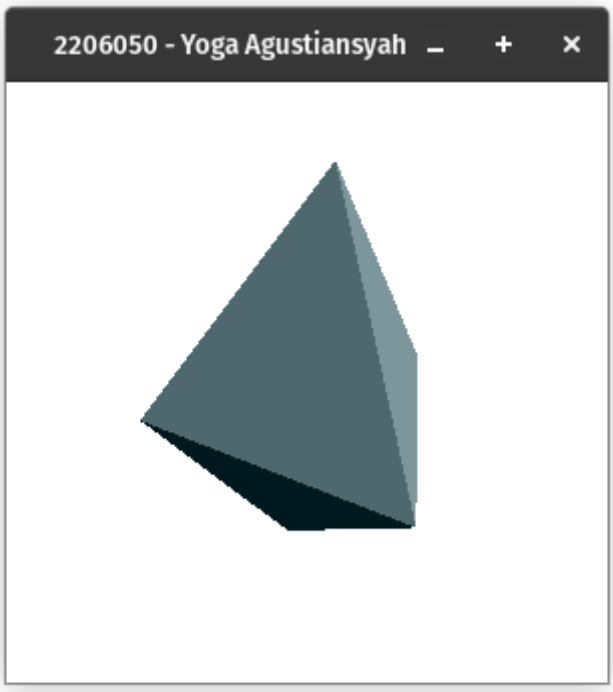
Dalam fungsi Inisial, beberapa pengaturan awal dilakukan, seperti mengatur tampilan OpenGL, pengaturan latar belakang berwarna putih, pengaturan proyeksi perspektif, pengaktifan pencahayaan, dan pengaturan lainnya yang diperlukan untuk rendering 3D.

d) Octahedron

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

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

void octaHedron(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);

    // Objek Tetrahedron
    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[] = {0.2f, 0.6f, 0.2f, 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);
    glTranslatef(0, 0.0, -5.0);
    glScaled(2, 2, 2);
    glRotated(45, 1, 1, 0);

    glutSolidOctahedron();

    glFlush();
}

void Display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    octaHedron();
    glutSwapBuffers();
}

void Init(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, 10.0);
    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("2206050 - Yoga Agustiansyah");
    glutDisplayFunc(Display);
    Init();
    glutMainLoop();
    return 0;
}
```

Output



Program ini menghasilkan sebuah octahedron. Octahedron adalah bangun ruang tiga dimensi yang terdiri dari delapan muka segitiga yang sama besar. Setiap titik sudut octahedron bertemu dengan empat rusuk.

e) Dodecahedron

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

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

void doodecaHedron(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);

    // Objek Tetrahedron
    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[] = {0.2f, 0.6f, 0.2f, 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);
    glTranslatef(0, 0.0, -5.0);
    glRotated(45, 1, 1, 0);

    glutSolidDodecahedron();

    glFlush();
}

void Display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    doodecaHedron();
    glutSwapBuffers();
}

void init(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, 10.0);
    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("2206050 - Yoga Agustiansyah");
    glutDisplayFunc(Display);
    init();
    glutMainLoop();
    return 0;
}
```

Output

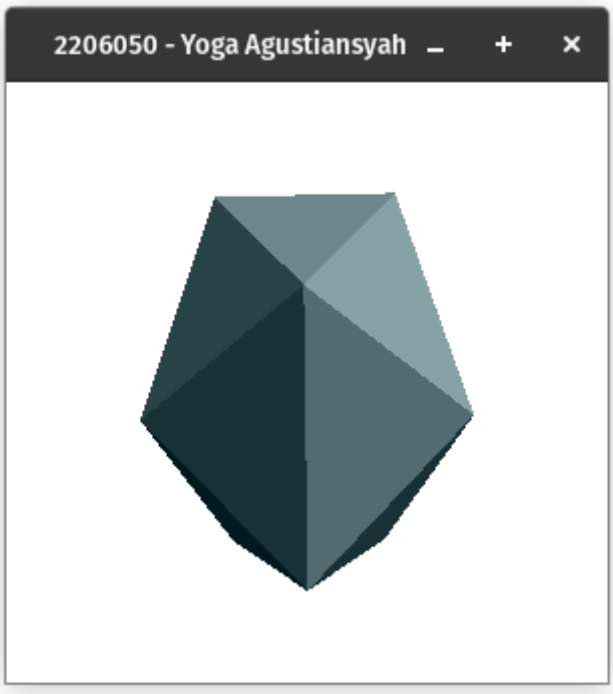
Program ini menghasilkan sebuah dodecahedron. Dodecahedron adalah bangun ruang tiga dimensi yang memiliki dua belas sisi atau muka. Dodecahedron yang paling terkenal adalah dodecahedron beraturan, yang memiliki dua belas pentagon beraturan sebagai mukanya dan dikategorikan sebagai bangun ruang Platonik.

f) Icosahedron

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

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

void icosaHedron(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);

    // Objek Tetrahedron
    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[] = {0.2f, 0.6f, 0.2f, 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);
    glTranslatef(0, 0.0, -5.0);
    glScaled(2, 2, 2);
    glRotated(45, 1, 1, 0);
    glutSolidIcosahedron();

    glFlush();
    // glutPostRedisplay();
}

void Display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    icosaHedron();
    glutSwapBuffers();
}

void init(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, 10.0);
    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("2206050 - Yoga Agustiansyah");
    glutDisplayFunc(Display);
    init();
    glutMainLoop();
    return 0;
}
```

Output



Program ini menghasilkan sebuah icosaHedron. Icosahedron adalah bangun ruang tiga dimensi yang memiliki 20 sisi atau muka. Icosahedron memiliki bentuk segitiga sama sisi pada setiap sisinya. Icosahedron merupakan salah satu dari lima bangun ruang Platonik, selain tetrahedron, kubus, oktahedron, dan dodecahedron.

g) Disatukan

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

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

void pentagonPrism(void)
{
    // isi function pentagonPrism sebelumnya

}

void pentagonAntiPrism(void)
{
    // isi function pentagonAntiPrism sebelumnya
}

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(-3.0, 1.0, -8.0);
    glutSolidTetrahedron();
    glPopMatrix();

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

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

    glPushMatrix();
    glTranslatef(-4.0, -2.5f, -8.0);
```

```
    glutSolidIcosahedron();
    glPopMatrix();

    glPushMatrix();
    glTranslatef(-2.0, 3.0f, -8.0);
    glScalef(0.2f, 0.2f, 0.2f);
    pentagonPrism();
    glPopMatrix();

    glPushMatrix();
    glTranslatef(2.0, 3.0f, -8.0);
    glScalef(0.2f, 0.2f, 0.2f);
    pentagonAntiPrism();
    glPopMatrix();

    glFlush();
}


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

void init(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("2206050 - Yoga Agustiansyah");
    glutDisplayFunc(display);
    init();
    glutMainLoop();
    return 0;
}
```

Output

A

SOAL 2

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

a) Prisma Segi Lima

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

GLfloat light_diffuse[] = {0.215f, 0.423f, 0.784f, 1.0f};
GLfloat light_position[] = {1.0, 1.0, 1.0, 0.0};

void pentagonPrism(void)
{
    glRotatef(1.0, 0.5, 1.0, 0.0);

    glBegin(GL_POLYGON);
    glNormal3f(0.0f, 0.0f, 1.0f);  // Normal menghadap ke depan
    glColor3f(1.0, 0.0, 0.0);
    glVertex3f(0.0f, 6.0f, 2.0f);
    glVertex3f(5.0f, 2.0f, 2.0f);
    glVertex3f(3.0f, -4.0f, 2.0f);
    glVertex3f(-3.0f, -4.0f, 2.0f);
    glVertex3f(-5.0f, 2.0f, 2.0f);
    glEnd();

    glBegin(GL_POLYGON);
    glNormal3f(0.0f, 0.0f, -1.0f);  // Normal menghadap ke belakang
    glColor3f(1.0, 0.0, 0.0);
    glVertex3f(0.0f, 6.0f, -2.0f);
    glVertex3f(5.0f, 2.0f, -2.0f);
    glVertex3f(3.0f, -4.0f, -2.0f);
    glVertex3f(-3.0f, -4.0f, -2.0f);
    glVertex3f(-5.0f, 2.0f, -2.0f);
    glEnd();

    glBegin(GL_POLYGON);
    glNormal3f(0.0f, 1.0f, 0.0f);  // Normal menghadap ke atas
    glColor3f(0, 0, 1);
    glVertex3f(0.0f, 6.0f, -2.0f);
    glVertex3f(5.0f, 2.0f, -2.0f);
    glVertex3f(5.0f, 2.0f, 2.0f);
    glVertex3f(0.0f, 6.0f, 2.0f);
    glEnd();

    glBegin(GL_POLYGON);
    glNormal3f(-1.0f, 0.0f, 0.0f);  // Normal menghadap ke kiri
    glColor3f(0, 0, 1);
    glVertex3f(5.0f, 2.0f, -2.0f);
    glVertex3f(3.0f, -4.0f, -2.0f);
    glVertex3f(3.0f, -4.0f, 2.0f);
    glVertex3f(5.0f, 2.0f, 2.0f);
    glEnd();
```

```
    glBegin(GL_POLYGON);
    glNormal3f(1.0f, 0.0f, 0.0f);  // Normal menghadap ke kanan
    glColor3f(0, 0, 1);
    glVertex3f(3.0f, -4.0f, -2.0f);
    glVertex3f(-3.0f, -4.0f, -2.0f);
    glVertex3f(-3.0f, -4.0f, 2.0f);
    glVertex3f(3.0f, -4.0f, 2.0f);
    glEnd();

    glBegin(GL_POLYGON);
    glNormal3f(0.0f, 0.0f, -1.0f);  // Normal menghadap ke belakang
    glColor3f(0, 0, 1);
    glVertex3f(-3.0f, -4.0f, -2.0f);
    glVertex3f(-5.0f, 2.0f, -2.0f);
    glVertex3f(-5.0f, 2.0f, 2.0f);
    glVertex3f(-3.0f, -4.0f, 2.0f);
    glEnd();

    glBegin(GL_POLYGON);
    glNormal3f(0.0f, 1.0f, 0.0f);  // Normal menghadap ke atas
    glColor3f(0, 0, 1);
    glVertex3f(0.0f, 6.0f, -2.0f);
    glVertex3f(-5.0f, 2.0f, -2.0f);
    glVertex3f(-5.0f, 2.0f, 2.0f);
    glVertex3f(0.0f, 6.0f, 2.0f);
    glEnd();

    glBegin(GL_POLYGON);
    glNormal3f(-1.0f, 0.0f, 0.0f);  // Normal menghadap ke kiri
    glColor3f(0, 0, 1);
    glVertex3f(-5.0f, 2.0f, -2.0f);
    glVertex3f(-3.0f, -4.0f, -2.0f);
    glVertex3f(-3.0f, -4.0f, 2.0f);
    glVertex3f(-5.0f, 2.0f, 2.0f);
    glEnd();

    glBegin(GL_POLYGON);
    glNormal3f(1.0f, 0.0f, 0.0f);  // Normal menghadap ke kanan
    glColor3f(0, 0, 1);
    glVertex3f(0.0f, 6.0f, -2.0f);
    glVertex3f(-5.0f, 2.0f, -2.0f);
    glVertex3f(-5.0f, 2.0f, 2.0f);
    glVertex3f(0.0f, 6.0f, 2.0f);
    glEnd();

    glutPostRedisplay();
}

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

```
void init(void)
{
    glClearColor(1, 1, 1, 1);
    // PENCAHAYAAN
    glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glEnable(GL_LIGHT0);

    glEnable(GL_LIGHTING);

    // BUFFER
    glEnable(GL_DEPTH_TEST);

    // SETUP CUBE
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(90.0, 1.0, 1.0, 100.0); // 45 derajat FOV, rasio
aspek 1.0, jarak dekat 1.0, jarak jauh 100.0
    // glOrtho(-10.0, 10.0, -10.0, 10.0, -10, 10);

    glMatrixMode(GL_MODELVIEW);
    gluLookAt(0.0, 0.0, 10.0, // Posisi kamera
              0.0, 0.0, 0.0,  // Titik pandang
              0.0, 1.0, 0.0); // Vektor arah atas

    // glRotatef(30.0, 0.0, 1.0, 1.0);
    // glRotatef(15.0, 1.0, 1.0, 0.0);
}


int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutCreateWindow("2206050 - Yoga Agustiansyah");
    glutDisplayFunc(display);
    init();
    glutMainLoop();
    return 0;
}
```

Output

Prisma segi lima diubah warnanya menjadi rgba(55, 108, 200, 1)

b) AntiPrisma Segi Lima

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

GLfloat light_diffuse[] = {0.215f, 0.784f, 0.294f, 1.0f};
GLfloat light_position[] = {1.0, 1.0, 1.0, 0.0};

void pentagonAntiPrism(void)
{
    glRotatef(1.0, 0.5, 1.0, 0.0);

    // Sisi atas
    glBegin(GL_POLYGON);
    glNormal3f(0.0f, 0.0f, 1.0f);  // Normal menghadap ke depan
    glColor3f(1.0, 0.0, 0.0);
    glVertex3f(0.0f, 6.0f, 2.0f);
    glVertex3f(5.0f, 2.0f, 2.0f);
    glVertex3f(3.0f, -4.0f, 2.0f);
    glVertex3f(-3.0f, -4.0f, 2.0f);
    glVertex3f(-5.0f, 2.0f, 2.0f);
    glEnd();

    // Sisi bawah
    glBegin(GL_POLYGON);
    glNormal3f(0.0f, 0.0f, -1.0f);  // Normal menghadap ke belakang
    glColor3f(0.0, 0.0, 1.0);
    glVertex3f(0.0f, -4.0f, -2.0f);
    glVertex3f(-5.0f, 0.0f, -2.0f);
    glVertex3f(-3.0f, 6.0f, -2.0f);
    glVertex3f(3.0f, 6.0f, -2.0f);
    glVertex3f(5.0f, 0.0f, -2.0f);
    glEnd();

    // Sisi depan
    glBegin(GL_POLYGON);
    glNormal3f(0.0f, 1.0f, 0.0f);
    glColor3f(0, 1, 0);
    glVertex3f(0.0f, 6.0f, 2.0f);
    glVertex3f(5.0f, 2.0f, 2.0f);
    glVertex3f(3.0f, 6.0f, -2.0f);
    glEnd();

    // Sisi kiri depan
    glBegin(GL_POLYGON);
    glNormal3f(-1.0f, 0.5f, 0.0f);
    glVertex3f(-3.0f, 6.0f, -2.0f);
    glVertex3f(3.0f, 6.0f, -2.0f);
    glVertex3f(0.0f, 6.0f, 2.0f);
    glEnd();

    // Sisi kanan depan
    glBegin(GL_POLYGON);
    glNormal3f(1.0f, 0.5f, 0.0f);
    glVertex3f(3.0f, 6.0f, -2.0f);
```

```
    glVertex3f(5.0f, 0.0f, -2.0f);
    glVertex3f(5.0f, 2.0f, 2.0f);
    glEnd();

    // Sisi kiri agak depan
    glBegin(GL_POLYGON);
    glNormal3f(-1.0f, 0.0f, 0.5f);
    glVertex3f(0.0f, 6.0f, 2.0f);
    glVertex3f(-5.0f, 2.0f, 2.0f);
    glVertex3f(-3.0f, 6.0f, -2.0f);
    glEnd();

    // Sisi kanan agak depan
    glBegin(GL_POLYGON);
    glNormal3f(1.0f, 0.0f, 0.5f);
    glVertex3f(5.0f, 2.0f, 2.0f);
    glVertex3f(3.0f, -4.0f, 2.0f);
    glVertex3f(5.0f, 0.0f, -2.0f);
    glEnd();

    // Sisi kiri
    glBegin(GL_POLYGON);
    glNormal3f(-1.0f, 0.0f, 0.0f);
    glVertex3f(-5.0f, 0.0f, -2.0f);
    glVertex3f(-3.0f, 6.0f, -2.0f);
    glVertex3f(-5.0f, 2.0f, 2.0f);
    glEnd();

    // Sisi kanan
    glBegin(GL_POLYGON);
    glNormal3f(1.0f, 0.0f, 0.0f);
    glVertex3f(0.0f, -4.0f, -2.0f);
    glVertex3f(5.0f, 0.0f, -2.0f);
    glVertex3f(3.0f, -4.0f, 2.0f);
    glEnd();

    // Sisi kiri agak belakang
    glBegin(GL_POLYGON);
    glNormal3f(-1.0f, -0.5f, -0.5f);
    glVertex3f(-3.0f, -4.0f, 2.0f);
    glVertex3f(-5.0f, 2.0f, 2.0f);
    glVertex3f(-5.0f, 0.0f, -2.0f);
    glEnd();

    // Sisi kanan agak belakang
    glBegin(GL_POLYGON);
    glNormal3f(1.0f, -0.5f, -0.5f);
    glVertex3f(3.0f, -4.0f, 2.0f);
    glVertex3f(-3.0f, -4.0f, 2.0f);
    glVertex3f(0.0f, -4.0f, -2.0f);
    glEnd();

    // Sisi belakang
    glBegin(GL_POLYGON);
```

```
    glNormal3f(0.0f, -1.0f, 0.0f);
    // glNormal3f(0.0f, 0.0f, -1.0f);
    glColor3f(1, 0, 1);
    glVertex3f(0.0f, -4.0f, -2.0f);
    glVertex3f(-5.0f, 0.0f, -2.0f);
    glVertex3f(-3.0f, -4.0f, 2.0f);
    glEnd();

    glutPostRedisplay();
}

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

void init(void)
{
    glClearColor(1, 1, 1, 1);
    // PENCAHAYAAN
    glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glEnable(GL_LIGHT0);

    glEnable(GL_LIGHTING);

    // BUFFER
    glEnable(GL_DEPTH_TEST);

    // SETUP CUBE
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    // gluPerspective(120.0, 1.0, 1.0, 100.0); // 45 derajat FOV, rasio
aspek 1.0, jarak dekat 1.0, jarak jauh 100.0
    glOrtho(-10.0, 10.0, -10.0, 10.0, -10, 10);

    glMatrixMode(GL_MODELVIEW);
    gluLookAt(0.0, 0.0, 10.0, // Posisi kamera
              0.0, 0.0, 0.0,  // Titik pandang
              0.0, 1.0, 0.0); // Vektor arah atas

    // glRotatef(30.0, 0.0, 1.0, 1.0);
    // glRotatef(15.0, 1.0, 1.0, 0.0);
}


int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutCreateWindow("2206050 - Yoga Agustiansyah");
    glutDisplayFunc(display);
```

```
    init();
    glutMainLoop();
    return 0;
}
```

Output

Warna anti prisma segi lima diubah menjadi rgba(55, 200, 75, 1).

c) Tetrahedron

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

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

void tetraHedron(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);

    // Objek Tetrahedron
    GLfloat bahan_ambient[] = {0.784f, 0.215f, 0.392f, 1.0f};
    GLfloat bahan_diffuse[] = {0.784f, 0.215f, 0.392f, 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);
    glRotated(1, 1, 1, 1);
    glTranslatef(0, 0.0, -5.0);
    glScaled(3, 3, 3);
    glRotated(90, 1, 1, 1);

    glutSolidTetrahedron();

    glFlush();
}

void Display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    tetraHedron();
    glutSwapBuffers();
}

void init(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, 10.0);
```

```
    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("2206050 - Yoga Agustiansyah");
    glutDisplayFunc(Display);
    init();
    glutMainLoop();
    return 0;
}
```

Output

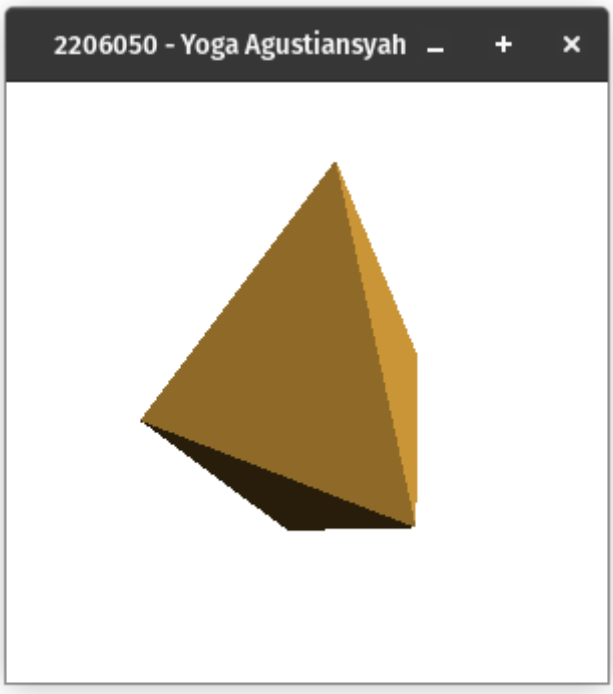
Warna tetrahedron diubah menjadi rgba(200, 55, 180, 1)

d) Octahedron

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

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

void octaHedron(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);

    // Objek Tetrahedron
    GLfloat bahan_ambient[] = {0.784f, 0.576f, 0.215f, 1.0f};
    GLfloat bahan_diffuse[] = {0.784f, 0.576f, 0.215f, 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);
    glTranslatef(0, 0.0, -5.0);
    glScaled(2, 2, 2);
    glRotated(45, 1, 1, 0);

    glutSolidOctahedron();

    glFlush();
}

void Display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    octaHedron();
    glutSwapBuffers();
}

void init(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, 10.0);
    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("2206050 - Yoga Agustiansyah");
    glutDisplayFunc(Display);
    init();
    glutMainLoop();
    return 0;
}
```

Output



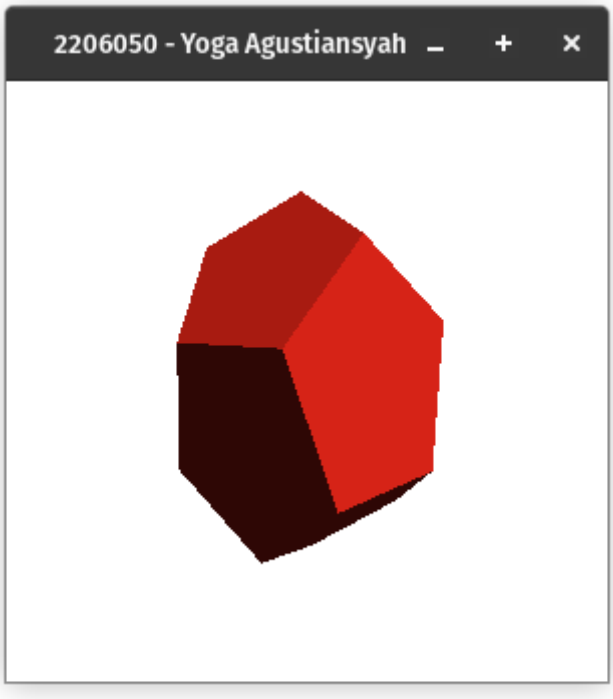
Warna octahedron diubah menjadi rgba(200, 147, 55, 1).

e) Dodecahedron

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

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

void dodecaHedron(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);

    // Objek Tetrahedron
    GLfloat bahan_ambient[] = {0.905f, 0.145f, 0.094f, 1.0f};
    GLfloat bahan_diffuse[] = {0.905f, 0.145f, 0.094f, 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);
    glTranslatef(0, 0.0, -5.0);
    glRotated(45, 1, 1, 0);

    glutSolidDodecahedron();

    glFlush();
}

void Display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    dodecaHedron();
    glutSwapBuffers();
}

void init(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, 10.0);
    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("2206050 - Yoga Agustiansyah");
    glutDisplayFunc(Display);
    init();
    glutMainLoop();
    return 0;
}
```

Output



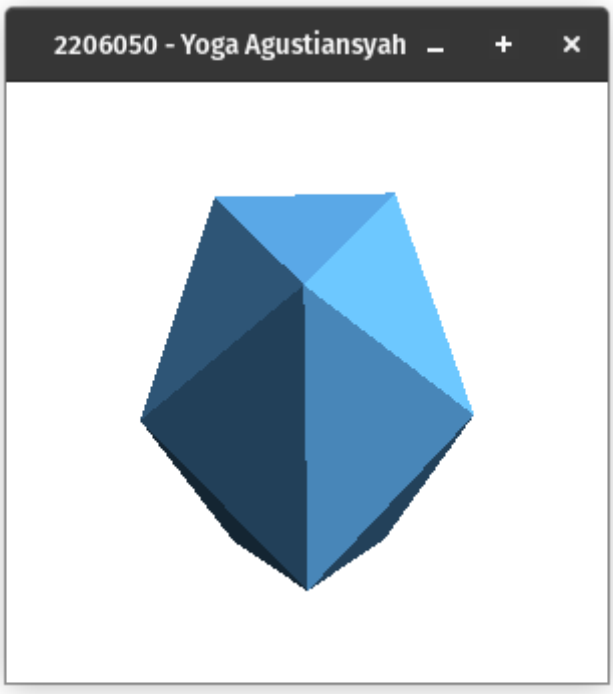
Warna dodecahedron diubah menjadi rgba(231, 37, 24, 1).

f) Icosahedron

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

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

void icosaHedron(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);

    // Objek Tetrahedron
    GLfloat bahan_ambient[] = {0.384f, 0.717f, 0.988f, 1.0f};
    GLfloat bahan_diffuse[] = {0.384f, 0.717f, 0.988f, 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);
    glTranslatef(0, 0.0, -5.0);
    glScaled(2, 2, 2);
    glRotated(45, 1, 1, 0);
    glutSolidIcosahedron();

    glFlush();
    // glutPostRedisplay();
}

void Display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    icosaHedron();
    glutSwapBuffers();
}

void init(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, 10.0);
    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("2206050 - Yoga Agustiansyah");
    glutDisplayFunc(Display);
    init();
    glutMainLoop();
    return 0;
}
```

Output



Warna icosahedron diubah menjadi rgba(98, 183, 252, 1).

i) Disatukan

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

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

void pentagonPrism(void)
{
    // isi function pentagonPrism sebelumnya

}

void pentagonAntiPrism(void)
{
    // isi function pentagonAntiPrism sebelumnya
}

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[][6] = {
    {0.784f, 0.215f, 0.392f, 1.0f},
    {0.784f, 0.576f, 0.215f, 1.0f},
    {0.905f, 0.145f, 0.094f, 1.0f},
    {0.384f, 0.717f, 0.988f, 1.0f},
    {0.215f, 0.423f, 0.784f, 1.0f},
    {0.215f, 0.784f, 0.294f, 1.0f}
    };
    GLfloat bahan_diffuse[][4] = {
    {0.784f, 0.215f, 0.392f, 1},
    {0.784f, 0.576f, 0.215f, 1},
    {0.905f, 0.145f, 0.094f, 1},
    {0.384f, 0.717f, 0.988f, 1},
    {0.215f, 0.423f, 0.784f, 1},
    {0.215f, 0.784f, 0.294f, 1}
};

    GLfloat bahan_specular[] = {1.0f, 1.0f, 1.0f, 1.0f};
    GLfloat bahan_shininess[] = {90.0f};

    glMaterialfv(GL_FRONT, GL_AMBIENT, bahan_ambient[0]);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, bahan_diffuse[0]);
    glMaterialfv(GL_FRONT, GL_SPECULAR, bahan_specular);
    glMaterialfv(GL_FRONT, GL_SHININESS, bahan_shininess);

    glPushMatrix();
    glTranslatef(-3.0, 1.0, -8.0);
```

```
    glutSolidTetrahedron();
    glPopMatrix();

    glMaterialfv(GL_FRONT, GL_AMBIENT, bahan_ambient[1]);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, bahan_diffuse[1]);
    glMaterialfv(GL_FRONT, GL_SPECULAR, bahan_specular);
    glMaterialfv(GL_FRONT, GL_SHININESS, bahan_shininess);
    glPushMatrix();
    glTranslatef(2.0, 1.0, -8.0);
    glutSolidOctahedron();
    glPopMatrix();

    glMaterialfv(GL_FRONT, GL_AMBIENT, bahan_ambient[2]);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, bahan_diffuse[2]);
    glMaterialfv(GL_FRONT, GL_SPECULAR, bahan_specular);
    glMaterialfv(GL_FRONT, GL_SHININESS, bahan_shininess);
    glPushMatrix();
    glTranslatef(2.5f, -2.5f, -8.0);
    glutSolidDodecahedron();
    glPopMatrix();

    glMaterialfv(GL_FRONT, GL_AMBIENT, bahan_ambient[3]);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, bahan_diffuse[3]);
    glMaterialfv(GL_FRONT, GL_SPECULAR, bahan_specular);
    glMaterialfv(GL_FRONT, GL_SHININESS, bahan_shininess);
    glPushMatrix();
    glTranslatef(-4.0, -2.5f, -8.0);
    glutSolidIcosahedron();
    glPopMatrix();


    glMaterialfv(GL_FRONT, GL_AMBIENT, bahan_ambient[4]);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, bahan_diffuse[4]);
    glMaterialfv(GL_FRONT, GL_SPECULAR, bahan_specular);
    glMaterialfv(GL_FRONT, GL_SHININESS, bahan_shininess);
    glPushMatrix();
    glTranslatef(-2.0, 3.0f, -8.0);
    glScalef(0.2f, 0.2f, 0.2f);
    pentagonPrism();
    glPopMatrix();

    glMaterialfv(GL_FRONT, GL_AMBIENT, bahan_ambient[5]);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, bahan_diffuse[5]);
    glMaterialfv(GL_FRONT, GL_SPECULAR, bahan_specular);
    glMaterialfv(GL_FRONT, GL_SHININESS, bahan_shininess);
    glPushMatrix();
    glTranslatef(2.0, 3.0f, -8.0);
    glScalef(0.2f, 0.2f, 0.2f);
    pentagonAntiPrism();
    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("2206050 - Yoga Agustiansyah");
    glutDisplayFunc(display);
    inisialisasi();
    glutMainLoop();
    return 0;
}
```

Output

Untuk kode lengkap semua programnya, bisa diakses di :
https://github.com/yoga220802/Praktikum_Grafkom/tree/main/Pertemuan%206/tasks

III. Kesimpulan

Dalam praktikum ini, kita telah mempelajari dasar-dasar pemodelan 3D dengan jaring poligon menggunakan OpenGL dan GLUT. kita telah menjelajahi konsep dasar pemodelan 3D dengan bentuk geometris dasar seperti prisma segi lima, antiprisma segi lima, tetrahedron, octahedron, dodecahedron, dan icosahedron.

Selama praktikum, kita telah mempelajari konsep pencahayaan dan material yang memengaruhi tampilan objek 3D. kita juga telah melihat bagaimana menggambar objek 3D dengan penggunaan glBegin dan glEnd untuk mendefinisikan poligon dan permukaan. Modifikasi warna objek telah kita lakukan untuk memberikan tampilan yang lebih menarik.

Dengan menyelesaikan praktikum ini, kita memiliki pemahaman dasar tentang pemodelan 3D dan penggunaan OpenGL dalam pengembangan aplikasi grafis. kita siap untuk menjelajahi konsep yang lebih canggih dalam dunia grafik komputer dan membangun aplikasi grafis yang lebih kompleks di masa depan.