Prodi Informatika Fakukltas Rekayasa System

LAPORAN PRAKTIKUM

MATERI GRAFIKA KOMPUTER

Semester 4 Tahun Akademik 2022/2023

Oleh:

Nama: Rizki

Nim: 211001113

LABOLATORIUM INFORMATIKA

PRODI INFORMATIKA

FAKULTAS REKAYASA SYSTEM

UNIVERSITAS TEKNOLOGI SUMBAWA

TAHUN 2023

Kata pengantar

Puji syukur kami panjatkan kehadirat Allah SWT atas segala rahmat, hidayah, dan karunia-Nya sehingga kami dapat menyelesaikan laporan praktikum ini dengan baik. Laporan ini disusun sebagai hasil dari kegiatan praktikum yang dilaksanakan dalam rangka memperdalam pemahaman dan keterampilan kami dalam bidang tertentu.

Kami juga ingin mengucapkan terima kasih yang sebesar-besarnya kepada dosen pengampu, laboran, dan seluruh pihak yang telah memberikan bimbingan, arahan, serta dukungan selama pelaksanaan praktikum ini. Tanpa bantuan mereka, penyelesaian laporan ini tidak akan menjadi mungkin.

Praktikum merupakan salah satu aspek penting dalam pendidikan, yang memungkinkan kami untuk mengaplikasikan pengetahuan teoritis yang telah kami pelajari dalam lingkungan praktis. Melalui praktikum ini, kami berkesempatan untuk memperdalam pemahaman tentang prosedur kerja, teknik, dan metodologi yang relevan dengan bidang studi kami.

Laporan ini berisi hasil pengamatan, analisis, dan interpretasi data yang kami peroleh selama praktikum. Kami telah melakukan serangkaian percobaan, pengukuran, atau observasi yang bertujuan untuk memvalidasi teori yang telah kami pelajari sebelumnya. Hasil-hasil yang tercantum di dalam laporan ini menjadi bukti nyata dari dedikasi dan kerja keras kami selama pelaksanaan praktikum.

Kami menyadari bahwa laporan ini belum sempurna dan masih memiliki keterbatasan-keterbatasan tertentu. Oleh karena itu, kami sangat mengharapkan masukan, saran, dan kritik yang konstruktif dari pembaca. Masukan tersebut akan sangat berharga bagi kami dalam meningkatkan pemahaman dan kualitas penulisan kami di masa depan.

Kami berharap laporan praktikum ini dapat bermanfaat bagi pembaca, terutama bagi mereka yang tertarik dalam bidang yang sama. Semoga laporan ini dapat memberikan wawasan dan pemahaman yang lebih baik tentang topik yang dibahas, serta menginspirasi untuk melakukan penelitian lebih lanjut di masa mendatang.

Akhir kata, kami berharap laporan praktikum ini dapat memenuhi harapan semua pihak yang terlibat. Segala kesalahan dan kekurangan yang mungkin ada adalah murni tanggung jawab kami, dan kami siap menerima kritik membangun guna perbaikan ke depan. Terima kasih atas perhatian dan kesempatan yang diberikan.

Hormat kami,

Universitas Teknologi Sumbawa

BAB I

BASIC OPEN GL

1.1.Tujuan praktikum

Praktikum ini diharapkan dapat Menguasai dasar-dasar grafika komputer dengan pembelajaran Open Gl memberikan pemahaman tentang prinsip-prinsip dasar grafika komputer, termasuk koordinat 2D dan 3D, pencahayaan, tekstur, transformasi, dan rasterisasi. Ini membantu pengembang untuk memahami bagaimana objek dan lingkungan 3D direpresentasikan dan dirender dalam komputer.

1.2.Dasar teori

1.2.1. Pengenalan Open Gl

Open Gl (Open Graphics Library) adalah sebuah API (Application Programming Interface) yang digunakan untuk membuat aplikasi grafis, terutama dalam pengembangan permainan komputer, simulasi, visualisasi data, dan aplikasi grafis interaktif lainnya. Open Gl memberikan serangkaian fungsi dan prosedur yang memungkinkan pengembang untuk menggambar objek dan lingkungan 2D dan 3D dengan cepat dan efisien.

Open Gl dikembangkan oleh Khronos Group, sebuah organisasi industri yang bertujuan untuk mengembangkan dan mendorong standar terbuka dalam bidang grafis dan multimedia. Open Gl dirancang untuk bekerja secara lintas platform, artinya dapat digunakan pada berbagai sistem operasi seperti Windows, macOS, Linux, dan platform mobile seperti Android dan iOS.

Open Gl juga sering digunakan bersama dengan library atau framework lain seperti GLUT (Open Gl Utility Toolkit), GLFW (Graphics Library Framework), atau Qt untuk mempermudah pengembangan aplikasi grafis dengan Open Gl.

Karena fleksibilitas, performa, dan dukungan lintas platformnya, Open Gl tetap menjadi salah satu API grafis yang populer dan digunakan secara luas dalam industri pengembangan permainan, simulasi, dan aplikasi grafis interaktif.

1.2.2. Sintaks perintah Open Gl

Sintaks perintah Open Gl mengikuti aturan panulisan dari library dimana fungsi tersebut berasal, format penulisan fungsi Open Gl adalah : <awalan library><perintah><optional jumlah argumen><optional tipe argumen>

Semua printah Open Gl menggunakan awalan gl diikuti dengan huruf kapital pada setiap kata membentuk nama perintah (sebagai contoh glClearColor). Untuk mendefinisikan konstanta diawali dengan GL_ dengan menggunakan huruf capital dan garis bawah untuk memisahkan kata (seperti GL_POLY_STIPPLE). Terkadang beberapa huruf dan angka ditambahkan pada akhir perintah (seperti 3f pada glVertex3f). Dalam hal ini angka 3 menunjukan berapa banyak argument yang harus ada pada perintah tersebut dan akhiran huruf f menunjukan jenis datanya yaitu floating. Sebagai contoh pada dua perintah berikut ini :

glVertex3i(1,0,-2);

glVertex3f(1.0, 0.0, -2.0); I-2

Adalah sama yaitu meletakan titik dilayar pada koordinat x = 1, y = 0 dan z = -2, perbedaanya yaitu pada perintah pertama menspesifikasikan titik

dengan tipe data integer 32-bit, sedangkan yang kedua dengan tipe data single precision floating point.

Beberapa perintah Open Gl menambahkan perintah huruf akhir v yang menunjukan bahwa perintah tersebut menggunakan pointer ke array/vector. Dibawah ii adalah contoh perbedaanya.

```
Float color_array[]={1.0,0.0,0.0}
glColor3f(1.0,0.0,0.0);
glColor3fv(color_array);
```

1.2.3. Library yang berhubungan dengan Open Gl

Ada beberapa library yang berhubungan dengan Open Gl dan digunakan bersama dengan Open Gl untuk mempermudah pengembangan aplikasi grafis. Beberapa library tersebut antara lain:

1. GLUT (Open Gl Utility Toolkit): GLUT menyediakan fungsi-fungsi yang membantu dalam pembuatan jendela, penanganan input pengguna, dan manajemen waktu dalam aplikasi Open Gl. GLUT sangat berguna dalam membuat aplikasi sederhana dengan Open Gl, seperti membuat jendela, menangani input keyboard dan mouse, serta mengatur loop tampilan.
2. GLFW (Graphics Library Framework): GLFW adalah library yang dirancang untuk membuat jendela dan menangani input pengguna dalam konteks Open Gl. GLFW memiliki dukungan yang lebih modern dan fleksibel dibandingkan GLUT, dan sering digunakan dalam pengembangan permainan dan aplikasi real-time.
3. GLEW (Open Gl Extension Wrangler): GLEW adalah library yang membantu dalam memuat ekstensi Open Gl pada berbagai sistem operasi. GLEW mempermudah penggunaan ekstensi Open Gl yang lebih baru dan tidak terdokumentasi secara luas, sehingga memungkinkan pengembang untuk menggunakan fitur-fitur terbaru dari Open Gl.
4. GLM (Open Gl Mathematics): GLM adalah library matematika yang dirancang khusus untuk pengembangan aplikasi Open Gl. GLM menyediakan berbagai fungsi matematika yang diperlukan dalam pemrosesan vektor, matriks, transformasi, dan perhitungan lainnya dalam konteks Open Gl.
5. SOIL (Simple Open Gl Image Library): SOIL adalah library yang membantu dalam memuat, menyimpan, dan memanipulasi tekstur dalam aplikasi Open Gl. SOIL mendukung berbagai format gambar dan menyediakan fungsi-fungsi untuk memanipulasi tekstur, termasuk pembuatan tekstur dari file gambar, pengaturan parameter tekstur, dan lainnya.
6. Assimp (Open Asset Import Library): Assimp adalah library yang digunakan untuk mengimpor model dan data mesh dari berbagai format file 3D ke dalam aplikasi Open Gl. Assimp mendukung banyak format file populer, seperti OBJ, FBX, Collada, dan lainnya, dan menyediakan fungsi untuk membaca data mesh dan material dari file-file ini.

Selain library-library di atas, ada juga library lain seperti SDL (Simple DirectMedia Layer), Qt, FreeGLUT, dan banyak lagi yang dapat digunakan bersama dengan Open Gl untuk mempermudah pengembangan aplikasi grafis. Pilihan library yang tepat tergantung pada kebutuhan dan preferensi pengembang.

1.2.4. Membersihkan windows

Menggambar pada layar computer berbeda dengan menggambar pada kertas putih yang dari pabriknya sudah berwarna putih. Pada computer, memory untuk menampilkan gambar biasanya diisi dengan gambar yang berasal dari perintah gambar paling akhir, jadi perlu dibersihkan dengan warna latar belakang sebelum digambar lagi. Warna latar belakang yang dipilih tergantung dari aplikasi yang akan dibuat. Sintaks (glClearColor(Glclamp red, Glclamp green, Glclamp blue, Glclamp alpha) digunakan untuk memilih warna, yang akan digunakan untuk membersikan latar belakang dalam mode RGBA. Selanjutnya perintah glClear(Glbitfield mask) digunakan untuk membersikan buffer yang dispesifikasikan dengan warna yang telah ditentukan. Contoh berikut ini perintah yang digunakan untuk membersikan layar latar belakang dengan warna hitam dan buffer apa yang akan dibersikan. Dalam hal ini, buffer warna yang akan dibersikan karna buffer warna merupkan tempat gambar disimpan.

```
glClearColor(0.0,0.0,0.0);
glClear(GL_COLOR_BUFFER_BIT);
```

1.2.5. Spesifikasi warna

Pada Open Gl mendeskripsikan objek dengan warna objek adalah proses yang berjalan sendiri-sendiri. Karna pada umumnya seorang programmer akan mengatur warna terlebih dahulu lalu menggambar objek. Sebelum warna dibah makan semua objek yang digambar sesudah perintah tersebut akan menggunakan warna terakhir yang terdapat pada coloring scheme.

Untuk warna digunakan perintah glColor3f(), jika lebih dari tiga maka argument keempat adalah alpha yang akan dijelaskan pada bagian I-4 blending sebagai salah satu efek yang dipunyai Open Gl. Contoh berikut menujukan urutan langkah dalam proses spesifikasi warna sebelum objek digambar.

```
glColor3f(0.0,1.0,0.0); //setting warna
draw_object(A); //gambar object A
```

1.2.6. Kode Warna Open Gl

Pada prinsipnya Open Gl mempunyai prinsip pembuatan warna sama seperti RGB pada warna-warna yang disediakan oleh berbagai macam pemrograman. namun perbedaannya adalah :

- kode warna RGB pada pemrograman lain mempunyai nilai default (255,255,255).
- sedangkan Open Gl memiliki kode warna default (1,1,1) jadi intinya kode warna pada Open Gl mempunyai rentang antara 0 sampai dengan 1

Berikut ini adalah sedikit kode warna yang saya kumpulkan dari beberapa percobaan.

- Warna Merah = glColor3f (1.0,0.0,0.0)
- Warna Hijau = glColor3f (0.0,1.0,0.0)
- Warna Biru = glColor3f(0.0,0.0,1.0)
- Warna Kuning = glColor3f(1.0,1.0,0.0)
- Warna Cyan = glColor3f(0.0,1.0,1.0)
- Warna Magenta = glColor3f(1.0,0.0,1.0)
- Warna putih = glColor3f(1.0,1.0,1.0)
- Warna Hitam = glColor3f(0.0,0.0,0.0)

1.3.Hasil dan pembahasan

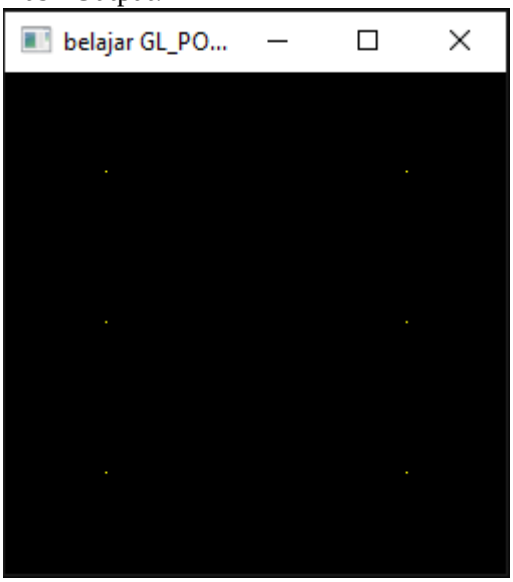
1.3.1. Membuat objek titik

Source code untuk objek titik

[*] main.cpp

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


void Draw() {
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1.0, 1.0, 1.0);        // digunakan untuk merubah warna yaitu warna putih
glBegin(GL_POINTS);              // awal kode untuk menggambar
glVertex3f(0.2, 0.2, 0.0);       // posisi titik yang akan digambar
glVertex3f(0.8, 0.2, 0.0);
glVertex3f(0.2, 0.5, 0.0);
glVertex3f(0.8, 0.5, 0.0);
glVertex3f(0.2, 0.8, 0.0);
glVertex3f(0.8, 0.8, 0.0);
glEnd();                              // akhir kode untuk menggambar
glFlush();                            // memastikan bahwa kode dieksekusi
}


void Initialize() {
glClearColor(0.0, 0.0, 0.0, 0.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
}


int main(int iArgc, char** cppArgv) {
glutInit(&iArgc, cppArgv);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize(250, 250);
glutInitWindowPosition(200, 200);
glutCreateWindow("belajar GL_POINTS");
Initialize();
glutDisplayFunc(Draw);
glutMainLoop();
return 0;
}
```

Hasil Output:



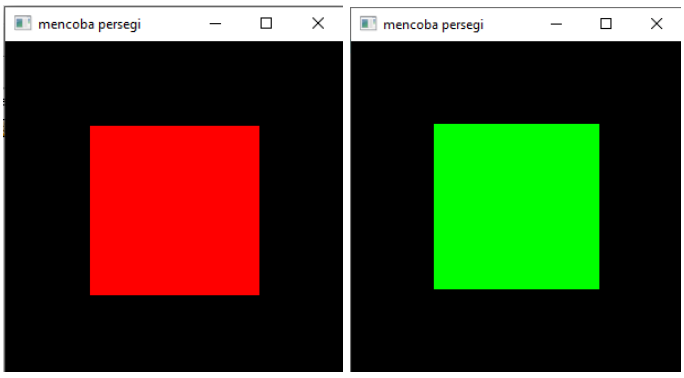
1.3.2. Membuat objek persegi

Source code untuk objek Persegi

[*] main.cpp

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

void display()
{
    glClear(GL_COLOR_BUFFER_BIT); // Membersihkan Layar
    glColor3f(1,0,0);// warna merah
    //glColor3f(0,1,0);// warna hijau
    //glColor3f(0,0,1);// warna biru
    glBegin(GL_QUADS);           // Mulai menggambar persegi
    glVertex2f(-0.5f, -0.5f);    // Sudut kiri bawah
    glVertex2f(0.5f, -0.5f);     // Sudut kanan bawah
    glVertex2f(0.5f, 0.5f);     // Sudut kanan atas
    glVertex2f(-0.5f, 0.5f);     // Sudut kiri atas
    glEnd();                    // Selesai menggambar persegi

    glFlush();                   // Memastikan semua perintah dieksekusi
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutCreateWindow("mencoba persegi");
    glutInitWindowSize(400, 400);
    glutInitWindowPosition(100, 100);
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}

```

Hasil Output:



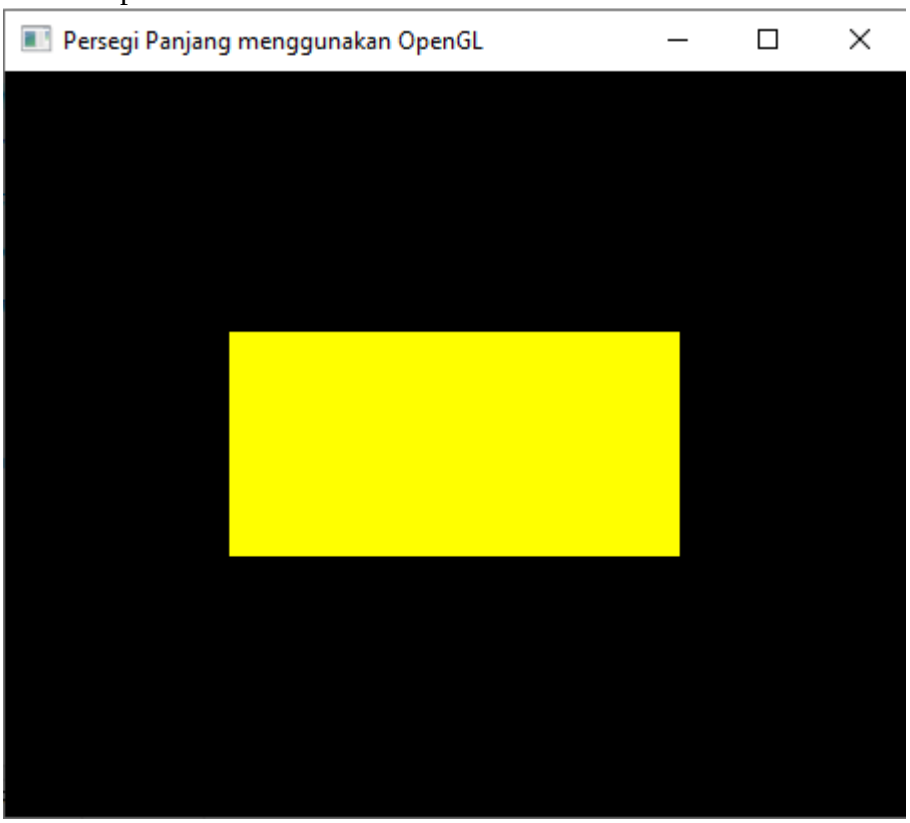
1.3.3. Persegi Panjang

Source code untuk objek Persegi Panjang

main.cpp

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

void display()
{
    glClear(GL_COLOR_BUFFER_BIT); // Membersihkan layar
    glColor3f(1,1,0);//warna kuning
    glBegin(GL_QUADS); // Mulai menggambar persegi panjang
    glVertex2f(-0.5f, -0.3f); // Sudut kiri bawah
    glVertex2f(0.5f, -0.3f); // Sudut kanan bawah
    glVertex2f(0.5f, 0.3f); // Sudut kanan atas
    glVertex2f(-0.5f, 0.3f); // Sudut kiri atas
    glEnd(); // Selesai menggambar persegi panjang

    glFlush(); // Memastikan semua perintah dieksekusi
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutCreateWindow("Persegi Panjang menggunakan OpenGL");
    glutInitWindowSize(400, 400);
    glutInitWindowPosition(100, 100);
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

Hasil output:



BAB II

PRAKTIKUM TRIANGLE STRIP

1.1. Tujuan praktikum

Praktikum ini Diharapkan dapat memberikan pengetahuan bagaimana cara mempelajari dan memahami penggunaan Triangle Strip sebagai salah satu metode penggambaran poligon dalam grafika komputer. Dapat memahami posisi dari setiap titik simpul (vertex) sehingga dapat membentuk suatu objek seperti segitiga, segiempat dan lain sebagainya.

1.2. Dasar teori

1.2.1.Triangle Strip

Pada dasarnya triangle strip digunakan untuk menggambar serangkaian segitiga terhubung. karena segitiganya terhubung kita tidak perlu berulang kali menentukan semua tiga titik simpul (vertex). Untuk membuat segitiga kita hanya perlu menentukan 7 simpul titik (vertex) saja untuk menggambarnya. • simpul v1,v2,v3 menarik simpul segitiga pertama.

• simpul v2,v3,v4 menarik simpul segitiga ke dua.

• simpul v3,v4,v5 menarik simpul segitiga ke tiga.

• simpul v4,v6,v5 menarik simpul segitiga ke empat dan seterusnya.

1.2.2. Quad Strip

Quad strip adalah sebuah penggambaran sebuah vertex (titik) membentuk segi empat. artinya quad strip ini setiap 4 vertex membentuk segi 4 dan semuanya saling berhubungan. Fungsi dari Quad strip adalah untuk menggambar bentuk-bentuk seperti gambar yang ada diatas.

1.2.3.GL Lines

Gl Lines Pada dasarnya GL Lines digunakan untuk menghubungkan antar titik menjadi sebuah garis. garis dibuat dengan menentukan dua end point atau II-2 posisi titik awal dan akhir dari suatu garis yang kemudian peralatan output membuat garis sesuai posisi titik-titik tersebut.

1.2.4.Line Strip

line strip pada dasarnya adalah untuk menggambar suatu garis yang menyambung dengan setiap titik ujung garis menyambung membentuk garis selanjutnya kemudian titik lainnya menyambung lagi garis di ujungnya. Ini biasanya digunakan untuk menggambar animasi game berupa rumput.

1.2.5.Line loop

Line Loop pada dasarnya digunakan untuk menggambar garis yang saling terhubung. Artinya line loop digunakan untuk membuat garis dan menghubungkannya dengan garis lainnya dan garis terakhir terhubung dengan garis pertama.

1.3. Hasil dan pembahasan

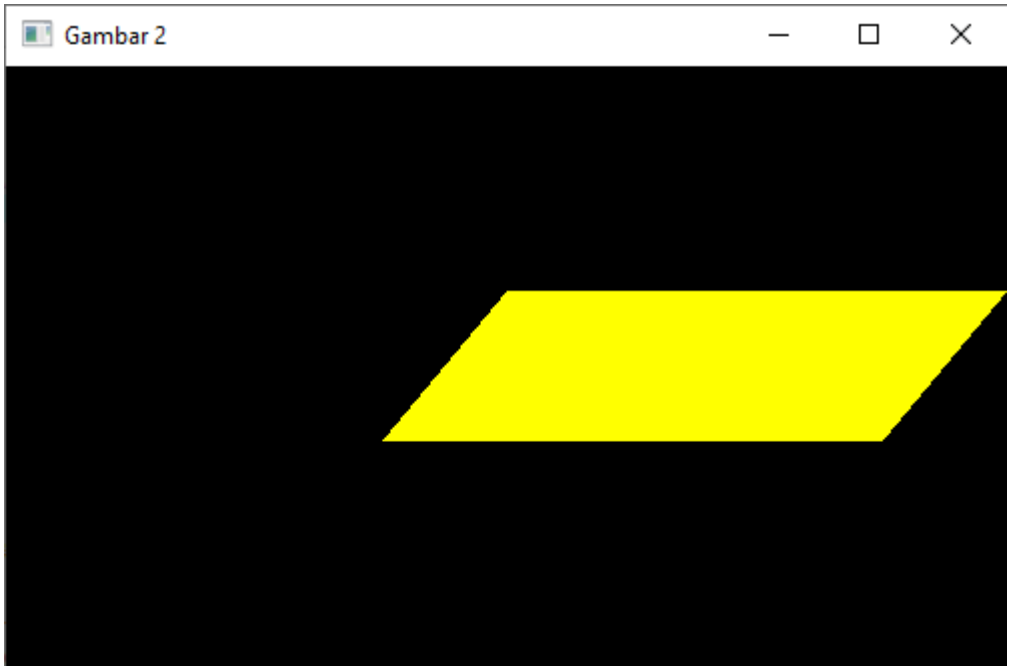
1.3.1.Triangle Strip

Source code

main.cpp

```
#include <GL/glut.h>
void Tampilan (void)
{
glClear(GL_COLOR_BUFFER_BIT);
//Gambar segitiga sama sisi berwarna kuning
glColor3f(1,1,0);
glBegin(GL_TRIANGLE_STRIP);
glVertex2f(-0.25, -0.25);
glVertex2f(0.0, 0.25);
glVertex2f(0.25, -0.25);
glVertex2f(0.5, 0.25);
glVertex2f(0.75, -0.25);
glVertex2f(1.0, 0.25);
glEnd();
glFlush();
glutSwapBuffers();
}
int main(int argc, char **argv)
{
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_DEPTH | GLUT_SINGLE | GLUT_RGBA);
glutInitWindowPosition(400,100);
glutInitWindowSize(500,300);
glutCreateWindow("Gambar 2");
glutDisplayFunc(Tampilan);
glutMainLoop();
}
```

Hasil Output:



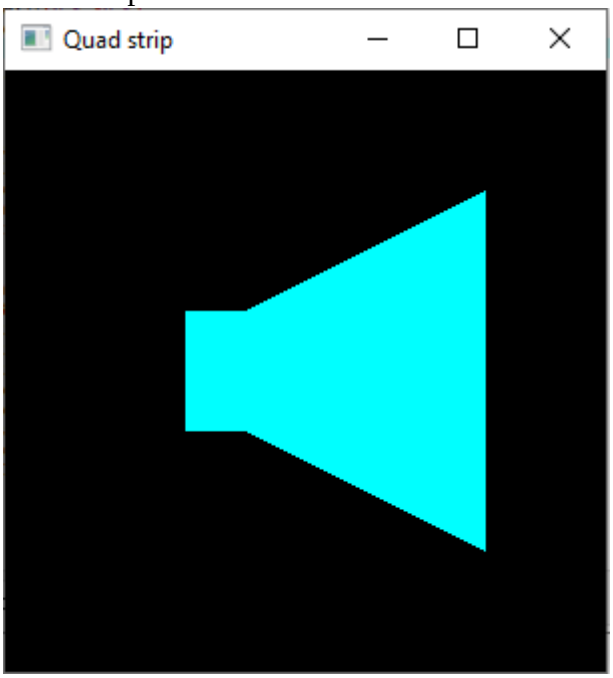
1.3.2.Quad strip

Source code

main.cpp

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

void Draw() {
 glClear(GL_COLOR_BUFFER_BIT);
 glColor3d(0,1,1);
 glBegin(GL_QUAD_STRIP);
    glVertex2f(-4.0,-2.0);
    glVertex2f(-4.0,2.0);
    glVertex2f(-2.0,-2.0);
    glVertex2f(-2.0,2.0);
    glVertex2f(6.0,-6.0);
    glVertex2f(6.0,6.0);
 glEnd();
 glFlush();
}


void Initialize() {
 glClearColor(0, 0, 0, 0.5);
 glMatrixMode(GL_PROJECTION);
 glLoadIdentity();
 glOrtho(-10.0, 10.0, -10.0, 10.0, -10.0, 10.0);
}

int main(int iArgc, char** cppArgv) {
 glutInit(&iArgc, cppArgv);
 glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
 glutInitWindowSize(300, 300);
 glutInitWindowPosition(250, 250);
 glutCreateWindow("Quad strip");
 Initialize();
 glutDisplayFunc(Draw);
 glutMainLoop();
 return EXIT_SUCCESS;
}

```

Hasil Output:



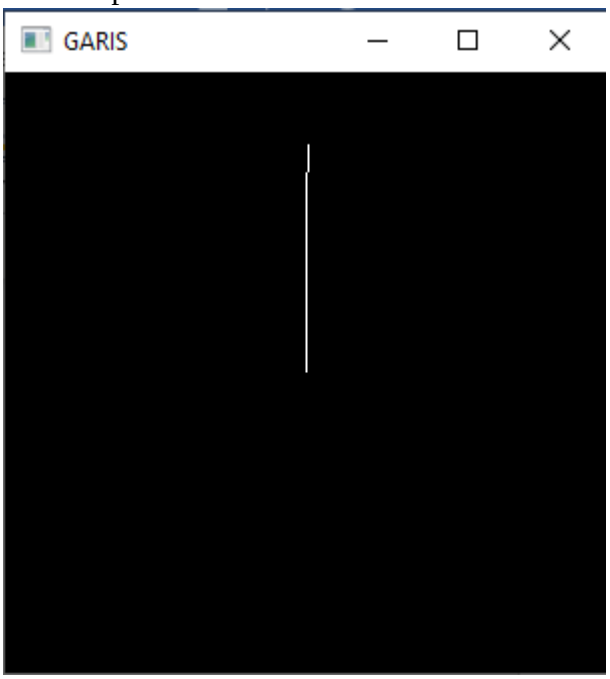
1.3.3.Gl Lines

Source code:

main.cpp

```
#include <GL/glut.h>
#include <math.h>
void display(void)
{
glClearColor (0.0f, 0.0f, 0.0f, 0.0f);
glClear (GL_COLOR_BUFFER_BIT);

glPushMatrix ();
glClearColor(1,1,1,0);
glColor3f(1,1,1); //
glBegin(GL_LINES);
glVertex3f(0,0,0.0);
glVertex3f(0.10,10.0,13.13);
glEnd ();
glFlush();
}

int main(int argc, char **argv)
{
glutInit(&argc, argv);
glutCreateWindow("GARIS");
glutDisplayFunc(display);
glutMainLoop();
return 0;
}
```

Hasil Otput:



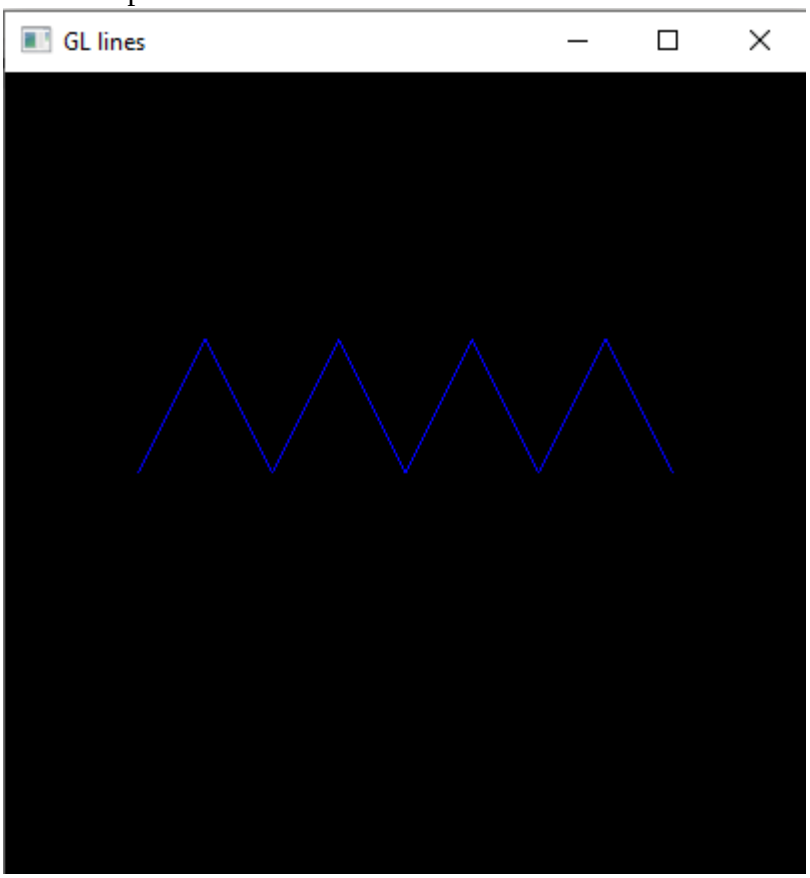
1.3.4. Line strip

Source code:

main.cpp

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

void display()
{
    glClear( GL_COLOR_BUFFER_BIT );

    glMatrixMode( GL_PROJECTION );
    glLoadIdentity();
    glOrtho( -6, 6, -6, 6, -1, 1);

    glMatrixMode( GL_MODELVIEW );
    glLoadIdentity();

    glColor3f( 0, 0, 1 );
    glBegin(GL_LINE_STRIP);
    glVertex2f(-4.0f, 0.0f);
    glVertex2f(-3.0f, 2.0f);
    glVertex2f(-2.0f, 0.0f);
    glVertex2f(-1.0f, 2.0f);
    glVertex2f(0.0, 0.0f);
    glVertex2f(1.0f, 2.0f);
    glVertex2f(2.0f, 0.0f);
    glVertex2f(3.0f, 2.0f);
    glVertex2f(4.0f, 0.0f);
    glEnd();

    glutSwapBuffers();
}

int main( int argc, char **argv )
{
    glutInit( &argc, argv );
    glutInitDisplayMode( GLUT_RGBA | GLUT_DOUBLE );
    glutInitWindowSize( 400, 400 );
    glutCreateWindow( "GL lines" );
    glutDisplayFunc( display );
    glutMainLoop();
    return 0;
}
```

Hasil Output:



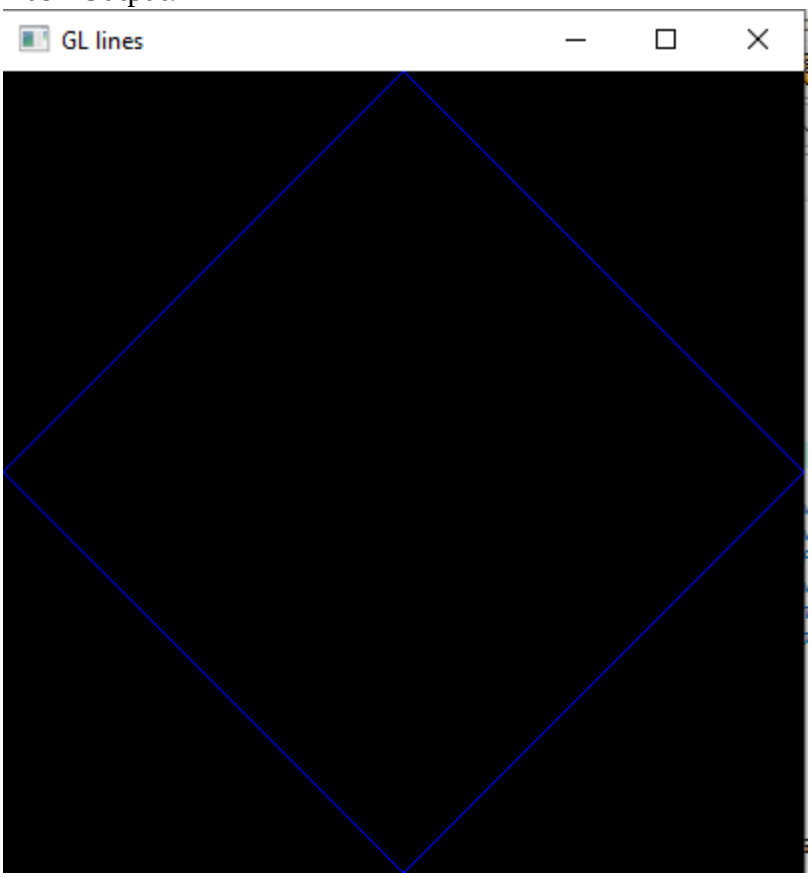
1.3.5.Line Loop

Source Code:

main.cpp

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

void display()
{
    glClear( GL_COLOR_BUFFER_BIT );
    glMatrixMode( GL_PROJECTION );
    glLoadIdentity();
    glMatrixMode( GL_MODELVIEW );
    glLoadIdentity();

    glColor3f( 0, 0, 1 );
    glBegin(GL_LINE_LOOP);//start drawing a line loop
      glVertex2f(-1.0,0.0);//Left of window
      glVertex2f(0.0,-1.0);//bottom of window
      glVertex2f(1.0,0.0);//right of window
      glVertex2f(0.0,1.0);//top of window
    glEnd();//end drawing of line loop

    glutSwapBuffers();
}

int main( int argc, char **argv )
{
    glutInit( &argc, argv );
    glutInitDisplayMode( GLUT_RGBA | GLUT_DOUBLE );
    glutInitWindowSize( 400, 400 );
    glutCreateWindow( "GL lines" );
    glutDisplayFunc( display );
    glutMainLoop();
    return 0;
}
```

Hasil Output:



BAB III

OBJEK 3D

3.1.Tujuan Praktukim

Tujuan praktikum objek 3D dapat bervariasi tergantung pada konteks dan tujuan spesifik dari praktikum tersebut. Beberapa tujuan umum yang dapat dicapai melalui praktikum objek 3D adalah:

1. Pemahaman Konsep 3D: Praktikum objek 3D dapat membantu siswa atau peserta praktikum untuk memahami konsep dasar dalam representasi objek dalam ruang 3D. Ini melibatkan pemahaman tentang koordinat 3D, transformasi geometri, proyeksi perspektif, dan konsep lain yang terkait dengan objek 3D.
2. Pemodelan dan Desain: Praktikum objek 3D sering melibatkan pemodelan dan desain objek 3D menggunakan perangkat lunak atau alat khusus. Tujuannya adalah mengajarkan siswa atau peserta praktikum untuk menggunakan teknik pemodelan 3D dan mengembangkan keterampilan dalam membuat objek 3D yang kompleks dan realistis.
3. Pengembangan Permainan atau Simulasi: Objek 3D sering digunakan dalam pengembangan permainan komputer dan simulasi. Praktikum objek 3D dapat membantu siswa atau peserta praktikum memahami konsep dan teknik pengembangan permainan 3D, termasuk pemodelan objek, animasi, pencahayaan, dan efek visual.
4. Pemrograman Grafis: Praktikum objek 3D juga dapat digunakan untuk mengenalkan konsep dan teknik pemrograman grafis. Peserta praktikum dapat belajar mengimplementasikan transformasi, menerapkan pencahayaan, membuat animasi, dan mengoptimalkan rendering grafis untuk menciptakan objek 3D yang interaktif dan real-time.
5. Kolaborasi Tim: Praktikum objek 3D sering melibatkan kerja tim dalam membuat objek 3D yang kompleks. Tujuan praktikum ini adalah mengajarkan siswa atau peserta praktikum untuk bekerja dalam tim, berkolaborasi dalam proses desain dan pemodelan, serta mengatur tugas dan tanggung jawab masing-masing anggota tim.
6. Presentasi dan Komunikasi: Praktikum objek 3D sering mengharuskan siswa atau peserta praktikum untuk membuat presentasi visual atau demonstrasi objek 3D yang mereka buat. Tujuannya adalah mengembangkan keterampilan presentasi dan komunikasi efektif dalam menggambarkan dan menjelaskan objek 3D kepada orang lain.

3.2.Sifat sifat 3D

Berikut adalah beberapa sifat umum yang terkait dengan objek 3D:

1. Dimensi Ruang: Objek 3D ada dalam tiga dimensi ruang, yaitu panjang, lebar, dan tinggi. Ini berarti objek memiliki dimensi dalam ketiga arah ini, memberikan tampilan yang lebih realistis dan kompleks dibandingkan dengan objek 2D.
2. Bentuk: Objek 3D dapat memiliki berbagai bentuk, baik yang sederhana seperti kubus, bola, atau silinder, maupun bentuk yang lebih kompleks dan organik seperti manusia, hewan, atau bangunan. Bentuk objek 3D tergantung pada desain dan pemodelan yang dilakukan.
3. Kedalaman dan Jarak: Objek 3D memiliki kedalaman yang dapat dilihat oleh mata manusia. Hal ini memungkinkan pengamatan jarak antara objek dalam tampilan 3D, memberikan ilusi tentang jarak dan perspektif.

4. Rotasi dan Transformasi: Objek 3D dapat dirotasi dan ditransformasikan dalam ruang 3D. Ini berarti objek dapat bergerak, berputar, diperbesar, atau dipindahkan sesuai dengan kebutuhan dan interaksi pengguna.
5. Pencahayaan dan Bayangan: Objek 3D dapat dipengaruhi oleh pencahayaan dan menghasilkan bayangan. Pencahayaan yang tepat memberikan ilusi kedalaman, tekstur, dan realisme pada objek 3D. Bayangan juga dapat memberikan efek visual dan memberikan informasi tentang posisi dan bentuk objek.
6. Texturing dan Material: Objek 3D dapat memiliki tekstur dan material yang berbeda, seperti kayu, logam, kaca, atau kulit. Penggunaan tekstur dan material yang tepat dapat memberikan tampilan yang lebih realistis pada objek 3D dan meningkatkan kualitas visualnya.
7. Animasi: Objek 3D dapat dihidupkan melalui animasi, yaitu perubahan posisi, bentuk, atau sifat objek seiring waktu. Animasi memberikan interaksi dinamis dengan objek dan meningkatkan keaslian serta pengalaman pengguna.
8. Interaksi Pengguna: Objek 3D dapat diinteraksikan oleh pengguna melalui perangkat input seperti mouse, keyboard, atau perangkat sentuh. Pengguna dapat memilih, memindahkan, memanipulasi, atau berinteraksi dengan objek 3D secara langsung.

Sifat-sifat di atas mempengaruhi pengalaman dan pemahaman kita terhadap objek 3D. Dalam pemodelan, animasi, dan pengembangan aplikasi 3D, sifat-sifat ini menjadi penting dalam menciptakan dunia virtual yang menarik dan realistis.

3.3.Konsep 3D

- Untuk mendapatkan tampilan 3D yang dimodelkan dalam koordinat dunia, pertama harus menentukan koordinat referensi untuk “kamera”.
- Koordinat referensi ini mendefinisikan posisi dan orientasi utk bidang datar kamera, yang digunakan untuk menampilkan objek.
- Deskripsi objek dikirim ke koordinat referensi kamera dan diproyeksikan ke display plane (bidang datar untuk tampilan).
- Titik-titik di dunia nyata dipetakan ke dalam ruang 2 dimensi.
- Cahaya menyebabkan suatu objek dapat terlihat.
- Warna objek ditentukan dari properti objek tersebut.

3.4.Repentasi Objek 3D

- Batasan objek dapat dibentuk dari berbagai kombinasi bidang datar dan kurva.
- Representasi objek 3D :

- Polygon surfaces / polyhedral
- Kurva
- Constructive Solid Geometry (CSG)
- Sweep representation

3.5.Polygon surfaces

- Gabungan polygon tertutup membentuk objek baru
- Paling umum digunakan
- Mendeskripsikan sebuah objek sebagai kumpulan polygon
- Polygon mudah untuk diproses sehingga proses akan menjadi lebih cepat
- Umumnya, basic polygon berbentuk segitiga

3.6.Kurva spline

- Memudahkan untuk menggambar bentuk kurva yang kompleks

- Caranya dengan memasukkan rangkaian titik, dan kurva akan terbentuk mengikuti rangkaian titik
- Titik-titik tersebut disebut titik kendali (control points)
- Kurva yang melewati tiap titik kendali disebut interpolasi kurva spline (interpolating curve)
- Kurva yang melewati di dekat titik kendali namun tidak melewati titik kendali disebut pendekatan kurva spline (approximating curve)
- Untuk mengubah bentuk kurva, caranya dengan memindahkan posisi titik kendali

3.7.Proyeksi

- Koordinat dunia -> koordinat kamera -> proyeksi objek 3 dimensi ke bidang pandang 2 dimensi
- metode : proyeksi paralel dan proyeksi proyeksi perspektif
- Objek yang diproyeksikan ditentukan dengan menghitung perpotongan garis proyeksi dengan bidang pandang

3.8.Hasil dan pembahasan

3.8.1. Wire Cone

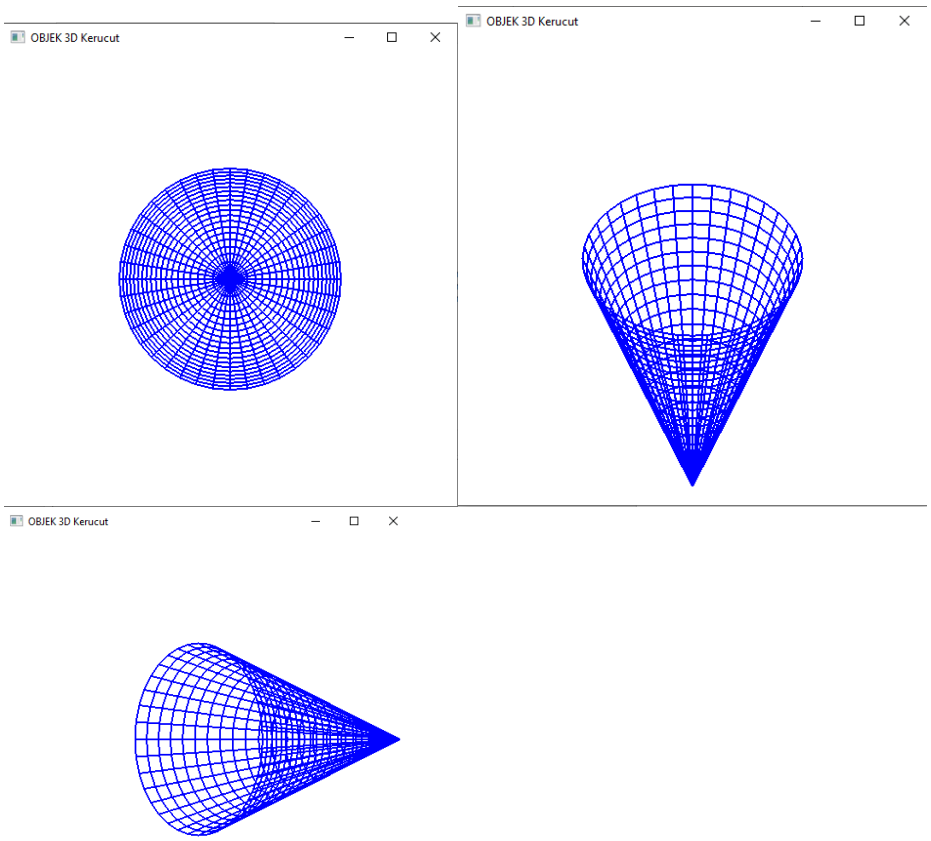
Source code:

3D Example_Use Fungtion WireCone.cpp

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

int w = 480;
int h = 480;
int z = -150;
int sudut = 45;
int sumbux, sumbuy, sumbuz;

void wirecone (void){
    glColor3f(0,0,1);
    glLineWidth(2);
    glutWireCone(40., 80, 40,20);
    }
void renderScene(){
    glClearColor(1,1,1,1);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    glTranslatef(0,0,z);
    glRotatef(sudut,sumbux,sumbuy,sumbuz);
    wirecone();
    glutSwapBuffers();
    }
void resize(int w1, int h1){
    glViewport(0,0,w1,h1);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, (float) w1/(float) h1, 1.0, 1300.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    }
```

```
void init(){
    glClearColor(0,0,0,1);
    glEnable(GL_DEPTH_TEST);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, (float) w/(float) h, 1.0, 1300.0);
    glMatrixMode(GL_MODELVIEW);
    }
void myKeyboard(unsigned char key, int x, int y){
    if (key == 'a') z+=2;
    else if (key == 's') z-=2;
    else if (key == 'x') {sudut +=2; sumbux = 1; sumbuy = 0; sumbuz = 0;}
    else if (key == 'y') {sudut +=2; sumbux = 0; sumbuy = 1; sumbuz = 0;}
    else if (key == 'z') {sudut +=2; sumbux = 0; sumbuy = 0; sumbuz = 1;}
    }
void update (int value){
    glutPostRedisplay();
    glutTimerFunc(50, update, 0);
    }
main (int argc, char **argv){
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_DEPTH | GLUT_RGBA);
    glutInitWindowPosition(100,100);
    glutInitWindowSize(480,480);
    glutCreateWindow("OBJEK 3D Kerucut");
    gluOrtho2D(-320.,320.,-320.,320.);
    glutTimerFunc(50, update, 0);
    glutDisplayFunc(renderScene);
    glutKeyboardFunc(myKeyboard);
    glutReshapeFunc(resize);

    init();
    glutMainLoop();
    }
```

Hasil Output:



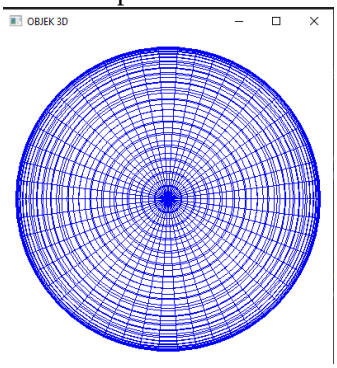
3.8.2. wire sphare

Source code:

3D Example_Use Fungtion WireSphere.cpp

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

int w=400, h=400, z=-50;
int x1=0, y1=0, sudut=0, z1=0;
void renderScene(void){
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glClearColor(1,1,1,1);
    glLoadIdentity();
    glTranslatef(0,0,z);
    glRotatef(sudut,x1,y1,z1);
    glColor3f(0,0,1);
    // glutWireCube(4);
    glutWireSphere(50,50,50);
    glutSwapBuffers();
}
void resize(int w1, int h1){
    glViewport(0,0,w1,h1);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0,(float) w1/(float) h1, 1.0,300.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}
void myKeyboard(unsigned char key, int x, int y){
if (key == 'x') {
    x1=1;
    y1=0;
    z1=0;
     sudut+=5;
     }
else if (key == 'y') {
     y1=1;
     x1=0;
     z1=0;
     sudut+=-5;
     }
else if (key == 'z') {
     y1=0;
     x1=0;
     z1=z;
     sudut+=5;
     }
}
void mySpecialKeyboard(int key, int x, int y){
     switch(key){
     case GLUT_KEY_UP : z-=5;
         break;
     case GLUT_KEY_DOWN : z+=5;
         break;
     }
}
void init(){
     glClearColor(0,0,0,1);
     glEnable(GL_DEPTH_TEST);
     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
     gluPerspective(45.0,(GLdouble) w/(GLdouble) h, 1.0,300.0);
     glMatrixMode(GL_MODELVIEW);
}
void timer(int value){
     glutPostRedisplay();
     glutTimerFunc(50,timer,0);
}
int main (int argc, char **argv){
     glutInit(&argc, argv);
     glutInitDisplayMode(GLUT_DOUBLE | GLUT_DEPTH | GLUT_RGBA);
     glutInitWindowPosition(100,100);
     glutInitWindowSize(w,h);
     glutCreateWindow("OBJEK 3D ");
     gluOrtho2D(-w/2,w/2,-h/2,h/2);
     glutDisplayFunc(renderScene);
     glutReshapeFunc(resize);
     glutKeyboardFunc(myKeyboard);
     glutSpecialFunc(mySpecialKeyboard);
     glutTimerFunc(1,timer,0);
     init();
     glutMainLoop();
}
```

Hasil Output:



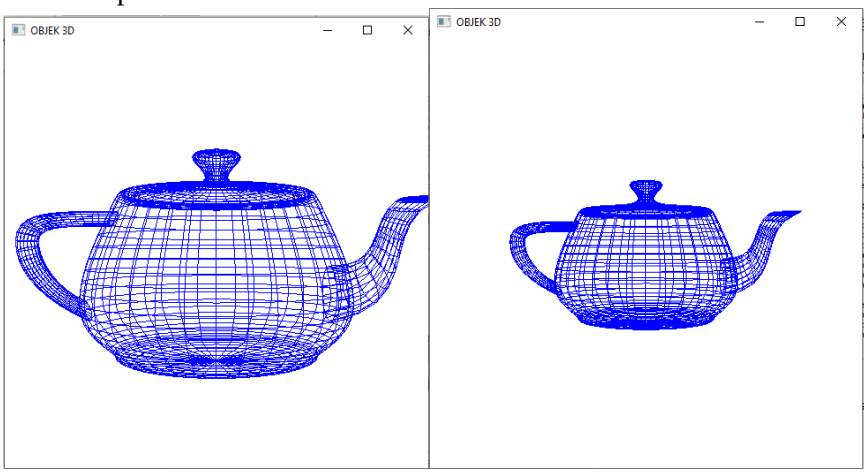
3.8.3. Wire Teapot
Source code:

3D Example_Use Fungtion WireTeapot.cpp

```
#include<stdlib.h>
#include<GL/glut.h>
#include<stdio.h>
float w=480, h=480;
float n=-10;
int sumbu=30;
int xx=0, yy=0, zz=0, s=2;

void display(){
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    glTranslatef(0,0,n);
    glRotatef(sumbu, xx,yy,zz);
    glColor3f(0,0,1);
    glutWireTeapot(s);
    glutSwapBuffers();
    }
void resize(int w1, int h1){
    glViewport(0,0,w1,h1);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, (float) w1/(float) h1, 1.0,300.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    }
void init(){
    glClearColor(1.0,1.0,1.0,1.0);
    glEnable(GL_DEPTH_TEST);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.,(GLdouble)w/(GLdouble)h,1.,300.);
    glMatrixMode(GL_MODELVIEW);
    }
void myKeyboard(unsigned char key, int x, int y){
    if (key=='x'){
        sumbu+=10;
        xx=1;
        yy=0;
        zz=0;
        }
    else if(key=='y'){
        sumbu+=10;
        xx=0;
        yy=1;
        zz=0;
        }
    else if(key=='z'){
        sumbu+=10;
        xx=0;
        yy=0;
        zz=1;
        }
    }
void Spesial(int key, int x, int y){
    switch(key){
        case GLUT_KEY_UP:
```

```
57          s++;
58          break;
59      case GLUT_KEY_DOWN:
60          s--;
61          break;
62          }
63      }
64  void timer(int value){
65      glutPostRedisplay();
66      glutTimerFunc(500,timer,0);
67      }
68  main(int argc, char **argv){
69      glutInit(&argc, argv);
70      glutInitDisplayMode(GLUT_DOUBLE | GLUT_DEPTH | GLUT_RGBA);
71      glutInitWindowPosition(100,100);
72      glutInitWindowSize(w,h);
73      glutCreateWindow("OBJEK 3D ");
74      gluOrtho2D(-w/2,w/2,-h/2,h/2);
75      glutDisplayFunc(display);
76      glutReshapeFunc(resize);
77      init();
78      glutTimerFunc(50,timer,0);
79      glutKeyboardFunc(myKeyboard);
80      glutSpecialFunc(Spesial);
81      glutMainLoop();
82      }
```

Hasil Output:



3.8.4. Wire Torus

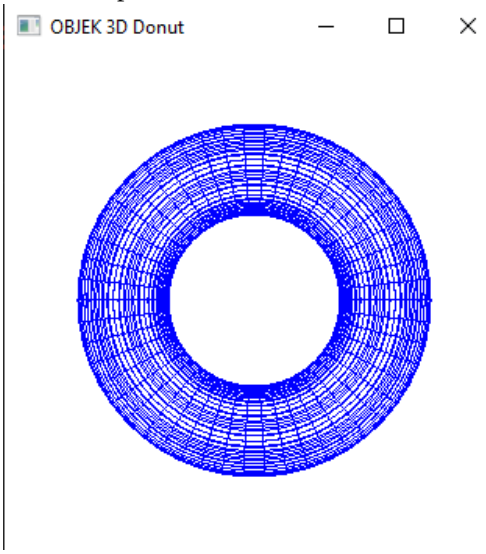
Source code:

3D Example_Use Fungtion WireTorus.cpp

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

float x=0, y=0, z=-10;
int w = 320, h = 320, i, j, k, l;
int dlm = 2, lr=4, sisi=50, cincin=50;

void objek(void){
    glColor3f(0,0,1);
    glutWireTorus(1, 3, 50, 50);
    }
void update(int value){
    glutPostRedisplay();
    glutTimerFunc(100, update, 0);
    }
void myDisplay(void){
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glClearColor(1,1,1,1);
    glLoadIdentity();
    glTranslatef(x,y,z);
    glRotatef(i,j,k,0);
    objek();
    glutSwapBuffers();
    }
void reshape (int w, int h){
    glViewport (0, 0, (GLsizei)w, (GLsizei)h);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity();
    gluPerspective (60, (GLfloat)w / (GLfloat)h, 1.0, 100.0);
    glMatrixMode (GL_MODELVIEW);
    }
void init(){
    glShadeModel (GL_SMOOTH);
    glClearColor (0.0f, 0.0f, 0.0f, 0.5f);
    glClearDepth (1.0f);
    glEnable (GL_DEPTH_TEST);
    glDepthFunc (GL_LEQUAL);
    glHint (GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST);
    return;
    }
void myKeyboard(unsigned char key, int x, int y){
    if(key == 'a'){
        i+=10; j=1; k=0;
        }
    else if(key == 'b'){
        i+=10; j=0; k=1;
        }
    else if(key == 'c'){
        i+=10; j=1; k=1;
        }
    }
void mySpecialKeyboard(int key, int x, int y){
    switch(key){
        case GLUT_KEY_UP:
            dlm+=1;
            break;
```

```
57          case GLUT_KEY_DOWN:
58              dlm-=1;
59              break;
60          case GLUT_KEY_LEFT:
61              lr-=1;
62              break;
63          case GLUT_KEY_RIGHT:
64              lr-=-1;
65              break;
66          }
67      }
68  main(int argc, char **argv) {
69      glutInit(&argc, argv);
70      glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH);
71      glutInitWindowPosition(100,100);
72      glutInitWindowSize(w,h);
73      glutCreateWindow("OBJEK 3D Donut ");
74      glutTimerFunc(50, update, 0);
75      glutKeyboardFunc(myKeyboard);
76      init();
77      glutDisplayFunc(myDisplay);
78      glutReshapeFunc (reshape);
79      gluOrtho2D(-w/2,w/2,-h/2,h/2);
80      glutMainLoop();
81  }
```

Hasil Output:



3.8.4. Wire Cube

Source code:

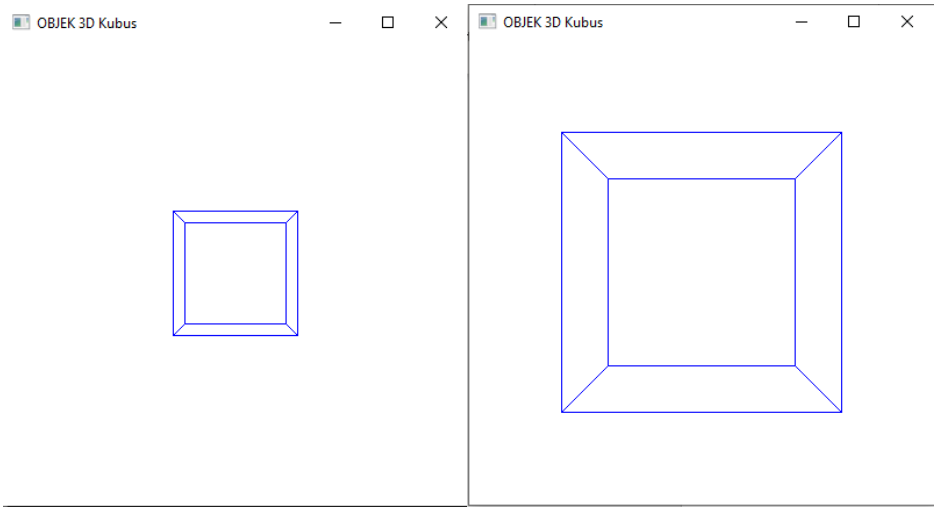
3D Example_Use Fungtion WireCube.cpp

D:\M.K\Grafika Komputer\Praktikum\3D Task Example\3D Example_Use Fungtion WireCube.cpp

```
#include <stdlib.h>

int w=400, h=400, z=-50;
int x1=0, y1=0, sudut=0, z1=0;

void renderScene(void){
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glClearColor(1,1,1,1);
    glLoadIdentity();
    glTranslatef(0,0,z);
    glRotatef(sudut,x1,y1,z1);
    glColor3f(0,0,1);
    glutWireCube(4);
    glutSwapBuffers();
}
void resize(int w1, int h1){
    glViewport(0,0,w1,h1);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0,(float) w1/(float) h1, 1.0,300.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}
void myKeyboard(unsigned char key, int x, int y){
if (key == 'x') {
    x1=1;
    y1=0;
    z1=0:
    sudut+=5;
}
else if (key == 'y') {
    y1=1;
    x1=0;
    z1=0;
    sudut+=-5;
}
else if (key == 'z') {
    y1=0;
    x1=0;
    z1=z;
    sudut+=5;
}
}
void mySpecialKeyboard(int key, int x, int y){
    switch(key){
    case GLUT_KEY_UP : z-=5;
        break;
    case GLUT_KEY_DOWN : z+=5;
    break;
    }
}
void init(){
    glClearColor(0,0,0,1);
    glEnable(GL_DEPTH_TEST);
    glMatrixMode(GL_PROJECTION);
```

```
57      glLoadIdentity();
58      gluPerspective(45.0,(GLdouble) w/(GLdouble) h, 1.0,300.0);
59      glMatrixMode(GL_MODELVIEW);
60  }
61  void timer(int value){
62      glutPostRedisplay();
63      glutTimerFunc(50,timer,0);
64  }
65  int main (int argc, char **argv){
66      glutInit(&argc, argv);
67      glutInitDisplayMode(GLUT_DOUBLE | GLUT_DEPTH | GLUT_RGBA);
68      glutInitWindowPosition(100,100);
69      glutInitWindowSize(w,h);
70      glutCreateWindow("OBJEK 3D Kubus ");
71      gluOrtho2D(-w/2,w/2,-h/2,h/2);
72      glutDisplayFunc(renderScene);
73      glutReshapeFunc(resize);
74      glutKeyboardFunc(myKeyboard);
75      glutSpecialFunc(mySpecialKeyboard);
76      glutTimerFunc(1,timer,0);
77      init();
78      glutMainLoop();
79  }
80
```

Hasil Output:

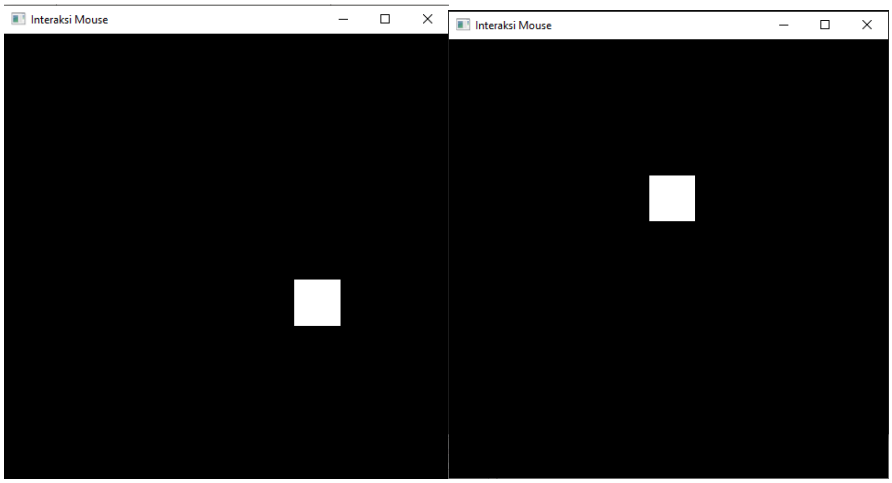


3.8.5. Interaksi mouse

Mouse Function i.cpp

```
#include <stdlib.h>
#include <gl/glut.h>

int w = 480, h = 480;
void drawQuad (){
    glBegin(GL_QUADS);
        glVertex2i(-25.,0.);
        glVertex2i(-25.,50.);
        glVertex2i(25.,50.);
        glVertex2i(25.,0.);
    glEnd();
}
float x=0,y=0,z=0;
void myDisplay(void) {
    glClear(GL_COLOR_BUFFER_BIT);
    glPushMatrix();
    glTranslatef(x,y,z);
    drawQuad();
    glPopMatrix();
    glFlush();
}
void mouse(int button, int state, int xmouse, int ymouse) {
    if(button==GLUT_LEFT_BUTTON && state==GLUT_DOWN){
        x=xmouse-(w/2);
        y=(h/2)-ymouse;
}
}
void motion(int x,int y){
}
void timer (int value) {
    glutPostRedisplay();
    glutTimerFunc(150,timer,0);
}
main (int argc, char **argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DEPTH | GLUT_SINGLE | GLUT_RGBA);
    glutInitWindowPosition(100,100);
    glutInitWindowSize(w,h);
    glutCreateWindow("Interaksi Mouse");
    gluOrtho2D(-w/2,w/2,-h/2,h/2);
    glutDisplayFunc(myDisplay);
    glutMouseFunc(mouse);
    glutMotionFunc(motion);
    glutTimerFunc(150,timer,0);
    glutMainLoop();
}

```

Hasil Output:



BAB IV

KESIMPULAN DAN SARAN

4.1. Kesimpulan

Dalam laporan praktikum OpenGl C++ ini, saya telah mempelajari dasar-dasar dari penggunaan library OpenGl untuk membuat grafik 2D dan 3D. Saya juga telah berhasil membuat beberapa objek geometri seperti persegi, segitiga, dan bola, serta menerapkan transformasi dasar seperti rotasi dan penskalaan. Saya menyimpulkan bahwa penggunaan OpenGl C++ sangatlah berguna dalam pembuatan grafik komputer yang realistis, intraktif dan inovatif. Selain itu, pemahaman tentang konsep dasar seperti koordinat dan vertex sangat bermanfaat untuk mencapai hasil yang diinginkan. Melalui praktikum ini, saya juga telah berhasil meningkatkan pemahaman dan kreativitas saya tentang OpenGl C++.

4.2. Saran

Saran saya adalah tetap menjaga ilmu yang tela saya pahami, dan tidak berhenti dalam mencoba berbagai hal yang berkaitan dengan open gl. Sehingga dapat mengetahui bagian yang mungkin tidak sempat di praktikkan dan bisa menjadi bekal di kemudian hari