MEMBUAT MODEL 3D WASHING MACHINE

”Diajukan untuk Memenuhi Tugas Akhir Praktikum Grafika Komputer”

KELOMPOK :

AHMAD KAMAL B (065118121)

DZIKRI FAIZZIYAN (065118123)

M. RAFELINO PUTRA (065118124)

PROGRAM STUDI ILMU KOMPUTER

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PAKUAN

BOGOR

2020

KATA PENGANTAR

Puji dan syukur kami panjatkan kehadirat Tuhan Yang Maha Esa, karena dengan kehendak-Nya penulis dapat menyelesaikan laporan tugas akhir Grafika Komputer ini dengan judul "Membuat Model 3D Washing Machine" sebagai salah satu syarat untuk memenuhi tugas akhir praktikum Grafika Komputer di Universitas Pakuan.

Penulis menyadari bahwa dalam penulisan laporan ini masih banyak terdapat kekurangan yang harus disempurnakan, sehingga kritik dan saran sangat dibutuhkan demi perbaikan di masa yang akan datang.

Bogor, 05 Mei 2020

Penulis

BAB I
PENDAHULUAN

1.1 Latar Belakang Masalah

Komputer grafika adalah bagian dari ilmu komputer yang berkaitan dengan pembuatan dan manipulasi gambar (visual) secara digital. Bentuk sederhana dari komputer grafika adalah komputer grafika 2D yang kemudian berkembang menjadi komputer 3D, pemrosesan citra (image processing), dan pengenalan pola (pattern recognition). Grafika komputer juga sering dikenal juga dengan istilah visualisasi data.

Seiring berkembangnya grafika komputer, banyak teknologi baru untuk membuat objek 3D, diantaranya adalah OpenGL. OpenGL adalah suatu graphic library yang sebagian bersifat open source, dipakai pada banyak platform (windows, linux) dan dapat digunakan pada berbagai jenis compiler seperti C++ atau Delphi. OpenGL bukanlah bahasa pemrograman tetapi merupakan suatu Application Programming Interface (API).

Berdasarkan penjelasan di atas, maka kami bermaksud untuk membuat model 3D untuk menerapkan komputer grafika dengan teknologi OpenGL, sehinga dapat menambah pemahaman kami tentang komputer grafika dan OpenGL. Objek 3D yang akan kami buat adalah Washing Machine (Mesin Cuci).

1.2 Perumusan Masalah

Berdasarkan pemaparan pada latar belakang masalah, maka dapat disimpulkan beberapa masalah :

1. Bagaimana memahami dan menerapkan konsep komputer grafika dalam aplikasi 3D.
2. Bagaimana memahami dan menerapkan fungsi-fungsi OpenGL dalam aplikasi 3D.

1.3 Maksud dan Tujuan

Untuk mengimplementasikan konsep komputer grafika dan fungsi OpenGL, maka kami bermaksud membuat model 3D Washing Machine.

Tujuan dari aplikasi ini adalah :

1. Memahami konsep komputer grafika dan fungsi OpenGL dalam model 3D.
2. Menerapkan konsep komputer grafika dan fungsi OpenGL sehingga lebih mengerti.

1.4 Batasan Masalah

Adapun batasan masalahnya sebagai berikut:

1. Bahasa yang digunakan bahasa pemrograman C++
2. IDE yang digunakan Dev C++ dengan library openGL
3. Informasi yang disampaikan tentang Model 3D Washing Machine

BAB II
LANDASAN TEORI

2.1 Visual C++

Visual C++ adalah sebuah produk IDE untuk bahasa pemrograman C dan C++ yang dikembangkan Microsoft. Bahasa C atau C++ adalah suatu bahasa pemrograman. Bahasa C termasuk sebagai bahasa pemrograman tingkat menengah, maksudnya bahasa C bisa dipelajari dengan lebih mudah karena mudah dimengerti tetapi mempunyai kemampuan yang tinggi.

Bahasa C++ bisa digunakan untuk merekayasa program untuk segala kebutuhan, baik untuk aplikasi bisnis, matematis atau bahkan game. Semua bahasa mempunyai kelemahan atau kelebihan sendiri-sendiri. Begitu juga dengan bahasa C. Adapun sebagian kelebihan dari bahasa C++ adalah sebagai berikut :

- Banyak memiliki operator untuk mengolah / memanipulasi data.
- Bahasa C++ termasuk sebagai bahasa yang terstruktur sehingga program dapat lebih mudah dipahami atau dikembangkan.
- Bahasa C++ lebih mudah dimengerti karena lebih mirip kepada bahasa manusia.
- Kecepatan eksekusi tinggi.
- Mengenal data pointer.

❖ Sedangkan kelemahan dari bahasa C++ adalah :

- Banyaknya operator atau cara penulisan program kadang menimbulkan kebingungan para pemakainya.
- Perlunya ketelitian dalam penulisan program karena perintah (*statement*)
- dalam bahasa C++ bersifat *case sensitiv*

2.2 OpenGL

OpenGL adalah suatu graphic library yang sebagian bersifat open source, dipakai pada banyak platform (windows, linux) dan dapat digunakan pada berbagai jenis compiler seperti C++ atau Delphi. OpenGL bukanlah bahasa pemrograman tetapi merupakan suatu Application Programming Interface (API).

Tabel 2.1 Contoh Perintah-Perintah dalam OpenGL

Perintah	Arti	Keterangan
glVertex2i(x,y);	Lokasi titik berada di (x,y)	Tipe argumennya adalah integer dan 2 dimensi yaitu x dan y
glVertex2f(x,y);	Lokasi titik berada di (x,y)	Tipe argumennya adalah float dan 2 dimensi yaitu x dan y
glVertex3i(x,y,z);	Lokasi titik berada di (x,y,z)	Tipe argumennya adalah integer dan 2 dimensi yaitu x, y dan z
glVertex3f(x,y,z);	Lokasi titik berada di (x,y,z)	Tipe argumennya adalah float dan 2 dimensi yaitu x, y dan z
glClearColour(R, G, B, α);	Warna latar belakang	Empat komponen warna yaitu Red, Green, Blue dan alpha
glColor3f(R, G, B);	Warna latar muka (pena)	Tiga komponen warna yaitu Red, Green dan Blue
glColor4f(R, G, B);	Warna latar muka (pena)	Empat komponen warna yaitu Red, Green, Blue dan alpha
glPointSize(k);	Ukuran titik k piksel	Besar kecilnya ukuran titik tergantung pada k (integer)
glBegin(GL_POINTS);	Titik	Objek primitive (lihat gambar 2.2)
glBegin(GL_LINES);	Garis	Objek primitive (lihat gambar 2.2)
glBegin(GL_LINE_STRIP);	Poligaris	Objek primitive (lihat gambar 2.2)
glBegin(GL_LINE_LOOP);	Poligaris tertutup (polygon)	Objek primitive (lihat gambar 2.2)

glBegin(GL_TRIANGLES);	Segitiga	Objek primitive (lihat gambar 2.2)
glBegin(GL_TRIANGLE_STRIP);	Segitiga	Objek primitive (lihat gambar 2.2)
glBegin(GL_TRIANGLE_FAN);	Segitiga	Objek primitive (lihat gambar 2.2)
glBegin(GL_QUADS);	Segiempat	Objek primitive (lihat gambar 2.2)
glBegin(GL_QUAD_STRIP);	Segiempat	Objek primitive (lihat gambar 2.2)
glBegin(GL_POLYGON);	Poligon	Objek primitive (lihat gambar 2.2)
glBegin(GL_LINE_STIPPLE);	Garis putus-putus	Objek primitive
glBegin(GL_POLY_STIPPLE);	Poligon dengan pola tertentu	Objek primitive
glRect(GLint x1, GLint y1, GLint x2, GLint y2);	Segiempat siku-siku	Objek primitive dan ukuran segiempat ditentukan oleh dua titik yaitu (x1,y1) dan (x2,y2)
glEnd( );	Akhir perintah OpenGL	-

Tabel 2.2 Format Fungsi OpenGL

Suffix	Tipe data	C atau C++	OpenGL
B	Integer 8-bit	signed char	GLbyte
S	Integer 16-bit	short	GLshort
I	Integer 32-bit	int atau long	GLint, GLsizei
F	Floating 32-bit	float	GLfloat, GLclampf
D	Floating 64-bit	double	GLdouble, GLclampd

Ub	unsigned 8-bit	unsigned char	GLubyte, GLboolean
Us	unsigned 16-bit	unsigned short	GLushort
Ui	unsigned 32-bit	unsigned int atau unsigned long	GLuint, GLenum, GLbitfield

1.1 Objek 3 Dimensi

Objek tiga dimensi adalah sebuah model struktur data yang menyatakan suatu gambar 3D dibentuk dan disusun. Objek 3D didefinisikan dengan :

1. Objek 3D adalah sekumpulan titik-titik 3D (x,y,z) yang membentuk luasan-luasan *(face)* yang digabungkan menjadi satu kesatuan.
2. *Face* adalah gabungan titik-titik yang membentuk luasan tertentu atau sering dinamakan dengan sisi.

Dari definisi ini, dapat dikatakan bahwa objek 3D merupakan kumpulan titik-titik dan kumpulan face yang merupakan susunan dari titik-titik yang ditentukan. Seperti gambar kubus, kubus terdiri dari 8 titik dan 6 sisi/*face*.Dimana *face* merupakan polygon atau kumpulan titik-titik yang disusun urutannya. Dalam kubus, *face*-nya semua berupa bujursangkar dengan 4 titik.

1.2 Bitmap

Bitmap adalah representasi atau gambaran yang terdiri dari baris dan kolom pada titik image graphics di komputer. Nilai dari titik disimpan dalam satu atau lebih data bit. Tampilan dari bitmap atau raster, menggunakan titik-titik berwarna yang dikenal dengan sebutan pixel. Pixel-pixel tersebut ditempatkan pada lokasi-lokasi tertentu dengan nilai-nilai warna tersendiri, yang secara keseluruhan akan membentuk sebuah tampilan gambar. Tampilan bitmap mampu menunjukkan kehalusan gradasi bayangan dan warna dari sebuah gambar, karena itu bitmap merupakan media elelktronik yang paling tepat untuk gambar-gambar dengan perpaduan gradasi warna yang rumit seperti foto dan lukisan digital.

Struktur bitmap terdiri dari Header, Info Header dan Color Tabel. Header adalah bagian dari file bitmap yang berisi informasi header dari file gambar bitmap. Ukuran dari header ini 14 byte,

masing-masing terdiri dari signature 2 bytes (berisi “BM” sebagai tanda gambar mempunyai format bmp), FileSize 4 bytes (besarnya ukuran gambar mempunyai satuan bytes), Reserved 4 bytes (tidak digunakan atau sama diisi dengan nilai nol) dan Data Offset 4 bytes (file offset untuk raster data). Info header adalah bagian dari header yang berisi informasi lebih detail dari file gambar bitmap. Color table adalah table yang berisi warna-warna yang ada pada gambar bitmap.

1.3 Lighting (Diffuse, Ambient dan Specular)

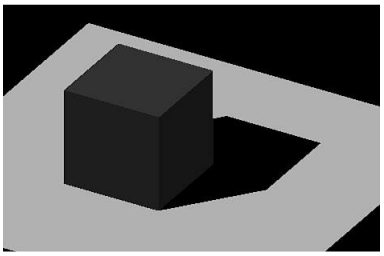
Proses menghitung intensitas cahaya terutama pada 3D point, biasanya diatas suatu permukaan :

- Bayangan

Bayangan akan muncul saat cahaya jatuh menyinari suatu objek.Pada dunia maya, layaknya cahaya, terdapat beberapa jenis bayangan yang dapat dihasilkan oleh komputer.

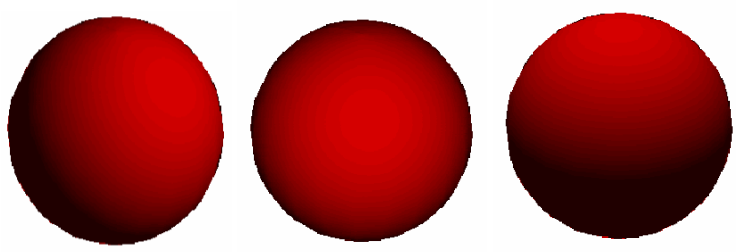
- Jenis Bayangan

Pada Maya, suatu sumber cahaya bisa tidak menghasilkan bayangan (*default*) atau bisa menghasilkan bayangan *depthmap* maupun *raytraced*. Anda dapat mengombinasikan kedua jenis bayangan ini *depthmap* maupun *raytraced* pada *scene* Anda. Dengan mengatur atribut bayangan *depth map* atau *raytraced*, Anda dapat mensimulasikan Bayangan yang dihasilkan oleh berbagai tipe cahaya di dunia nyata. Bayangan *depth map* maupun *raytraced*memberikan efek yang hamper sama, namun bayangan depth map waktu *render*-nya lebih cepat. Umumnya kebanyakan orang akan memilih bayangan depth map kecuali jika tipe bayangan tersebut tidak dapat membantu mencapai visualisasi yang diinginkan.



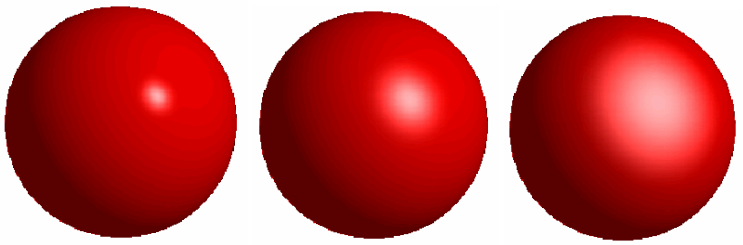
Gambar 1 Penambahan bayangan akan meningkatkan kesan realistik

- Model Bayangan dibagi menjadi beberapa bagian :
  1. Cahaya *diffuse*(tersebar)
     - Perhitungan cahaya tersebar menggunakan m,v dan s.
     - Sebagaimana cahaya tersebar disebarkan secara seragam dalam semiua arah, lokasi mata, v, tidak penting kecuali kalau v. m < 0 jika diinginkan intensitas cahaya l = 0
     - Hubungan antara kecerahan permukaan dan orientasinya cahaya.



Gambar 2 Contoh *Diffuse*

  2. Cahaya *Specular* (Spekular)
     - Objek nyata tidak menyebarkan cahaya secara seragam
     - Komponen spekular perlu diperhitungkan
     - Digunakan model phong, jumlah cahaya dipantulkan terbesar dalam arah pantulan cermin
     - Ini merupakan arah semua lintasan cahaya terpantul untuk cermin yang sempurna



Gambar 3 Contoh *Specular*

  3. Cahaya *Ambient* (Lingkungan)
     - Cahaya jatuh ke objek dari berbagai sudut pantul dari objek lain dan lingkungan
     - Secara perhitungan sangat mahal untuk di hitung
     - Cahaya ambient tidak mempunyai titik asal khusus.
     - Koefesien cahaya ambient : p_a, dipakaikan untuk masing-masing permukaan.

- Sumber intensitas l_a dikalikan dengan koefesien dan digunakan dan ditambahkan ke sembarang cahaya tersebar atau spekular.

1.4 *Mapping* (Pemetaan)

Texture mapping merupakan teknik pemetaan sebuah tekstur pada pola gambar wireframe, dimana wireframe yang telah dibuat akan ditampilkan memiliki kulit luar seperti tekstur yang diinginkan. Dalam pemberian tekstur, perlu diperhatikan dasarnya seperti:

- Menentukan tekstur
  - Membaca atau membangkitkan tekstur
  - Menandai tekstur
  - Mengenablekan tekstur
- Menandai koordinat tekstur pada vertex
- Menentukan parameter tekstur
  - *wrapping* , *filtering*, dsb.

Langkah-langkah dalam memulai mapping sebuah tekstur yaknidengan spesifikasi dibawah ini :

a. Menentukan Tekstur Image

Cara menentukan tekstur image adalah sebagai berikut :

1.Mendefinisikan tekstur image dari sebuah array teksel (element tekstur) ke dalam memory cpu : Glubyte my_texels[512][512];

2.Mendefinisikan seperti semua peta piksel yang lain

- ✓ Gambar yang didefinisikan (baik secara manual maupun dengn suatu fungsi matematik tertentu)
- ✓ Membangkitkan dengan kode aplikasi

3. Mengenablekan tekstur mapping
   - ✓ glEnable(GL_TEXTURE_2D)
   - ✓ OpenGL mendukung 1 sampai 4 dimensional tekstur mapping

b. Mendefinisikan gambar sebagai sebuah tekstur

glTexImage2D(target,level,components,w,h,border,format,type, texels);

Keterangan :

- ✓ target: tipe dari teksture, e.g. GL_TEXTURE_2D
- ✓ level: digunakan untuk *mipmapping*
- ✓ components: element per texel
- ✓ w, h: lebar dan tinggi dari texels pada pixels
- ✓ border: digunakan untuk smoothing
- ✓ format and type: menjelaskan texels
- ✓ texels: pointer ke array texel
- ✓ glTexImage2D(GL_TEXTURE_2D, 0,3, 512, 512, 0, GL_RGB, GL_UNSIGNED_BYTE, my_texels);

c. Mengubah gambar tekstur :

- ✓ OpenGL meminta dimensi tekstur untuk menjadi dasar dari 2
- ✓ Jika dimensi dari image bukan power ke 2,
- ✓ gluScaleImage(format,w_in,h_in,type_in,*data_in,w_out,h_out,type_out,*data_out);
- ✓ data_in adalah gambar inputan.
- ✓ data_out adalah gambar hasil

d. Mapping Tekstur :

- ✓ Didasarkan pada koordinat tekstur *parametric*
- ✓ glTexCoord*() ditetapkan pada masing – masing *vertex*

e. Parameter tekstur

- ✓ OpenGL mempunyai berbagai parameter yang menentukanbagaimana tekstur diterapkan
- ✓ Parameters wrapping menentukan apa yang terjadi pada s dan t
- ✓ diluar jangkauan (0,1)
- ✓ Mode filter mengijinkan penggunaan area rata-rata sebagai ganti contoh titik
- ✓ Mipmapping mengijinkan penggunaan tekstur pada berbagai resolusi

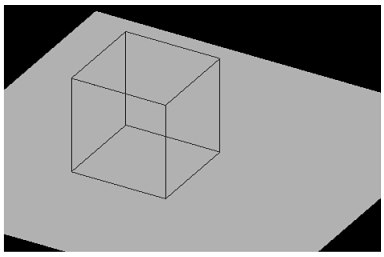
- ✓ Parameter lingkungan menentukan bagaimana tekstur mapping berinteraksi dengan shading

f. Magnification dan Minification

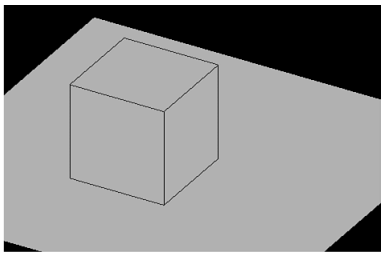
- ✓ Lebih dari satu texel dapat dicakup sebuah pixel (minification) atau lebih dari satu piksel dapat dicakup sebuah texel (magnification). Dapat menggunakan contoh titik (texel terdekat) atau filter linier (filter 2x2) untuk mengisi nilai tekstur
- ✓ Modes ditentukan oleh :
- ✓ glTexParameteri(target, type, mode)
- ✓ glTexParameteri(GL_TEXTURE_2D,GL_TEXURE_MAG_FILTER,GL_NEAREST);
- ✓ glTexParameteri(GL_TEXTURE_2D,GL_TEXURE_MIN_FILTER,GL_LINEAR);
- ✓ Catatan bahwa filter linier tersebut meminta sebuah batas dari extra texel untuk filter pada tepi (batas = 1)

1.5 *Perspective*

Perspective adalah sudut-sudut diantara garis yang membuat kenampakan dari ilusi 3 dimensi. Gambar 6 berikut merupakan gambar kubus 3 dimensi yang disusun dari garis-garis. Gambar tersebut telah mengilustrasikan sebuah kubus, tetapi masih menimbulkan persepsi yang membingungkan. Sedangkan Gambar 7 memberikan ilusi 3 dimensi yang lebih kuat dengan menghilangkan garis yang tertutup (*hidden line remofal*) dan menghapus bidang atau sisi belakang yang tak nampak (*hidden surface remofal*) untuk menghasilkan kenampakan permukaan yang solid.



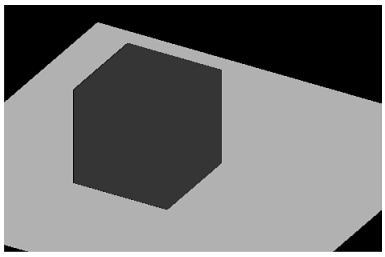
Gambar 4Kubus 3 Dimensi yang tersusun dari Garis



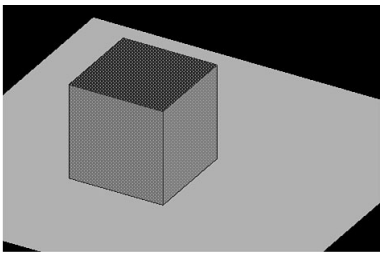
Gambar 5Image Kubus yang lebih baik

1.6 *Color* and *Shading*

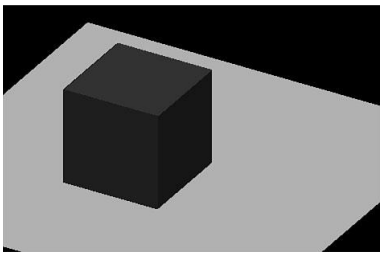
Untuk memberikan kesan bidang 3 dimensi yang lebih baik kita dapat memberikan pewarnaan terhadap objek tersebut. Tetapi teknik pewarnaan tersebut harus digunakan dengan hati-hati untuk menghasilkan efek diinginkan. Jika kita mewarnai kubus dengan suatu warna yang sama justru akan menghilangkan kesan 3 dimensi dari kubus tersebut seperti pada gambar 8 (seluruh bidang diberi warna yang sama). Sedangkan teknik pewarnaan yang baik akan menambah kesan ilusi 3 dimensi seperti pada gambar 9 (setiap bidang diberi warna yang berbeda). Sama halnya dengan pemberian shading seperti pada gambar 10 akan memberikan kenampakan/ilustrasi objek 3 dimensi yang lebih kuat.



Gambar 6 Penambahan warna yang menimbulkan kebingungan



Gambar 7 Menambahkan warma yang berbeda akan menimbulkan ilusi 3 dimensi yang lebih baik



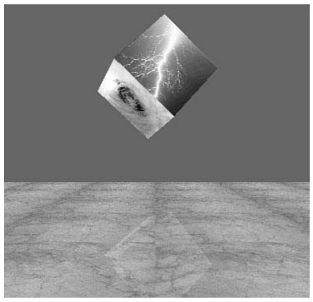
Gambar 8 Pemberian Shading yang cukup akan meningkatkan ilusi obek 3 dimensi

1.7 *Blending*(Pencampuran)

Pencampuran merupakan fungsi yang menggabungkan nilai warna dari sumber dan tujuan. Pencampuran dapat mengontrol berapa banyak warna yang dapat dikombinasikan. Dengan demikian untuk proses membuat terang fragmen dapat menggunakan alpha pencampuran. Warna pencampuran terletak pada teknik utama, seperti transparan, digital composite dan lukisan. Operasi campuran yaitu cara yang paling alami untuk mengetahui bahwa komponen RGB adalah suatu fragmen yang mewakili warna dan komponen alfa adalah suatu fragmen yang mewakili sifat tidak tembus cahaya. Dengan demikian, transparan mempunyai permukaan lebih rendah daripada yang buram.

Dengan menggunakan glBlendFunc () untuk persediaan pada dua hal utama, yang pertama menentukan bagaimana faktor sumber dan tujuan harus dihitung dan yang kedua menunjukan bagaimana faktor sumber dan tujuan dihitung. Dan untuk proses pencampurannya harus ada faktor pengaktifannya menggunakan : glEnable (GL_BLEND). Menggunakan **glDisable ()** dengan GL_BLEND untuk menonaktifkan Pencampuran dan menggunakan konstan GL_ONE (sumber)

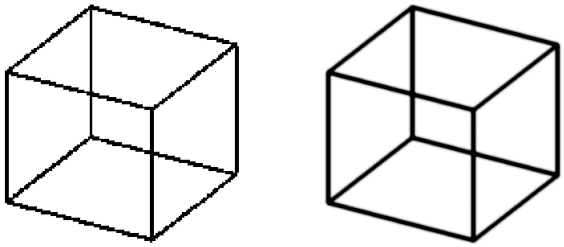
dan GL_ZERO (tujuan) memberikan hasil yang sama seperti ketika Pencampuran dinonaktifkan. Nilai-nilai ini bersifat default dengan void **glBlendFunc** (GLenum *sfactor*, GLenum *dfactor)*.Teknik *blending* juga digunakan untuk ilusi dari cerminan sebuah objek pada gambar 11.



Gambar 9 Tehnik *blending* untuk mendapatkan efek pantulan

1.8 *Antialiasing*

Antialiasing adalah efek yang muncul akibat fakta bahwa image yang dibuat tersusun oleh pixel-pixel yang bersifat diskrit. Seperti pada gambar 12, terlihat bahwa garis yang membentuk kubus tersebut memiliki *jagged edges* (*jaggies*). Dengan melakukan *blending* pada garis terhadap warna latar, kita dapat menghilangkan *jaggies* dan menghasilkan tampilan garis yang mulus (*smooth*). Teknik tersebut dinamai teknik *Antialiasing*.



Gambar 10*Jagged Lines VS Smooth Lines*

1.9 *Translation*

Sebuah objek dalam tiga dimensi di translasikan dengan mengubah setiap setiao point yang mendefinisikan objek. Untuk sebuah objek yang dibangun atau direpresentasikan dengan sebuah set dari permukaan – permukaan poligon , translasi dilakukan pada setiap permukaan dan menggambar kembali permukaan poligon pada posisi baru.

1.10 *Rotation*

Untuk membangkitkan rotasi pada objek 3D kita harus membuat aksis dari rotasi dan jumlah sudut rotasi . Tidak seperti melakukan rotasi pada objek 2D yang semua proses transformasi dilakukan di koordinat xy , sebuah rotasi objek tiga dimensi bisa dilakukan di *space* manapun.

Dengan menggunakan notasi matrix, maka besaran R bisa dikatakan sbb:

$$\mathbf{R} = \begin{pmatrix} \cos(0) & \sin(0) \\ -\sin(0) & \cos(0) \end{pmatrix}$$

1.11 *Scaling*

Penskalaan pada objek 3D dengan transformasi mengubah ukuran dan posisi objek relatif terhadap koordinat asli. Jika tidak semua parameternya sama dimensi – dimensi relative pada objek akan diubah.

Matriks transformasi untuk skala dapat juga dinyatakn sebagai berikut :

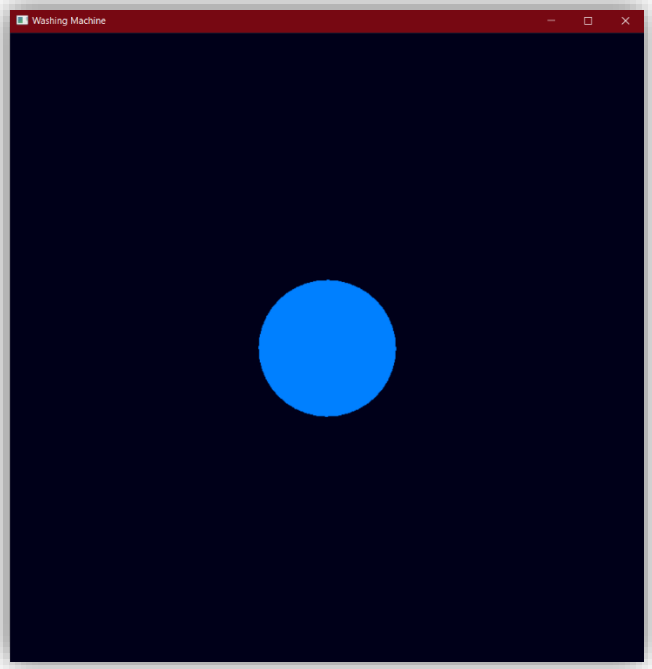
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

BAB III
IMPLEMENTASI

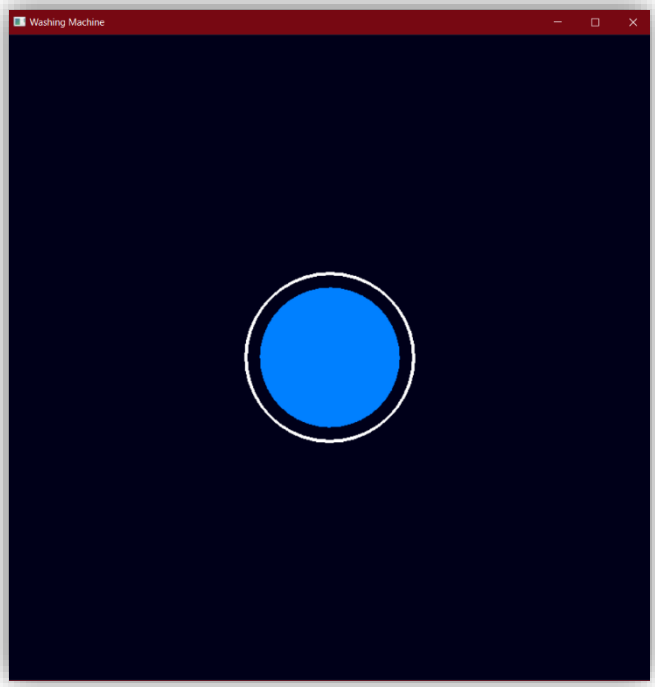
3.1 Tampilan Awal

Berikut ini merupakan tampilan awal model 3D menggunakan objek Washing Machine dengan menggunakan aplikasi Dev C++. Tampilan awal dari aplikasi dapat dilihat pada gambar berikut ini.

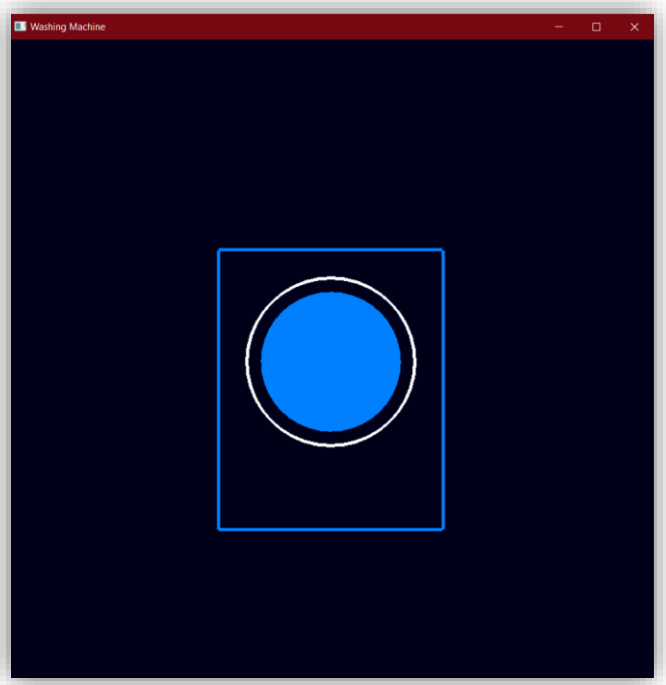
3.1.1 Proses Pembuatan Lingkaran



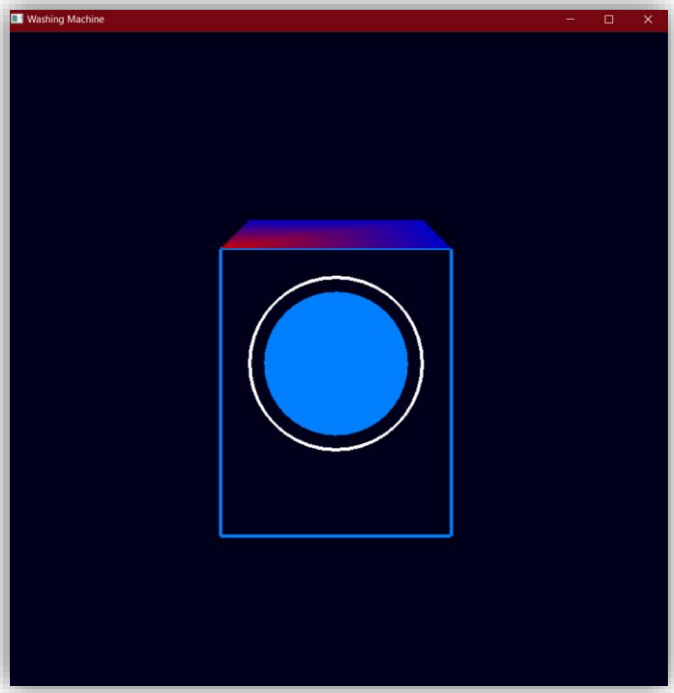
3.1.2 Proses Pembuatan Lingkaran Luar



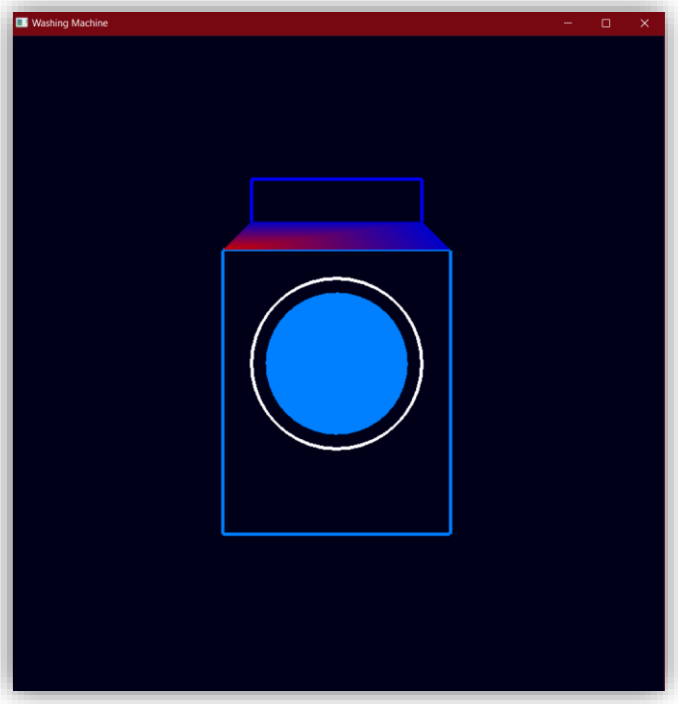
3.1.3 Proses Pembuatan Kotak



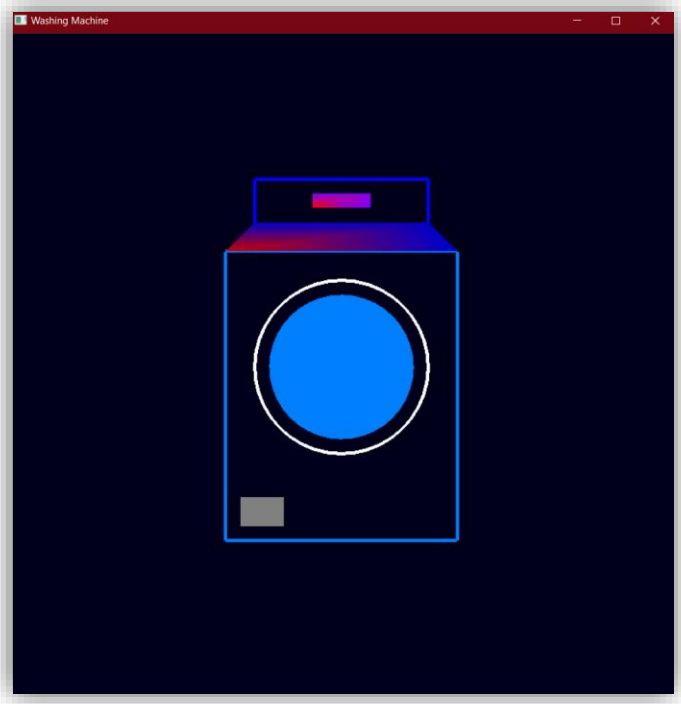
3.1.4 Proses Pembuatan Kotak Miring



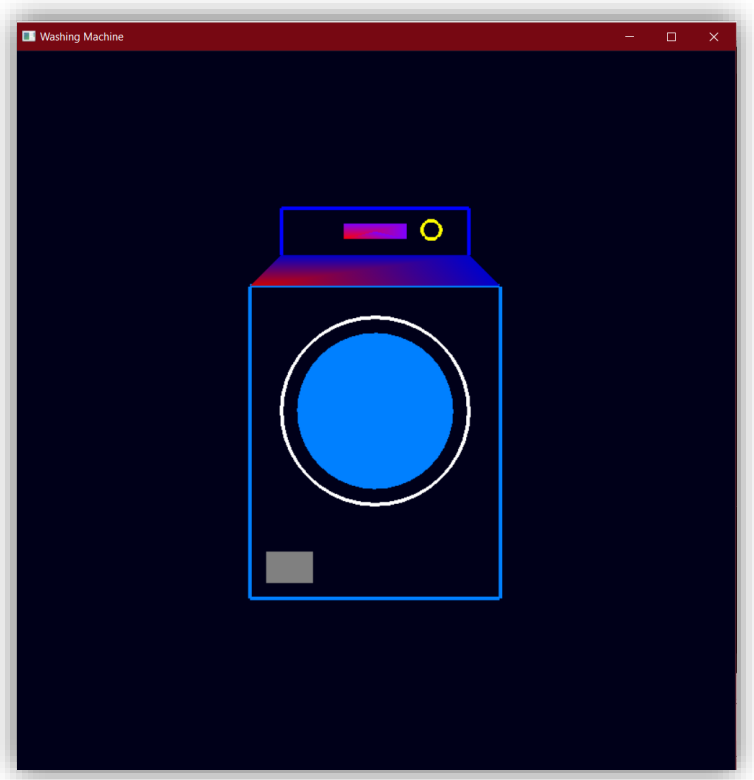
3.1.5 Proses Pembuatan Kotak Atas



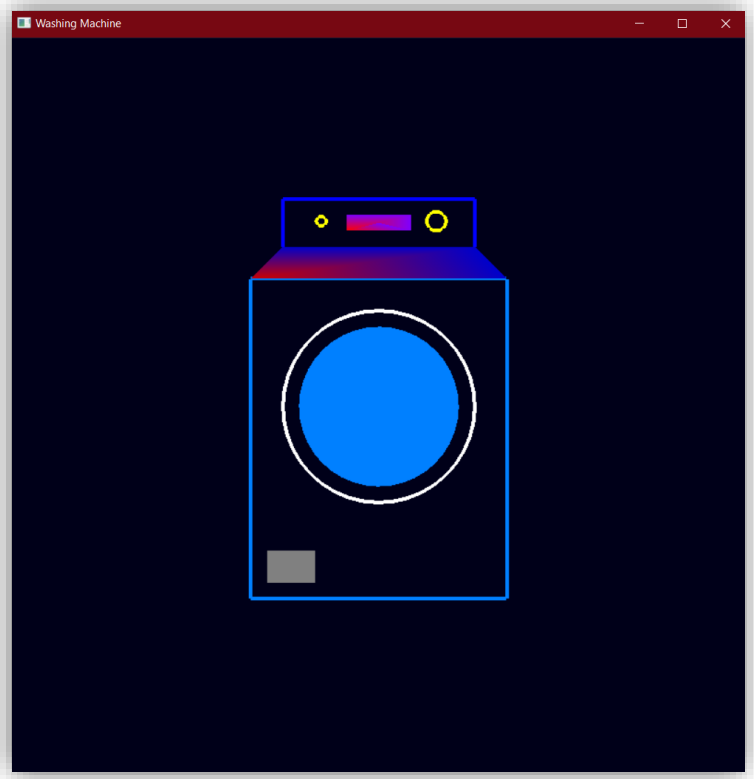
3.1.6 Proses Pembuatan Kotak Pojok Bawah



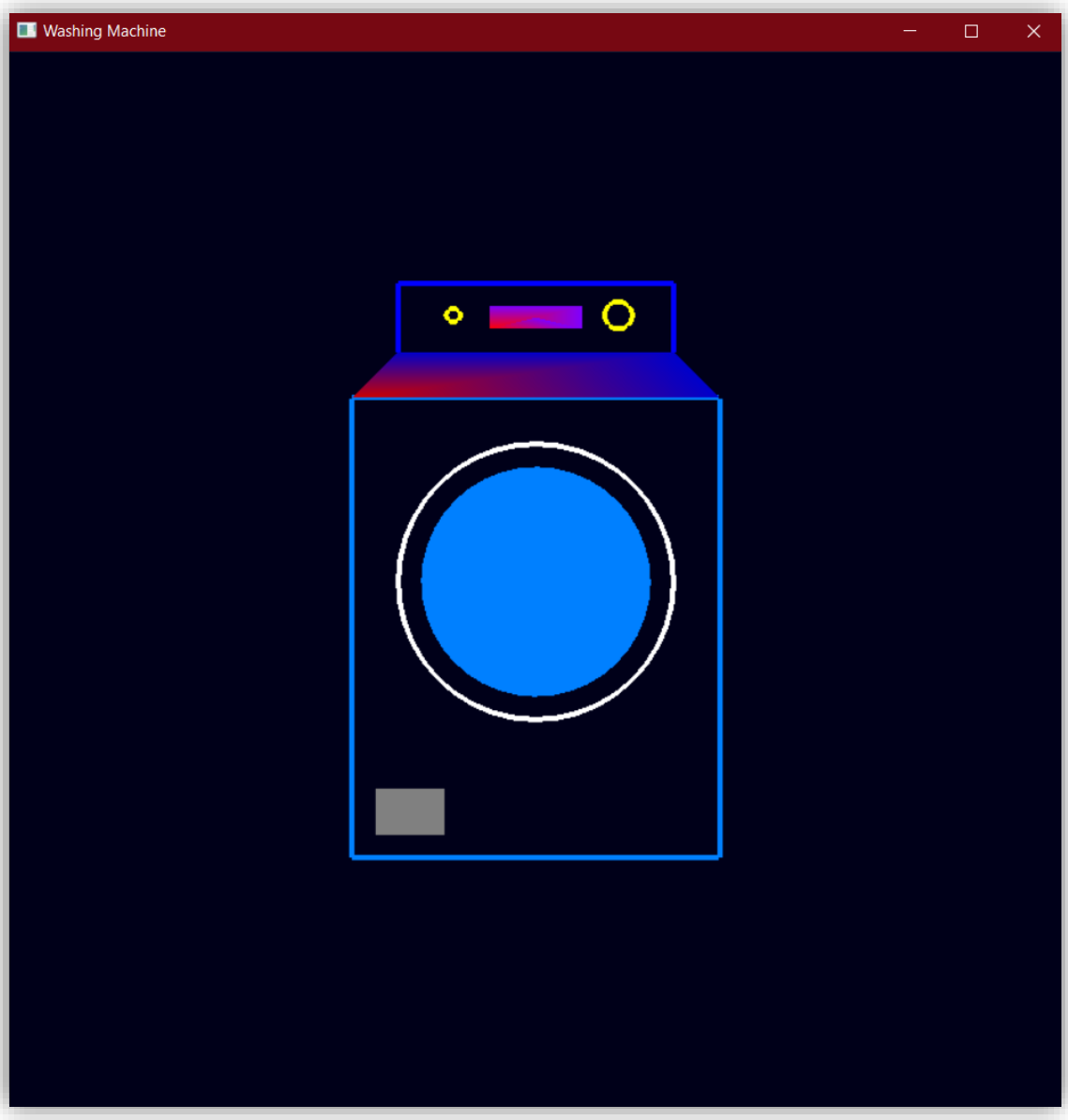
3.1.7 Proses Pembuatan Lingkaran Atas



3.1.8 Proses Pembuatan Lingkaran Atas Kecil



3.1.9 Hasil Akhir Washing Machine



3.2 Source Code

```
#ifdef __APPLE__
#include <GLUT/glut.h>
#else
#include <GL/glut.h>
#endif
#include <math.h>
#include <stdlib.h>


//Mendeklarasikan
const double PI = 3.14 ; //NILAI PHI

float i, radius, jumlah_titik, x_tengah, y_tengah;
GLfloat xRotated, yRotated, zRotated; // deklarasi sumbu rotasi x,y,z

void bentuk(void) { //menampilkan

glClear(GL_COLOR_BUFFER_BIT);

glColor3f(1.0, 1.0, 1.0);

glLineWidth(4.0f);

glLoadIdentity(); //definisi objek

glTranslatef(0.0,0.0,-2.0); //translasi

// Menentukan Bentuk Rotasi
glRotatef(xRotated,1.0,0.0,0.0);
```

```
glRotatef(yRotated,0.0,1.0,0.0);
glRotatef(zRotated,0.0,0.0,1.0);
```

//PROSES PEMBUATAN LINGKARAN1 (WARNA)

```
glBegin(GL_POLYGON);

glColor3f(0.0,0.5,1.0);

radius=25; //ukuran lingkaran

jumlah_titik=40;

x_tengah=0;

y_tengah=0;
```

//proses pembuatan lingkaran

```
for(i=0; i<=360; i++) {

float sudut=i*(2*PI/jumlah_titik);

float x=x_tengah+radius*cos(sudut);

float y=y_tengah+radius*sin(sudut);

glVertex2f(x/100, y/100);
```

```
}

glEnd();
glFlush();
glutSwapBuffers(); //fungsi perpindahan


// PROSES PEMBUATAN LINGKARAN2 LUAR HITAM

glColor3f(1.0,1.0,1.0);

glBegin(GL_LINE_LOOP);

radius=30;

jumlah_titik=40;

x_tengah=0;

y_tengah=0;


//proses pembuatan lingkaran

for(i=0; i<=360; i++) {

float sudut=i*(2*PI/jumlah_titik);

float x=x_tengah+radius*cos(sudut);

float y=y_tengah+radius*sin(sudut);
```

```
glVertex2f(x/100, y/100);

}

glEnd();
glFlush();
glutSwapBuffers(); //fungsi perpindahan
```

//PROSES PEMBUATAN LINGKARAN SEDANG ATAS KANAN

```
glColor3f(1.0,1.0,0.0); //WARNA KUNING

glBegin(GL_LINE_LOOP);

radius=3;

jumlah_titik=40;

x_tengah=18;

y_tengah=58;
```

//proses pembuatan lingkaran sedang

```
for(i=0; i<=360; i++) {

float sudut=i*(2*PI/jumlah_titik);
```

```
float x=x_tengah+radius*cos(sudut);

float y=y_tengah+radius*sin(sudut);

glVertex2f(x/100, y/100);

}

glEnd();
glFlush();
glutSwapBuffers(); //fungsi perpindahan
```

//PROSES PEMBUATAN LINGKARAN ATAS KIRI KECIL

```
glColor3f(1.0,0.0,0.0); //WARNA MERAH

glBegin(GL_LINE_LOOP);

radius=1.5;

jumlah_titik=40;

x_tengah=-18;

y_tengah=58;
```

//proses pembuatan lingkaran

```
for(i=0; i<=360; i++) {
```

```
float sudut=i*(2*PI/jumlah_titik);

float x=x_tengah+radius*cos(sudut);

float y=y_tengah+radius*sin(sudut);

glVertex2f(x/100, y/100);

}

glEnd();
glFlush();
glutSwapBuffers(); //fungsi perpindahan
```

// PROSES PEMBUATAN KOTAK1 LUAR BESAR

```
glBegin(GL_LINES);
glColor3f(0.0,0.5,1.0);

glVertex3f(-0.4,-0.6,0.0); //garis kiri
glVertex3f(-0.4,0.4,0.0);

glVertex3f(0.4,-0.6,0.0); //garis kanan
glVertex3f(0.4,0.4,0.0);

glVertex3f(-0.4,-0.6,0.0); //garis bawah
glVertex3f(0.4,-0.6,0.0);

glVertex3f(-0.4,0.4,0.0); //garis atas
glVertex3f(0.4,0.4,0.0);
```

```
glEnd();
glFlush();
glutSwapBuffers(); //fungsi perpindahan
```

// PROSES PEMBUATAN KOTAK MIRING TENGAH BERWARNA

```
glBegin(GL_POLYGON);
glColor3f(0.8,0.0,0.0); //MERAH

glVertex3f(-0.4,0.4,0.0); //garis kiri
glColor3f(0.0,0.0,0.8); //BIRU KEMERAHAN
glVertex3f(-0.3,0.5,0.0);

glVertex3f(0.4,0.4,0.0); //garis kanan
glVertex3f(0.3,0.5,0.0);

glVertex3f(-0.3,0.5,0.0); //garis atas
glVertex3f(0.3,0.5,0.0);

glEnd();
glFlush();
glutSwapBuffers(); //fungsi perpindahan
```

//PROSES PEMBUATAN KOTAK3 ATAS

```
glBegin(GL_LINES);
glColor3f(0.0f,0.0f,1.0f);


glVertex3f(-0.3,0.5,0.0); //garis kiri
```

```
glVertex3f(-0.3,0.65,0.0);

glVertex3f(0.3,0.5,0.0); //garis kanan
glVertex3f(0.3,0.65,0.0);

glVertex3f(-0.3,0.65,0.0); //garis atas
glVertex3f(0.3,0.65,0.0);

glEnd();
glFlush();
glutSwapBuffers(); //fungsi perpindahan
```

// PROSES PEMBUATAN KOTAK ATAS TENGAH BERWARNA

```
glBegin(GL_POLYGON);
glColor3f(1.0,0.0,0.0); //MERAH


glVertex3f(-0.1,0.55,0.0); //garis kiri
glColor3f(0.5,0.0,1.0); //BIRU KEMERAHAN
glVertex3f(-0.1,0.6,0.0);

glVertex3f(0.1,0.55,0.0); //garis kanan
glVertex3f(0.1,0.6,0.0);

glVertex3f(-0.1,0.55,0.0); //garis bawah
glVertex3f(0.1,0.55,0.0);

glVertex3f(-0.1,0.6,0.0); //garis atas
glVertex3f(0.1,0.6,0.0);
```

```
glEnd();
glFlush();
glutSwapBuffers(); //fungsi perpindahan
```

// PROSES PEMBUATAN KOTAK5 POJOK BAWAH

```
glBegin(GL_POLYGON);
glColor3f(0.5,0.5,0.5);

glVertex3f(-0.35,-0.55,0.0); //garis kiri
glVertex3f(-0.35,-0.45,0.0);

glVertex3f(-0.2,-0.55,0.0); //garis kanan
glVertex3f(-0.2,-0.45,0.0);

glVertex3f(-0.35,-0.55,0.0); //garis bawah
glVertex3f(-0.2,-0.55,0.0);

glVertex3f(-0.35,-0.45,0.0); //garis atas
glVertex3f(-0.2,-0.45,0.0);

glEnd();
glFlush();
glutSwapBuffers(); //fungsi perpindahan
}
void Reshape(int x,int y) //fungsi
{
```

//PERINTAH PERSPEKTIF 3D

```
if (y==0 || x==0) return;

glMatrixMode(GL_PROJECTION); //fungsi mode proyeksi

glLoadIdentity();

gluPerspective(60.0,(GLdouble)x/(GLdouble)y,0.5,20.0);

glMatrixMode(GL_MODELVIEW); //mengatur model display objek

}

void idle(void)

{

// ROTASI

xRotated +=0.4; //mengatur kecepatan rotasi bergerak sumbu x

yRotated +=0.7; //mengatur kecepatan rotasi bergerak sumbu y

zRotated +=0.5; //mengatur kecepatan rotasi bergerak sumbu z

bentuk(); //menampilkan gerakan rotasi

}

int main( int argc, char* argv[] ) {
```

```
// Inisialisasi GLUT
glutInit(&argc, argv);

// Membuat kanvas dengan dukungan warna RGBA.
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);

 // Membuat jendela dengan ukuran 800x800 pixel
glutInitWindowSize(800,800);

// Membuat jendela dengan judul "Washing Machine"
glutCreateWindow("Washing Machine");

// Set warna background (R, G, B, A)
glClearColor(0.0f, 0.0f, 0.1f, 0.2f);

// Memanggil fungsi untuk membuat gambar
glutDisplayFunc(bentuk);
glutReshapeFunc(Reshape);
glutIdleFunc(idle);

// Memulai "Windows Message Pump" untuk menampilkan jendela
glutMainLoop();
return 0;

}
```

BAB IV
KESIMPULAN

Setelah melakukan perancangan objek 3D yaitu model 3D Washing Machine, penulis mengambil kesimpulan sebagai berikut :

1. Penulis sedikit mengetahui fungsi-fungsi OpenGL dalam aplikasi 3D.
2. Penulis sedikit mengetahui teknik-teknik 3D dengan beberapa fungsi memberi kesan nyata pada objek yang dibuat.