# MAKALAH
# COMPUTER GRAPHICS

# MENDESAIN RUMAH 3D

**Nama : Stevani Maria Ayu Rajagukguk**

**NPM : 230210038**

**Dosen : Sunarsan Sitohang, S.Kom., M.TI**

**PROGRAM STUDI TEKNIK INFORMATIKA**
**FAKULTAS TEKNIK DAN KOMPUTER**
**UNIVERSITAS PUTERA BATAM**
**2025**

# KATA PENGANTAR

Puji syukur atas limpahan rahmat dan karunia-NYA sehingga Pedoman Penulisan Tugas Mandiri ini dapat diselesaikan dengan baik walaupun masih terdapat kekurangan namun diharapkan dapat diperbaiki kedepannya.

Tugas Mandiri (TM) disusun menurut kaidah keilmuan dan ditulis berdasarkan kaidah Bahasa Indonesia di bawah pengawasan atau pengarahan dosen pengampu untuk memenuhi kriteria-kriteria kualitas yang telah ditetapkan sesuai keilmuannya masing-masing. Tugas Mandiri dibuat sebagai salah satu persyaratan untuk menyelesaikan suatu mata kuliah di Universitas Putera Batam (UPB). Dalam upaya mendokumentasikan seluruh Tugas Mandiri mahasiswa, diperlukan Pedoman Penulisan Tugas Mandiri yang dapat digunakan di semua fakultas di lingkungan Universitas Putera Batam. Pedoman ini disusun oleh Tim Penyusun Pedoman Penulisan Tugas Mandiri dari LPPM UPB dengan tujuan memberikan tuntunan kepada mahasiswa dan pedoman bagi semua fakultas. Tim Penyusun memberi kesempatan kepada Program Studi/Fakultas untuk membuat petunjuk tambahan mengenai hal-hal yang tidak diatur dalam pedoman ini.

Semoga dengan adanya Panduan Penulisan Tugas Mandiri ini maka penyelesaian Tugas Mandiri bagi setiap mahasiswa UPB dapat lebih lancar.

Batam, 30 June 2025

Ketik nama penyusun.

# DAFTAR ISI

# BAB I PENDAHULUAN

## 1.1 Latar Belakang

Di era modern ini, kebutuhan akan visualisasi desain dalam bidang arsitektur dan konstruksi semakin meningkat. Visualisasi tersebut digunakan untuk membantu klien dalam memahami konsep bangunan yang akan dibuat sebelum proses pembangunan dimulai. Salah satu cara untuk menampilkan visualisasi adalah dengan menggunakan pemrograman grafika komputer.

OpenGL merupakan library grafika komputer yang banyak digunakan untuk pembuatan objek 2D dan 3D. Dengan OpenGL, kita dapat membuat berbagai macam objek tiga dimensi, termasuk rumah minimalis, secara fleksibel dan interaktif. Dalam makalah ini, penulis mencoba untuk membuat desain rumah 3D minimalis sederhana yang dapat dirotasi menggunakan mouse, sehingga memberikan pengalaman interaktif kepada pengguna.

## 1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, maka rumusan masalah dalam makalah ini adalah:

1. Bagaimana membuat desain rumah minimalis 3D menggunakan OpenGL?
2. Bagaimana cara menambahkan interaksi rotasi objek dengan mouse?
3. Apa saja fungsi-fungsi penting dalam pembuatan desain rumah 3D ini?

## 1.3 Tujuan

Tujuan penulisan makalah ini adalah:

1. Untuk membuat desain rumah minimalis 3D menggunakan OpenGL.
2. Untuk mengimplementasikan fungsi interaksi rotasi objek menggunakan mouse.
3. Untuk memahami penggunaan fungsi-fungsi dasar OpenGL dalam pemodelan objek 3D.

# BAB II TINJAUAN PUSTAKA

## 2.1 OpenGL

OpenGL (Open Graphics Library) adalah sebuah API standar industri untuk membuat grafis 2D dan 3D yang bersifat open source dan multiplatform. OpenGL digunakan pada berbagai aplikasi grafika komputer, mulai dari CAD, game development, hingga visualisasi data ilmiah.

## 2.2 GLUT

GLUT (OpenGL Utility Toolkit) adalah library tambahan yang menyediakan fungsi untuk membuat window, menerima input keyboard, dan mouse secara lebih mudah. GLUT sangat membantu programmer dalam pembuatan aplikasi grafika berbasis OpenGL tanpa perlu menulis kode pengaturan window secara manual.

## 2.3 Transformasi 3D

Dalam grafika komputer, transformasi objek 3D meliputi translasi (perpindahan posisi), rotasi (perputaran), dan scaling (perbesaran/perkecilan). Pada desain rumah ini, transformasi rotasi digunakan agar rumah dapat dilihat dari berbagai sudut pandang.

## 2.4 Perancangan Rumah Minimalis

Rumah minimalis merupakan konsep desain rumah yang menekankan kesederhanaan bentuk, penggunaan warna netral, serta penataan ruangan yang efisien. Dalam desain grafika komputer, rumah minimalis biasanya dibuat dengan bentuk geometri dasar seperti kubus dan prisma segitiga yang dikombinasikan sehingga tampak modern dan profesional.

# BAB III METODOLOGI

## 3.1 Alat dan Bahan

- Laptop/PC dengan sistem operasi Windows atau Linux
- Compiler C/C++ (Code::Blocks atau Visual Studio)
- Library OpenGL dan GLUT

## 3.2 Tahapan Implementasi Program

1. **Inisialisasi Window**

   Menggunakan glutInitWindowSize() dan glutInitWindowPosition() untuk mengatur ukuran dan posisi window program.

2. **Pengaturan Dasar (myinit)**

   Fungsi myinit() berisi pengaturan warna background, mode proyeksi, serta mengaktifkan depth test.

3. **Fungsi Ukur (ukur)**

   Mengatur perspektif kamera dengan fungsi gluPerspective() dan gluLookAt() agar tampilan rumah terlihat proporsional dan nyata.

4. **Fungsi Mouse dan Motion**

   - Fungsi mouse() untuk mendeteksi klik mouse.
   - Fungsi motion() untuk mendeteksi pergerakan mouse dan mengatur rotasi objek.

5. **Fungsi Tampilan (tampilan)**

   Membuat semua objek rumah menggunakan glBegin(GL_POLYGON) di antaranya:

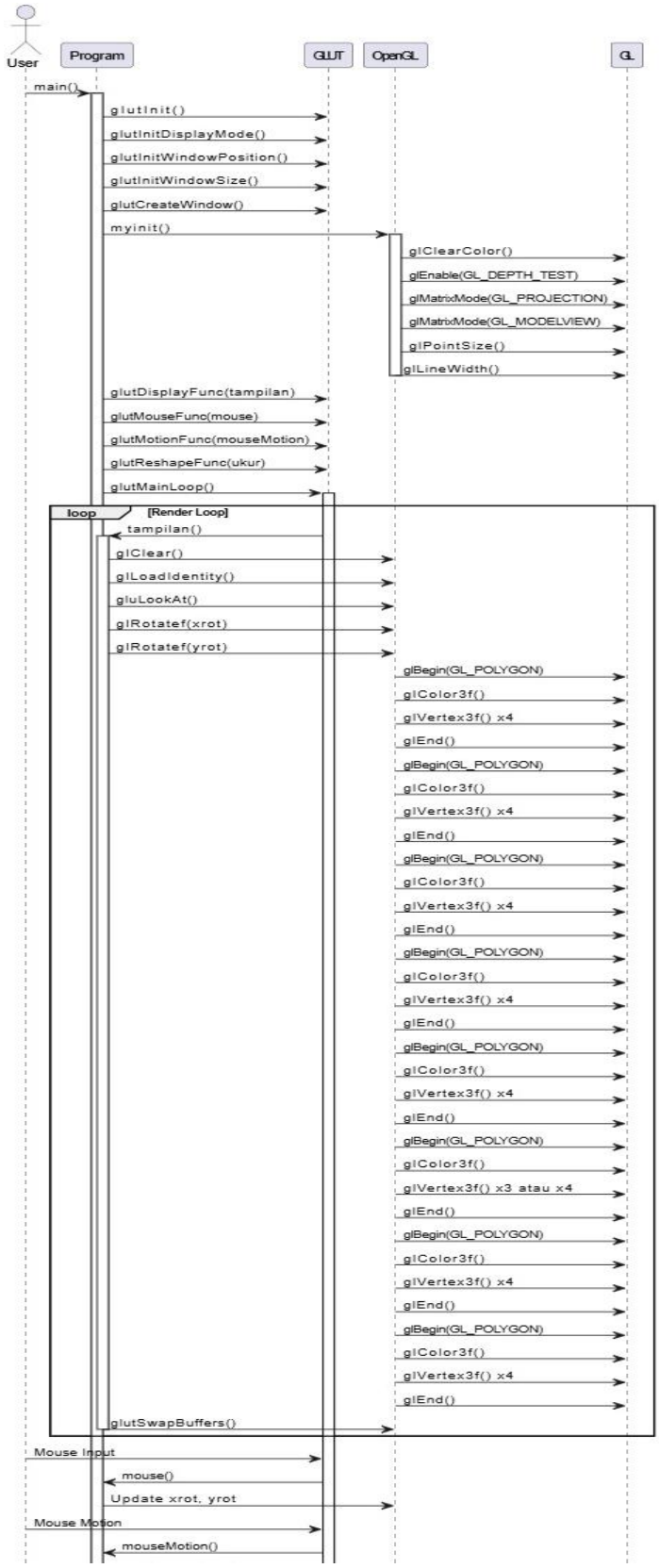
   - Lantai dan tanah
   - Tiang rumah
   - Ruangan 1, 2, 3
   - Atap datar dan segitiga

- Jendela dan pintu

6. **Perulangan Program**

   Menggunakan glutMainLoop() agar program berjalan terus-menerus menampilkan desain rumah.

## 3.3 Diagram Alir Program



# BAB IV HASIL DAN PEMBAHASAN

## 4.1 Hasil Implementasi

Program berhasil membuat desain rumah minimalis 3D yang terdiri atas:

- **Lantai dan Tanah**: dibuat dengan polygon berwarna hijau dan abu-abu sebagai alas rumah.
- **Tiga Ruangan Utama**: dibuat dengan kombinasi kubus yang disusun berdampingan dan saling terhubung.
- **Tiang Rumah**: dibuat dengan polygon panjang pada sudut depan rumah.
- **Atap**: menggunakan dua variasi, atap datar dan atap segitiga (prisma segitiga).
- **Jendela dan Pintu**: dibuat dengan polygon berwarna berbeda pada bagian dinding rumah.

## 4.2 Pembahasan dan Analisis

- Fungsi **glRotatef** digunakan untuk mengatur rotasi rumah sesuai pergerakan mouse.
- Fungsi **glColor3f** digunakan untuk pewarnaan tiap bagian rumah agar tampak lebih realistis.
- Fungsi **gluLookAt** digunakan untuk mengatur sudut pandang kamera sehingga rumah tampak dari perspektif 3D yang proporsional.
- Mouse dikonfigurasi untuk mendeteksi klik kiri sebagai trigger rotasi objek, sedangkan gerakan mouse menentukan sudut rotasi pada sumbu X dan Y.

Program berjalan dengan lancar tanpa error, tampilan rumah stabil dan dapat dirotasi 360 derajat, sehingga pengguna dapat melihat desain rumah dari segala sisi.

### 4.3 Koding

```
#include<gL/freeglut.h> // atau include<gL/glut.h> yang umum untuk pemanggilan glut.

//Deklarasi fungsi Mouse agar gambar 3d dapat diputar putar menggunakan Mouse
float xrot = 0;
float yrot = 0;
float xdiff = 0;
float ydiff = 0;
bool mouseDown = false;

//Deklarasi pengaturan lembaran kerja agar Gambar 3d yang kita buat saat diputar atau di geser tidak kemana mana
void ukur(int lebar, int tinggi) {
if (tinggi == 0) tinggi = 1;
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(45,lebar/tinggi, 5, 450);
glTranslatef(0,0,-300);// jarak object dari lembaran kerja
glMatrixMode(GL_MODELVIEW);
}
void myinit(void) {
glClearColor(0.0,0.0,0.0,0.0);
glMatrixMode(GL_PROJECTION);
glEnable(GL_DEPTH_TEST);

glMatrixMode(GL_MODELVIEW);
glPointSize(10.0);
glLineWidth(7.0f);
}
```

```
//Dan selanjutnya yaitu fungsi mouse
void idle()
{
if (!mouseDown)
{
xrot += 0.3;
yrot += 0.4;
}
glutPostRedisplay();
}

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

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

```
}

//Dibawah ini dimulai koding untuk membuat object
void tampilan(void) {
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glLoadIdentity();
gluLookAt(0,0,3,0,0,0,0,1,0);
glRotatef(xrot, 1, 0,0);
glRotatef(yrot, 0,1,0);
glPushMatrix();

//LUAS TANAH
glBegin(GL_POLYGON);
glColor3f(1, 0.5, 0);
glVertex3f(-80,-30,80);
glVertex3f(80,-30,80);
glVertex3f(80,-30,-80);
glVertex3f(-80, -30, -80);
glEnd();

//LUAS BANGUNAN
glBegin(GL_POLYGON);
glColor3f(1,1,0.5);
glVertex3f(-60,-29.5,60);
glVertex3f(60,-29.5,60);
glVertex3f(60,-29.5,-60);
glVertex3f(-60,-29.5,-60);
glEnd();

//TIANG 1
glBegin(GL_POLYGON);
```

```
glColor3f(0,1,0.5);
glVertex3f(-20,-30,60);
glVertex3f(-10,-30,60);
glVertex3f(-10,10,60);
glVertex3f(-20,10,60);
glEnd();
glBegin(GL_POLYGON);
glColor3f(0,1,0.5);
glVertex3f(-10,-30,60);
glVertex3f(-10,-30,50);
glVertex3f(-10,10,50);
glVertex3f(-10,10,60);
glEnd();
glBegin(GL_POLYGON);
glColor3f(0,1,0.5);
glVertex3f(-20,-30,50);
glVertex3f(-10,-30,50);
glVertex3f(-10,10,50);
glVertex3f(-20,10,50);
glEnd();
glBegin(GL_POLYGON);
glColor3f(0,1,0.5);
glVertex3f(-20,-30,60);
glVertex3f(-20,-30,50);
glVertex3f(-20,10,50);
glVertex3f(-20,10,60);
glEnd();

//TIANG 2
glBegin(GL_POLYGON);
glColor3f(0,1,0.5);
```

```
glVertex3f(20,-30,60);
glVertex3f(10,-30,60);
glVertex3f(10,10,60);
glVertex3f(20,10,60);
glEnd();
glBegin(GL_POLYGON);
glColor3f(0,1,0.5);
glVertex3f(10,-30,60);
glVertex3f(10,-30,50);
glVertex3f(10,10,50);
glVertex3f(10,10,60);
glEnd();
glBegin(GL_POLYGON);
glColor3f(0,1,0.5);
glVertex3f(20,-30,50);
glVertex3f(10,-30,50);
glVertex3f(10,10,50);
glVertex3f(20,10,50);
glEnd();
glBegin(GL_POLYGON);
glColor3f(0,1,0.5);
glVertex3f(20,-30,60);
glVertex3f(20,-30,50);
glVertex3f(20,10,50);
glVertex3f(20,10,60);
glEnd();

//RUANGAN 1
glBegin(GL_POLYGON);
glColor3f(0, 0.5, 0);
glVertex3f(-60,-30,40);
```

```
glVertex3f(-20,-30,40);
glVertex3f(-20,20,40);
glVertex3f(-60,20,40);
glEnd();
glBegin(GL_POLYGON);
glColor3f(0,1,0.5);
glVertex3f(-20,-30,40);
glVertex3f(-20,-30,-30);
glVertex3f(-20,20,-30);
glVertex3f(-20,20,40);
glEnd();
glBegin(GL_POLYGON);
glColor3f(0, 0.5, 0);
glVertex3f(-60,-30,-30);
glVertex3f(-20,-30,-30);
glVertex3f(-20,20,-30);
glVertex3f(-60,20,-30);
glEnd();
glBegin(GL_POLYGON);
glColor3f(0, 0.5, 0);
glVertex3f(-60,-30,40);
glVertex3f(-60,-30,-30);
glVertex3f(-60,20,-30);
glVertex3f(-60,20,40);
glEnd();

// RUANGAN 2
glBegin(GL_POLYGON);
glColor3f(0, 0.5, 0);
glVertex3f(20,-30,20);
glVertex3f(-20,-30,20);
```

```
glVertex3f(-20,20,20);
glVertex3f(20,20,20);
glEnd();
glBegin(GL_POLYGON);
glColor3f(0,1,0.5);
glVertex3f(-20,-30,20);
glVertex3f(-20,-30,-60);
glVertex3f(-20,20,-60);
glVertex3f(-20,20,20);
glEnd();
glBegin(GL_POLYGON);
glColor3f(0, 0.5, 0);
glVertex3f(20,-30,-60);
glVertex3f(-20,-30,-60);
glVertex3f(-20,20,-60);
glVertex3f(20,20,-60);
glEnd();
glBegin(GL_POLYGON);
glColor3f(0,0.5,0);
glVertex3f(20,-30,20);
glVertex3f(20,-30,-60);
glVertex3f(20,20,-60);
glVertex3f(20,20,20);
glEnd();


// RUANGAN 3
glBegin(GL_POLYGON);
glColor3f(0, 0.5, 0);
glVertex3f(60,-30,40);
glVertex3f(20,-30,40);
```

```
glVertex3f(20,20,40);
glVertex3f(60,20,40);
glEnd();
glBegin(GL_POLYGON);
glColor3f(0,1,0.5);
glVertex3f(20,-30,40);
glVertex3f(20,-30,-60);
glVertex3f(20,20,-60);
glVertex3f(20,20,40);
glEnd();
glBegin(GL_POLYGON);
glColor3f(0, 0.5, 0);
glVertex3f(60,-30,-60);
glVertex3f(20,-30,-60);
glVertex3f(20,20,-60);
glVertex3f(60,20,-60);
glEnd();
glBegin(GL_POLYGON);
glColor3f(0, 0.5, 0);
glVertex3f(60,-30,40);
glVertex3f(60,-30,-60);
glVertex3f(60,20,-60);
glVertex3f(60,20,40);
glEnd();

// BAGIAN DIATAS TIANG
glBegin(GL_POLYGON);
glColor3f(0,0.5,0);
glVertex3f(20,20,60);
glVertex3f(-20,20,60);
glVertex3f(-20,20,20);
```

```
glVertex3f(20,20,20);
glEnd();
glBegin(GL_POLYGON);
glColor3f(0,1,0.5);
glVertex3f(20,10,60);
glVertex3f(-20,10,60);
glVertex3f(-20,20,60);
glVertex3f(20,20,60);
glEnd();
glBegin(GL_POLYGON);
glColor3f(0,1,0.5);
glVertex3f(20,10,60);
glVertex3f(20,10,40);
glVertex3f(20,20,40);
glVertex3f(20,20,60);
glEnd();
glBegin(GL_POLYGON);
glColor3f(0,1,0.5);
glVertex3f(-20,10,60);
glVertex3f(-20,10,40);
glVertex3f(-20,20,40);
glVertex3f(-20,20,60);
glEnd();

//BAGIAN ATAP TENGAH
glBegin(GL_POLYGON);
glColor3f(1, 0, 0);
glVertex3f(-20,20,60);
glVertex3f(20,20,60);
glVertex3f(0,40,60);
glEnd(); //tambahkan ini sebelum lanjut
```

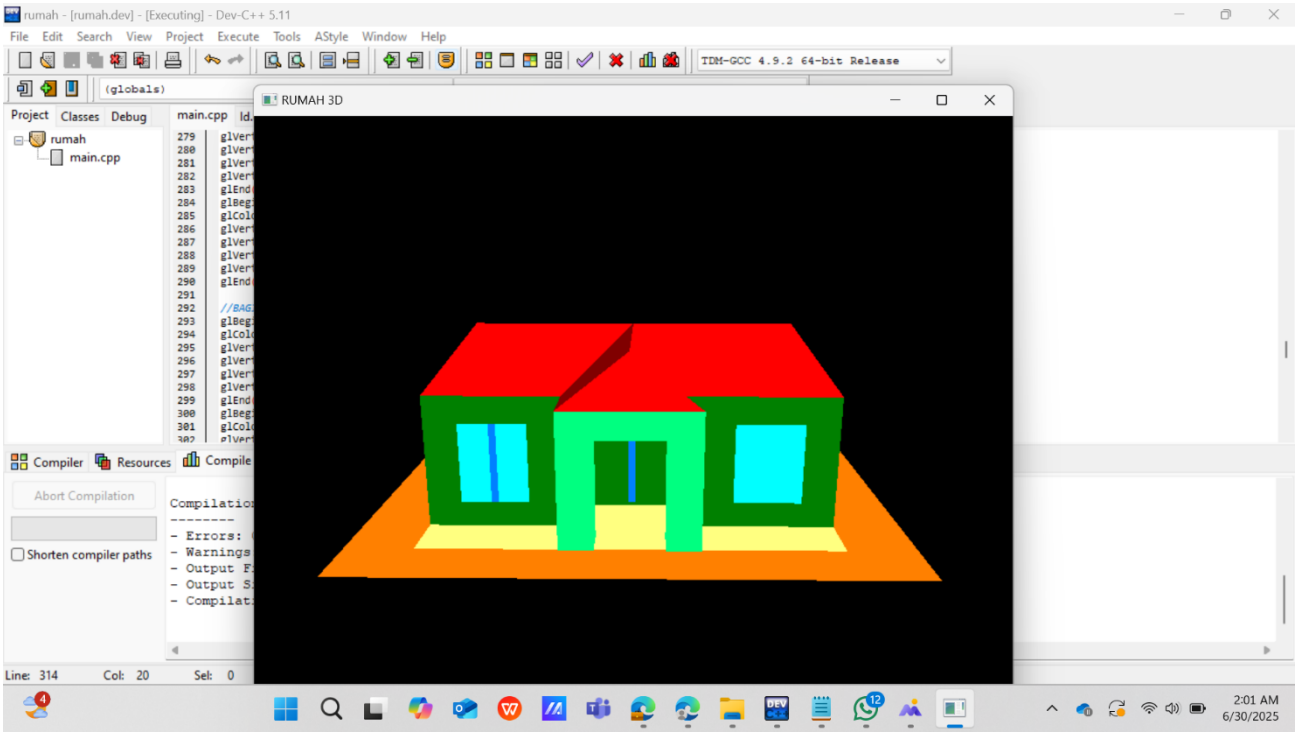
```
glVertex3f(60, 20, 40);
glColor3f(0.5,0,0);
glVertex3f(20,20,60);
glVertex3f(20,20,0);
glVertex3f(0,40,0);
glVertex3f(0,40,60);
glEnd();
glBegin(GL_POLYGON);
glColor3f(0.5,0,0);
glVertex3f(-20,20,60);
glVertex3f(-20,20,0);
glVertex3f(0,40,0);
glVertex3f(0,40,60);
glEnd();

//BAGIAN ATAP BELAKANG
glBegin(GL_POLYGON);
glColor3f(1,0,0);
glVertex3f(60,20,40);
glVertex3f(-60,20,40);
glVertex3f(-50,40,0.5);
glVertex3f(50,40,0.5);
glEnd();
glBegin(GL_POLYGON);
glColor3f(1, 0, 0);
glVertex3f(60,20,-30);
glVertex3f(-60,20,-30);
glVertex3f(-50,40,0.5);
glVertex3f(50,40,0.5);
glEnd();
glBegin(GL_POLYGON);
```

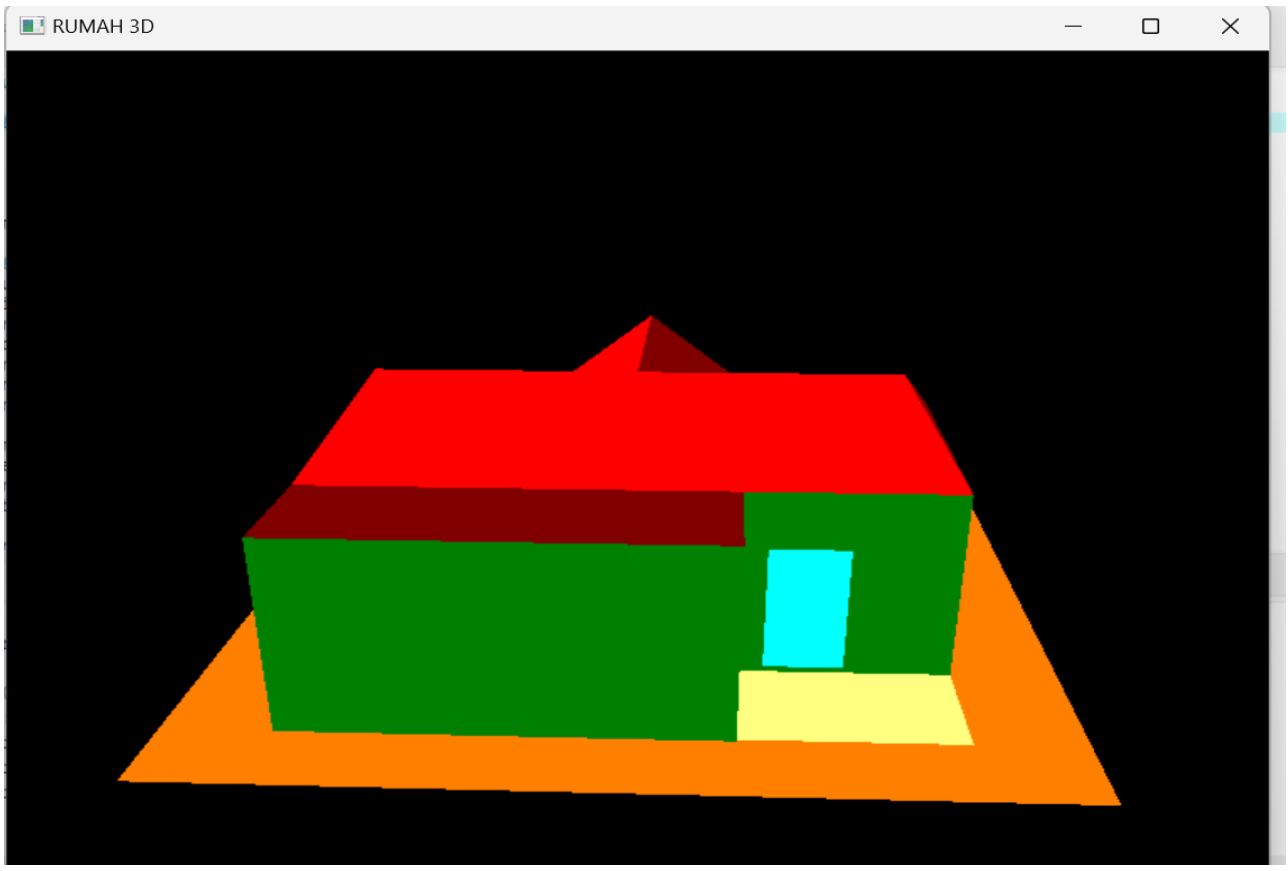
```
glColor3f(0.5,0,0);
glVertex3f(60,20,-30);
glVertex3f(50,40,0.5);
glVertex3f(60,20,40);
glEnd();
glBegin(GL_POLYGON);
glColor3f(0.5,0,0);
glVertex3f(-60,20,-30);
glVertex3f(-50,40,0.5);
glVertex3f(-60,20,40);
glEnd();

//ATAP DATAR BELAKANG
glBegin(GL_POLYGON);
glColor3f(0.5,0,0);
glVertex3f(-20,20,-30);
glVertex3f(60,20,-30);
glVertex3f(60,20,-60);
glVertex3f(-20,20,-60);
glEnd();

//JENDELA
glBegin(GL_POLYGON);
glColor3f(0, 1, 1);
glVertex3f(-50,-20,40.5);
glVertex3f(-30,-20,40.5);
glVertex3f(-30,10,40.5);
glVertex3f(-50,10,40.5);
glEnd();
glBegin(GL_LINES);
glColor3f(0, 0.5, 1);
```

```
glVertex3f(-40,-20,40.6);
glVertex3f(-40,10,40.6);
glEnd();

//JENDELA
glBegin(GL_POLYGON);
glColor3f(0, 1, 1);
glVertex3f(50,-20,40.5);
glVertex3f(30,-20,40.5);
glVertex3f(30,10,40.5);
glVertex3f(50,10,40.5);
glEnd();
glVertex3f(-40,10,40.6);
glEnd();

//JENDELA
glBegin(GL_POLYGON);
glColor3f(0, 1, 1);
glVertex3f(50,-20,40.5);
glVertex3f(30,-20,40.5);
glVertex3f(30,10,40.5);
glVertex3f(50,10,40.5);
glEnd();
glBegin(GL_LINES);
glColor3f(0, 0.5, 1);
glEnd();
glBegin(GL_LINES);
glColor3f(0, 0.5, 1);
glVertex3f(0,-28,20.6);
glVertex3f(0,5,20.6);
glEnd();
```

```
//PINTU BELAKANG
glBegin(GL_POLYGON);
glColor3f(0, 1, 1);
glVertex3f(-40,-28,-30.1);
glVertex3f(-25,-28,-30.1);
glVertex3f(-25,5,-30.1);
glVertex3f(-40,5,-30.1);
glEnd();

glPushMatrix();
glPopMatrix();
glutSwapBuffers();
}

int main(int argc, char **argv){
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
   glutInitWindowPosition(240, 80);
   glutInitWindowSize(750, 600);
   glutCreateWindow("RUMAH 3D");
   myinit();
   glutDisplayFunc(tampilan);
   glutMouseFunc(mouse);
   glutMotionFunc(mouseMotion);
   glutReshapeFunc(ukur);
   glutMainLoop();
}
```

## 4.4 Hasil Gambar Koding



# BAB V PENUTUP

## 5.1 Kesimpulan

Dari hasil pembuatan program desain rumah 3D ini, dapat disimpulkan:

1. OpenGL mampu digunakan untuk membuat desain rumah 3D sederhana dengan bentuk geometri dasar.
2. Program berhasil menampilkan rumah minimalis 3D yang dapat dirotasi menggunakan mouse.
3. Fungsi dasar OpenGL seperti polygon, transformasi, dan pengaturan kamera dapat digunakan secara efektif untuk visualisasi arsitektur.

## 5.2 Saran

Untuk pengembangan lebih lanjut, program dapat ditambahkan:

- **Tekstur dan Material**: agar rumah terlihat lebih nyata.
- **Animasi**: seperti pintu dapat terbuka dan tertutup.
- **Lighting dan Shadowing**: agar efek cahaya dan bayangan memberikan kesan realistis.
- **User Interface**: seperti tombol zoom in/out dan reset posisi kamera.

## DAFTAR PUSTAKA

Hearn, D. & Baker, M. P. (2010). *Computer Graphics with OpenGL*. Pearson.

Shreiner, D. (2013). *OpenGL Programming Guide (The Red Book)*. Addison-Wesley.

Documentation OpenGL: https://www.opengl.org/documentation/

Tutorialspoint. (2024). *OpenGL - Quick Guide*.