LAPORAN FINAL PROJECT

UJIAN AKHIR SEMESTER GRAFIKA KOMPUTER

"3D ANIMASI MOBIL BERGERAK"

Disusun Oleh:

Nama Anggota Kelompok:

Alvira Rhiza Ridwani (19051397007)

Nur Wulan Cahyani (19051397010)

Tania Nastika Putri Mosha (19051397026)

UNIVERSITAS NEGERI SURABAYA

FAKULTAS TEKNIK

JURUSAN TEKNIK INFORMATIKA

PRODI MANAJEMEN INFORMATIKA 2019

2020/2021

KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Esa karena atas rahmat dan anugerah yang diberikan, penulis dapat menyelesaikan Laporan Akhir Project Ujian Akhir Semester. Laporan ini bertujuan sebagai pemenuhan ujian akhir semester mata kuliah Grafika Komputer yang berjudul “3D ANIMASI”.

Pada kesempatan ini, penyusun mengucapkan terima kasih kepada dosen pembimbing atas bimbingan dan arahannya. Dan terima kasih juga untuk pihak-pihak yang telah membantu penulis dalam penyusunan laporan ini. Penulis berharap agar laporan yang disusun ini dapat bermanfaat bagi setiap pembaca.

Penulis sadar dalam pembuatan laporan ini masih banyak kekurangan serta jauh dari kata sempurna. Dengan itu, penulis berharap adanya kritik dan saran yang membangun untuk perbaikan laporan yang hendak ditulis. Selanjutnya, menyadari tidak ada suatu hal yang sempurna tanpa saran yang konstruktif.

DAFTAR ISI

BAB I 1

1.1 Latar Belakang 1

1.2 RUMUSAN MASALAH 2

1.3 BATASAN MASALAH 2

BAB II 3

2.1 Objek 3 Dimensi 3

2.2 Fungsi OpenGL 3D 4

BAB III 5

3.2 Pembagian Kerja 5

3.3 Tampilan Program 5

3.4 Interaksi objek dengan mouse dan keyboard 7

BAB IV 8

4.1 Kesimpulan……..11

4.2 Saran……11

BAB I

PENDAHULUAN

1.1 Latar Belakang

Grafika Komputer merupakan salah satu cabang disipllin ilmmu informatika yang mempelajari pembuatan dan manipulasi gambar dengan komputer atau secara digital. Grafika komputer dikenal dengan istilah visualisasi data. Grafika komputer pertama kali dikembangkan oleh The Whirlwind Computer padaa tahun 1950 untuk memperagakan output dari suatu hardcopy. Kemudia dikembangkan lagi oleh Dr. Ivan Suherland sehingga menghasilkan fitur-fitur grafika saat ini.

Pembuatan suatu program yang bergerak diperlukan menampilkan objek—objek dengan menggunakan bahasa C dengan openGL. OpenGL merupakan sebuah program aplikasi interface yang digunakan untuk mendefinisikan komputer grafis 2D dan 3D. Program lintas-platform API ini umumnya dianggap ketetapan standar dalam industri komputer dalam interaksi dengan komputer grafis 2D dan juga telah menjadi alat yang biasa untuk digunakan dengan grafis 3D.

Oleh karena itu, semakin banyak bahasa pemrograman yang dilengkapi dengan tools pembuatan grafik. Salah satu tools pembuatan aplikasi grafik adalah OpenGL. Penulis menggunakan tools openGL mempermudah pembuatan grafik primitif termasuk titik, garis, dan lingkaran.

Pengerjaan yang begitu kompleks menyebabkan program tersebut tidak dapat dilakukan oleh perorangan, maka dari itu pengerjaan program ini dilakukan

perkelompok, efisiensi daan kerja sama dibutuhkan dalam proses pengerjaan program menampilkan objek-objek menggunakan bahasa C.

Objek yang dibuat merupakan gabungan dari objek yang menjadi kesatuan yang dapat berinteraksi dengan yang menjalankan program tersebut.

1.2 RUMUSAN MASALAH

Bagaimana cara memodelkan mobil bergerak dijalan dalam bentuk 3D animasi pada OpenGL dengan menggunakan aplikasi CodeBlocks?

1.3 BATASAN MASALAH

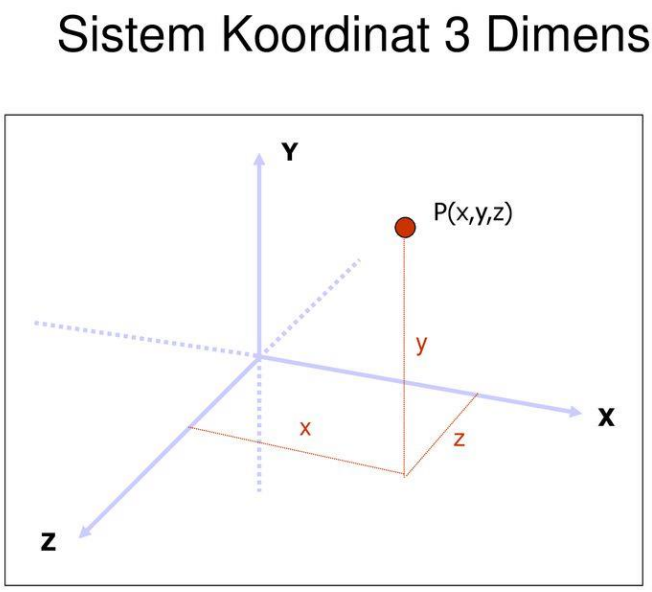
1. Membahas tentang bagaimana cara membuat dan merancang bentuk mobil bergerak 3D dengan menggunakan OpenGL
2. Membahas aplikasi pembuatan bentuk 3D menggunakan Codeblocks dengan library OpenGL.

BAB II

LANDASAN TEORI

2.1 Objek 3 Dimensi

Objek 3D merupakan sekumpulan titik-titik 3D yang membentuk luasan-luasan (face) yang digabungkan menjadi satu kesatuan. Luasan-luasan yang dimaksud adalah gabungan titik-titik yang membentuk luasan tertentu atau sering dinamakan sisi.



Gambar 2.1 Koordinat 3Dimensi

2.2 Fungsi OpenGL 3D

1. Membersihkan Windows

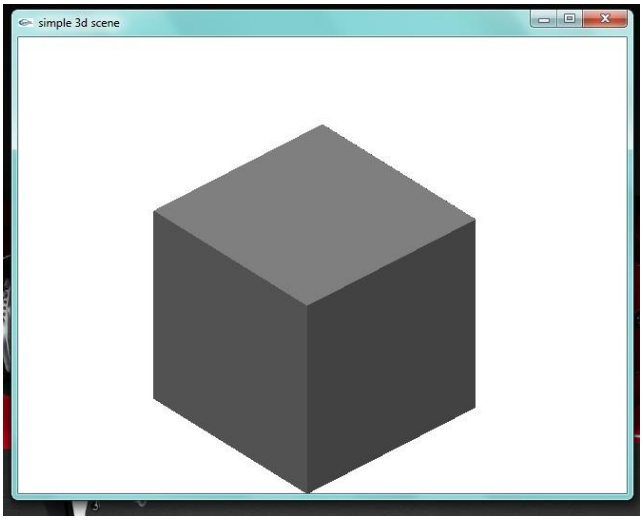
Pada komputer, memory untuk menampilkan gambar biasanya diisi dengan gambar yang berasal dari perintah gsmbar paling akhir, jadi perlu dibersihkan dengan warna latar belakang sebelum digambar lagi.

2. Spesifikasi Warna

OpenGL mendeskripsikan objek dengan warna objek adalah proses yang berjalan sendiri-sendiri. Sebelum warna diubah maka semua objek yang digambar sesudah perintah tersebut akan menggunakan warna terakhir yang terdapat pada *coloring scheme*.

3. Memaksa Proses Menggambar Sampai Selesai

Kebanyakan sistem grafik modern sudah menggunakan sistem *grapics pipeline*. CPU utama memberikan issue perintah menggambar dan hardware lain.



Gambar 2.2 3D Animasi Kubus

BAB III

IMPLEMENTASI

3.1 Perencanaan Kerja

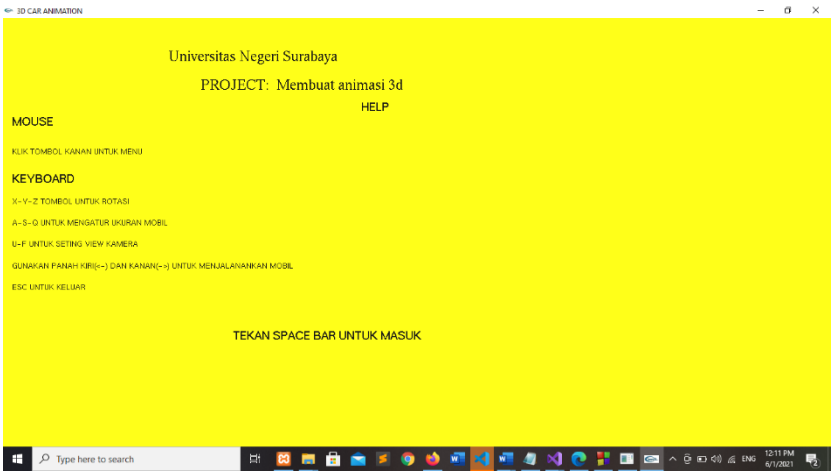
Kinerja dalam proses pembuatan program ini harus dilakukan semaksimal mungkin agar mendapatkan hasil yang baik.

3.2 Pembagian Kerja

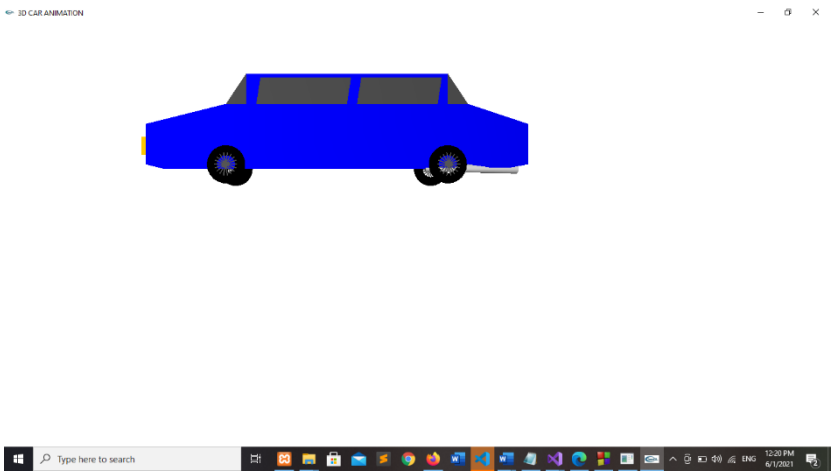
Pembagian Kerja dalam pengerjaan Final Project Ujian Akhir Semester dibagi sedemikian mungkin agar mendapatkan hasil yang memuaskan.

Nama Mahasiswa	Tugas
Alvira Rhiza Ridwani	Penentuan Konsep dan Pengerjaan design
Nur Wulan Cahyani	Penentuan Konsep dan Pengerjaan design
Tania Nastika Putri Mosha	Pengerjaan Aplikasi dan Pembuatan Laporan

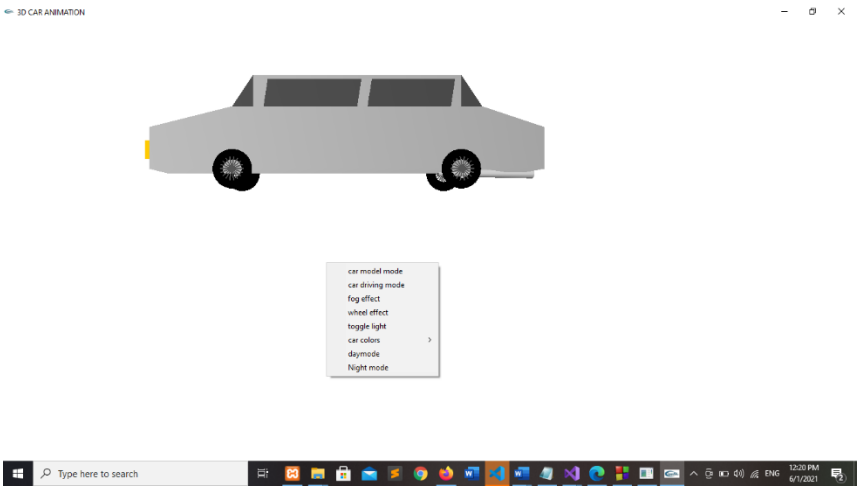
3.3 Tampilan Program



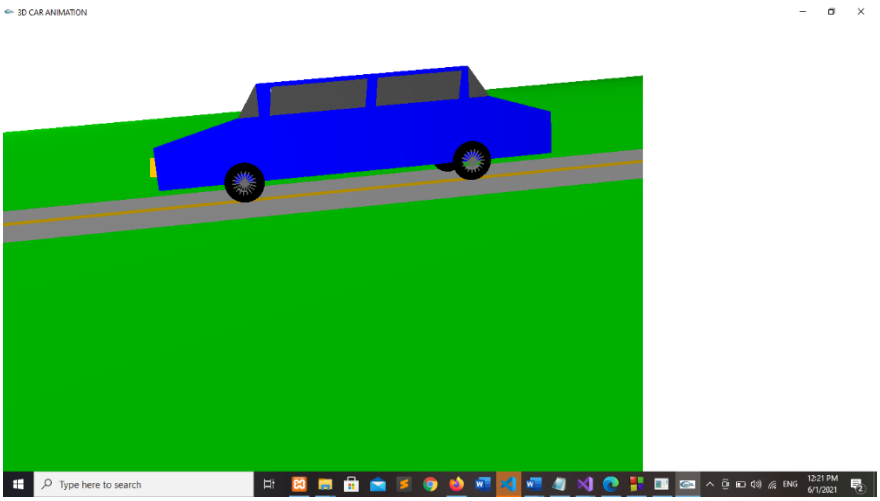
Gambar 1. Tampilan Program



Gambar 2. Tampilan Program



Gambar 3. Tampilan Program



Gambar 4. Tampilan Program

3.4 Interaksi objek dengan mouse dan keyboard

Interaksi	Fungsi
Mouse(Klik tombol kanan)	Untuk melihat menu yang akan digerakkan
Keyboard(X-Y-Z)	Untuk tombol rotasi
Keyboard(A-S-Q)	Untuk mengatur ukuran mobil
Keyboard(U-F)	Untuk setting view kamera
Panah kanan -> dan panah kiri<-	Untuk menggerakan mobil

BAB IV

KESIMPULAN DAN SARAN

4.1 Kesimpulan

Ketika dalam pembuatan grafik di jaman serba teknologi ini kita bebas menentukan model setiap animasi 3D, untuk memahami secara mendalam transformasi didalam grafik komputer sangat penting untuk menguasai ilmu sistematik. Perbandingan hasil pembuatan program dengan menggunakan bahasa pemrograman lebih sulit dan berbeda jauh dari segi tampilan, maupun tata cara pembuatannya dibandingkan dengan program aplikasi yang menerapkan sistem just click

4.2 Saran

Untuk saran yang harus diperhatikan adalah memahami syntax yang terfapat dalam bahasa pemrograman seperti pembuatan animasi 3D ini ataupun memahami fitur-fitur yang ada pada aplikasi tersebut. Selain itu juga, pilih program aplikasi yang lebih kompleks untuk pembuatan grafik serta yang mengikuti perkembangan modern.

LAMPIRAN

```
#include <stdio.h>
#include <stdlib.h>
#include <gl/glut.h>
#include <math.h>
#include <string.h>

#define ESCAPE 27

GLint window;
GLint window2;
GLint Xsize=1000;
GLint Ysize=800;
float i,theta;
GLint nml=0,day=1;

char name3[]="PROJECT:  Membuat animasi 3d";

GLfloat xt=0.0,yt=0.0,zt=0.0,xw=0.0;
GLfloat tx=295,ty=62;
GLfloat xs=1.0,ys=1.0,zs=1.0;

GLfloat xangle=0.0,yangle=0.0,zangle=0.0,angle=0.0;

GLfloat r=0,g=0,b=1;
GLint light=1;
int count=1,flg=1;
```

```
int view=0;
int flag1=0,aflag=1;
int flag2=0,wheelflag=0;
GLUquadricObj *t;

static void SpecialKeyFunc( int Key, int x, int y );

/* Simple  transformation routine */
GLvoid Transform(GLfloat Width, GLfloat Height)
{
  glViewport(0, 0, Width, Height);
  glMatrixMode(GL_PROJECTION);                /* Select the projection matrix */

  glLoadIdentity();    /* Reset The Projection Matrix */
  gluPerspective(45.0,Width/Height,0.1,100.0);  /* Calculate The Aspect Ratio Of The Window
*/
  glMatrixMode(GL_MODELVIEW);                 /* Switch back to the modelview matrix */
}


/* A general OpenGL initialization function.  Sets all of the initial parameters. */
GLvoid InitGL(GLfloat Width, GLfloat Height)
{

  glClearColor(1.0, 1.0, 1.0, 1.0);
  glLineWidth(2.0);             /* Add line width,   ditto */
  Transform( Width, Height ); /* Perform the transformation */
  //newly added
  t=gluNewQuadric();
```

```
        gluQuadricDrawStyle(t, GLU_FILL);

glEnable(GL_LIGHTING);

glEnable(GL_LIGHT0);

// Create light components
GLfloat ambientLight[] = { 0.2f, 0.2f, 0.2f, 1.0f };
GLfloat diffuseLight[] = { 0.8f, 0.8f, 0.8, 1.0f };
GLfloat specularLight[] = { 0.5f, 0.5f, 0.5f, 1.0f };
GLfloat position[] = { 1.5f, 1.0f, 4.0f, 1.0f };

// Assign created components to GL_LIGHT0
glLightfv(GL_LIGHT0, GL_AMBIENT, ambientLight);
glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuseLight);
glLightfv(GL_LIGHT0, GL_SPECULAR, specularLight);
glLightfv(GL_LIGHT0, GL_POSITION, position);

}


GLvoid ReSizeGLScene(GLint Width, GLint Height)
{
  if (Height==0)     Height=1;                /* Sanity checks */
  if (Width==0)      Width=1;
  Transform( Width, Height );                 /* Perform the transformation */
}
```

```
void init()
{
   glClearColor(0,0,0,0);
 glPointSize(5.0);
 glMatrixMode(GL_PROJECTION);
 glLoadIdentity();
 glOrtho(0.0,900.0,0.0,600.0,50.0,-50.0);
 glutPostRedisplay();
}


void display_string(int x, int y, char *string, int font)
{
   int len,i;
 glColor3f(0.8,0.52,1.0);
 glRasterPos2f(x, y);
   len = (int) strlen(string);
   for (i = 0; i < len; i++) {
   if(font==1)
 glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,string[i]);
 if(font==2)
   glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,string[i]);
 if(font==3)
       glutBitmapCharacter(GLUT_BITMAP_HELVETICA_12,string[i]);
 if(font==4)
   glutBitmapCharacter(GLUT_BITMAP_HELVETICA_10,string[i]);
 }
```

```
}

void display1(void)
{

 glClearColor(1.0,1.0,0.1,1.0);
 display_string(180,540,"Universitas Negeri Surabaya",1); //correct cordinate according to name
 display_string(215,500,name3,1);
 display_string(390,470,"HELP",2);
 display_string(10,450,"MOUSE",2);
 display_string(10,410,"KLIK TOMBOL KANAN UNTUK MENU",3);
 display_string(10,370,"KEYBOARD",2);
 display_string(10,340,"X-Y-Z TOMBOL UNTUK ROTASI ",3);
 display_string(10,310,"A-S-Q UNTUK MENGATUR UKURAN MOBIL",3);
 display_string(10,280,"U-F UNTUK SETING VIEW KAMERA",3);
 display_string(10,250,"GUNAKAN PANAH KIRI(<-) DAN KANAN(->) UNTUK
MENJALANANKAN MOBIL",3);
 display_string(10,220,"ESC UNTUK KELUAR",3);
 display_string(250,150,"TEKAN SPACE BAR UNTUK MASUK",2);
 glutPostRedisplay();
 glutSwapBuffers();

}

GLvoid DrawGLScene()
{
```

```
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); /* Clear The Screen And The Depth Buffer */
if(view==0)
{
init();
display1();
}
else
{
  if(count==1)
 InitGL(Xsize,Ysize);
  if(aflag==1)/* Initialize our window. */
  glClearColor(1,1,1,1);
  else
   glClearColor(0.1,0.1,0.1,0);
  glPushMatrix();
  glLoadIdentity();
  glTranslatef(-1.0,0.0,-3.5);
  glRotatef(xangle,1.0,0.0,0.0);
  glRotatef(yangle,0.0,1.0,0.0);
  glRotatef(zangle,0.0,0.0,1.0);
  glTranslatef(xt,yt,zt);
  glScalef(xs,ys,zs);
  glEnable(GL_COLOR_MATERIAL);
  glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);

  if(flag2==1)
  {
  GLfloat fogcolour[4]={1.0,1.0,1.0,1.0};
```

```
  glFogfv(GL_FOG_COLOR,fogcolour);            /* Define the fog colour */
  glFogf(GL_FOG_DENSITY,0.1);                 /* How dense */

  glFogi(GL_FOG_MODE,GL_EXP);                 /* exponential decay */
  glFogf(GL_FOG_START,3.0);                 /* Where wwe start fogging */
  glFogf(GL_FOG_END,100.0);                   /* end */
  glHint(GL_FOG_HINT, GL_FASTEST);            /* compute per vertex */
  glEnable(GL_FOG);/* ENABLE */
  }
  if(flag2==0)
  {
   glDisable(GL_FOG);
  }

if(!aflag){
  glBegin(GL_POINTS);
  glColor3f(1,1,1);
  glPointSize(200.0);
  int ccount=0;
  float x=10,y=10;
  while(ccount<20)
  {
   glVertex2f(x,y);
   x+=10;
   y+=10;
   if(y>Ysize) y-=10;
   if(x>Xsize) x-=10;
```

```
 ccount++;
 }
 glEnd();}

 glColor3f(1.0,.75,0.0);
 glPointSize(30.0);
 glBegin(GL_POINTS);
 glVertex3f(0.2,0.3,0.3);
 glVertex3f(0.2,0.3,0.5);
 glEnd();
 glPointSize(200.0);



 glBegin(GL_QUADS);              /* OBJECT MODULE*/

 /* top of cube*/
 //************************FRONT
BODY*****************************************
 glColor3f(r,g,b);
 glVertex3f( 0.2, 0.4,0.6);
 glVertex3f(0.6, 0.5,0.6);
 glVertex3f(0.6, 0.5,0.2);
 glVertex3f( 0.2,0.4,0.2);

 /* bottom of cube*/
 glVertex3f( 0.2,0.2,0.6);
 glVertex3f(0.6,0.2,0.6);
 glVertex3f(0.6,0.2,0.2);
```

```
  glVertex3f( 0.2,0.2,0.2);

  /* front of cube*/
  glVertex3f( 0.2,0.2,0.6);
  glVertex3f(0.2, 0.4,0.6);
  glVertex3f(0.2,0.4,0.2);
  glVertex3f( 0.2,0.2,0.2);

  /* back of cube.*/
  glVertex3f(0.6,0.2,0.6);
  glVertex3f(0.6,0.5,0.6);
  glVertex3f(0.6,0.5,0.2);
  glVertex3f( 0.6,0.2,0.2);

  /* left of cube*/
  glVertex3f(0.2,0.2,0.6);
  glVertex3f(0.6,0.2,0.6);
  glVertex3f(0.6,0.5,0.6);
  glVertex3f(0.2,0.4,0.6);

  /* Right of cube */
  glVertex3f(0.2,0.2,0.2);
  glVertex3f( 0.6,0.2,0.2);
  glVertex3f( 0.6,0.5,0.2);
  glVertex3f( 0.2,0.4,0.2);
//*************************************************************************
  glVertex3f(0.7,0.65,0.6);
  glVertex3f(0.7,0.65,0.2);
```

```
  glVertex3f(1.7,0.65,0.2);        //top cover
  glVertex3f(1.7,0.65,0.6);
//*****************************back guard*******************************
  glColor3f(r,g,b);            /* Set The Color To Blue*/
  glVertex3f( 1.8, 0.5,0.6);
  glVertex3f(1.8, 0.5,0.2);
  glVertex3f(2.1, 0.4, 0.2);
  glVertex3f(2.1,0.4,0.6);

  /* bottom of cube*/
  glVertex3f( 2.1,0.2,0.6);
  glVertex3f(2.1,0.2,0.2);
  glVertex3f(1.8,0.2,0.6);
  glVertex3f( 1.8,0.2,0.6);

  /* back of cube.*/
  glVertex3f(2.1,0.4,0.6);
  glVertex3f(2.1,0.4,0.2);
  glVertex3f(2.1,0.2,0.2);
  glVertex3f(2.1,0.2,0.6);

  /* left of cube*/
  glVertex3f(1.8,0.2,0.2);
  glVertex3f(1.8,0.5,0.2);
  glVertex3f(2.1,0.4,0.2);
  glVertex3f(2.1,0.2,0.2);

  /* Right of cube */
```

```
  glVertex3f(1.8,0.2,0.6);
  glVertex3f(1.8,0.5,0.6);
  glVertex3f(2.1,0.4,0.6);
  glVertex3f(2.1,0.2,0.6);
//******************MIDDLE BODY************************************
  glVertex3f( 0.6, 0.5,0.6);
  glVertex3f(0.6, 0.2,0.6);
  glVertex3f(1.8, 0.2, 0.6);
  glVertex3f(1.8,0.5,0.6);

  /* bottom of cube*/
  glVertex3f( 0.6,0.2,0.6);
  glVertex3f(0.6,0.2,0.2);
  glVertex3f(1.8,0.2,0.2);
  glVertex3f( 1.8,0.2,0.6);

  /* back of cube.*/
  glVertex3f(0.6,0.5,0.2);
  glVertex3f(0.6,0.2,0.2);
  glVertex3f(1.8,0.2,0.2);
  glVertex3f(1.8,0.5,0.2);
//**********************ENTER WINDOW**********************************
  glColor3f(0.3,0.3,0.3);
  glVertex3f( 0.77, 0.63,0.2);
  glVertex3f(0.75, 0.5,0.2);        //quad front window
  glVertex3f(1.2, 0.5, 0.2);
  glVertex3f( 1.22,0.63,0.2);
```

```
glVertex3f(1.27,0.63,.2);
glVertex3f(1.25,0.5,0.2);       //quad back window
glVertex3f(1.65,0.5,0.2);
glVertex3f(1.67,0.63,0.2);

glColor3f(r,g,b);
glVertex3f(0.7,0.65,0.2);
glVertex3f(0.7,0.5,.2);      //first separation
glVertex3f(0.75,0.5,0.2);
glVertex3f(0.77,0.65,0.2);

glVertex3f(1.2,0.65,0.2);
glVertex3f(1.2,0.5,.2);      //second separation
glVertex3f(1.25,0.5,0.2);
glVertex3f(1.27,0.65,0.2);

glVertex3f(1.65,0.65,0.2);
glVertex3f(1.65,0.5,.2);     //3d separation
glVertex3f(1.7,0.5,0.2);
glVertex3f(1.7,0.65,0.2);

glVertex3f( 0.75, 0.65,0.2);
glVertex3f(0.75, 0.63,0.2);       //line strip
glVertex3f(1.7, 0.63, 0.2);
glVertex3f( 1.7,0.65,0.2);

glVertex3f( 0.75, 0.65,0.6);
glVertex3f(0.75, 0.63,0.6);       //line strip
```

```
glVertex3f(1.7, 0.63, 0.6);
glVertex3f( 1.7,0.65,0.6);

glColor3f(0.3,0.3,0.3);
glVertex3f( 0.77, 0.63,0.6);
glVertex3f(0.75, 0.5,0.6);        //quad front window
glVertex3f(1.2, 0.5, 0.6);
glVertex3f( 1.22,0.63,0.6);

glVertex3f(1.27,0.63,.6);
glVertex3f(1.25,0.5,0.6);        //quad back window
glVertex3f(1.65,0.5,0.6);
glVertex3f(1.67,0.63,0.6);

glColor3f(r,g,b);
glVertex3f(0.7,0.65,0.6);
glVertex3f(0.7,0.5,.6);       //first separation
glVertex3f(0.75,0.5,0.6);
glVertex3f(0.77,0.65,0.6);

glVertex3f(1.2,0.65,0.6);
glVertex3f(1.2,0.5,.6);       //second separation
glVertex3f(1.25,0.5,0.6);
glVertex3f(1.27,0.65,0.6);

glVertex3f(1.65,0.65,0.6);
glVertex3f(1.65,0.5,.6);
glVertex3f(1.7,0.5,0.6);
```

```
 glVertex3f(1.7,0.65,0.6);
 glEnd();


//*************************************************************
 glBegin(GL_QUADS);

 /* top of cube*/
 glColor3f(0.3,0.3,0.3);
 glVertex3f( 0.6, 0.5,0.6);
 glVertex3f(0.6, 0.5,0.2);       //quad front window
 glVertex3f(0.7, 0.65, 0.2);
 glVertex3f( 0.7,0.65,0.6);

 glVertex3f(1.7,0.65,.6);
 glVertex3f(1.7,0.65,0.2);       //quad back window
 glVertex3f(1.8,0.5,0.2);
 glVertex3f(1.8,0.5,0.6);


//*****************************road and surrounding
development************************************
 if(flag1)
 {
glPushMatrix();
glTranslatef(xw,0,0);
 glColor3f(0,1,0);
 glVertex3f(-100,0.1,-100);
 glVertex3f(-100,0.1,0);         //a green surroundings
```

```
glVertex3f(100,0.1,0);
glVertex3f(100,0.1,-100);

glColor3f(0.7,0.7,0.7);
glVertex3f(-100,0.1,0);
glVertex3f(-100,0.1,0.45);        //a long road
glVertex3f(100,0.1,0.45);
glVertex3f(100,0.1,0);

glColor3f(1.0,0.75,0.0);
glVertex3f(-100,0.1,0.45);      //a median
glVertex3f(-100,0.1,0.55);
glVertex3f(100,0.1,0.55);
glVertex3f(100,0.1,0.45);

glColor3f(0.7,0.7,0.7);
glVertex3f(-100,0.1,0.55);
glVertex3f(-100,0.1,1);         //a long road
glVertex3f(100,0.1,1);
glVertex3f(100,0.1,0.55);

glColor3f(0,1,0);
glVertex3f(-100,0.1,1);
glVertex3f(-100,0.1,100);        //a green surroundings
glVertex3f(100,0.1,100);
glVertex3f(100,0.1,1);
  glPopMatrix();
}
```

```
 glEnd();

 if(wheelflag)
 {
  glPushMatrix();
  glTranslatef(xw,0,0);
  glColor3f(0.5,.2,0.3);
  glBegin(GL_QUADS);
  for(i=0;i<200;i+=0.2)
  {
   glVertex3f(-100+i,0,1);
   glVertex3f(-99.9+i,0,1);
   glVertex3f(-99.9+i,0.2,1);
   glVertex3f(-100+i,0.2,1);
   i+=0.5;
  }
  for(i=0;i<200;i+=0.2)
  {
   glVertex3f(-100+i,0,0);
   glVertex3f(-99.9+i,0,0);
   glVertex3f(-99.9+i,0.2,0);
   glVertex3f(-100+i,0.2,0);
   i+=0.5;
  }
  glEnd();
  glPopMatrix();
 }
//*************************************************************************
**********************
```

```
glBegin(GL_TRIANGLES);              /* start drawing the cube.*/

  /* top of cube*/
  glColor3f(0.3,0.3,0.3);
  glVertex3f( 0.6, 0.5,0.6);
  glVertex3f( 0.7,0.65,0.6);       //tri front window
  glVertex3f(0.7,0.5,0.6);

  glVertex3f( 0.6, 0.5,0.2);
  glVertex3f( 0.7,0.65,0.2);       //tri front window
  glVertex3f(0.7,0.5,0.2);

  glVertex3f( 1.7, 0.65,0.2);
  glVertex3f( 1.8,0.5,0.2);       //tri back window
  glVertex3f( 1.7,0.5,0.2);

  glVertex3f( 1.7, 0.65,0.6);
  glVertex3f( 1.8,0.5,0.6);       //tri back window
  glVertex3f(1.7,0.5,0.6);

glEnd();
//************IGNITION SYSTEM**********************************
glPushMatrix();
   glColor3f(0.7,0.7,0.7);
   glTranslatef(1.65,0.2,0.3);
   glRotatef(90.0,0,1,0);
   gluCylinder(t,0.02,0.03,.5,10,10);
glPopMatrix();
```

```
//*********************WHEEL*********************************************

glColor3f(0.7,0.7,0.7);
glPushMatrix();
  glBegin(GL_LINE_STRIP);
   for(theta=0;theta<360;theta=theta+20)
   {
  glVertex3f(0.6,0.2,0.62);

glVertex3f(0.6+(0.08*(cos(((theta+angle)*3.14)/180))),0.2+(0.08*(sin(((theta+angle)*3.14)/180)
)),0.62);
   }
glEnd();


glBegin(GL_LINE_STRIP);
   for(theta=0;theta<360;theta=theta+20)
   {
  glVertex3f(0.6,0.2,0.18);

glVertex3f(0.6+(0.08*(cos(((theta+angle)*3.14)/180))),0.2+(0.08*(sin(((theta+angle)*3.14)/180)
)),0.18);
   }
glEnd();


glBegin(GL_LINE_STRIP);
for(theta=0;theta<360;theta=theta+20)
   {
  glVertex3f(1.7,0.2,0.18);
```

```
glVertex3f(1.7+(0.08*(cos(((theta+angle)*3.14)/180))),0.2+(0.08*(sin(((theta+angle)*3.14)/180)
)),0.18);
   }
glEnd();

glBegin(GL_LINE_STRIP);
for(theta=0;theta<360;theta=theta+20)
   {
  glVertex3f(1.7,0.2,0.62);

glVertex3f(1.7+(0.08*(cos(((theta+angle)*3.14)/180))),0.2+(0.08*(sin(((theta+angle)*3.14)/180)
)),0.62);
   }
glEnd();
  glTranslatef(0.6,0.2,0.6);
  glColor3f(0,0,0);
  glutSolidTorus(0.025,0.07,10,25);

  glTranslatef(0,0,-0.4);
  glutSolidTorus(0.025,0.07,10,25);

  glTranslatef(1.1,0,0);
  glutSolidTorus(0.025,0.07,10,25);

  glTranslatef(0,0,0.4);
  glutSolidTorus(0.025,0.07,10,25);
  glPopMatrix();
//************************************************************
  glPopMatrix();
```

```
  glEnable(GL_DEPTH_TEST);
  glutPostRedisplay();
  glutSwapBuffers();
 }
}


/*  The function called whenever a "normal" key is pressed. */
void NormalKey(GLubyte key, GLint x, GLint y)
{
    switch ( key )    {
     case ESCAPE : printf("escape pressed. exit.\n");
                glutDestroyWindow(window); /* Kill our window */
                exit(0);
                              break;

      case ' ':view=1;
                 DrawGLScene();
                 break;

      case 'x': xangle += 5.0;
                   glutPostRedisplay();
                   break;

       case 'X':xangle -= 5.0;
                    glutPostRedisplay();
                    break;
```

```
 case 'y':
   yangle += 5.0;
   glutPostRedisplay();
   break;

case 'Y':
   yangle -= 5.0;
   glutPostRedisplay();
   break;

case 'z':
   zangle += 5.0;
   glutPostRedisplay();
   break;

case 'Z':
   zangle -= 5.0;
   glutPostRedisplay();
   break;

 case 'u':                    /* Move up */
   yt += 0.2;
   glutPostRedisplay();
   break;

 case 'U':
   yt -= 0.2;                 /* Move down */
   glutPostRedisplay();
```

```
    break;

  case 'f':                    /* Move forward */
    zt += 0.2;
    glutPostRedisplay();
    break;

  case 'F':
    zt -= 0.2;                  /* Move away */
    glutPostRedisplay();
    break;

case 's':zs+=.2;
 glutPostRedisplay();
 break;

case 'S':zs-=0.2;
 glutPostRedisplay();
 break;

case 'a':ys+=.2;
 glutPostRedisplay();
 break;

case 'A':ys-=0.2;
 glutPostRedisplay();
 break;
```

```
  case 'q':xs+=.2;
   glutPostRedisplay();
   break;

  case 'Q':xs-=0.2;
   glutPostRedisplay();
   break;


    default:
 break;
   }

}

static void SpecialKeyFunc( int Key, int x, int y )
{
 switch ( Key ) {
 case GLUT_KEY_RIGHT:
  if(!wheelflag)
  xt += 0.2;
  if(wheelflag)
  {
  angle+=5;
  xw+=0.2;
  }
    glutPostRedisplay();
  break;
```

```
                case GLUT_KEY_LEFT:
  if(!wheelflag)
  xt -= 0.2;
  if(wheelflag)
  {
  angle+=5;
  xw-=0.2;
  }
       glutPostRedisplay();
  break;
 }
}

void myMenu(int id)
{
  if (id==1)
 {
  flag1=0;
  wheelflag=0;
 glutPostRedisplay();

 }
 if(id ==2)
 {
  flag1=1;
  flag2=0;
  wheelflag=0;
```

```
 xangle += 5.0;
 glutPostRedisplay();
}
if(id==3)
{
 flag2=1;
 wheelflag=0;
 xangle += 5.0;
 glutPostRedisplay();
}
if (id==4)
{
  wheelflag=1;
  glutPostRedisplay();
}
if (id==5)
{
            if(day)
             {

 if(light)
 {
  count++;
  glDisable(GL_LIGHTING);
  glDisable(GL_LIGHT0);
  light=0;
 }
 else
```

```
{
 count--;
 light=1;
 glEnable(GL_LIGHTING);
 glEnable(GL_LIGHT0);
}
glutPostRedisplay();
            }
           else
            {

             if(nml==0 && flag2==2)
               {
                 flag2=0;
                 nml=1;
               }
              else
               {
                  flag2=2;
           nml=0;

           aflag=0;
           day=0;

      glClearColor(0.1,0.1,0.1,0);
      GLfloat fogcolour[4]={0.0,0.0,0.0,1.0};

           glFogfv(GL_FOG_COLOR,fogcolour);          /* Define the fog colour */
```

```
            glFogf(GL_FOG_DENSITY,0.5);                /* How dense */
            glFogi(GL_FOG_MODE,GL_EXP);                /* exponential decay */
                  /* end */
           glHint(GL_FOG_HINT, GL_FASTEST);            /* compute per vertex */
           glEnable(GL_FOG);


      glutPostRedisplay();
            }
         }

}


    if(id==12)
{
aflag=1;
     day=1;
glClearColor(1,1,1,1);
glDisable(GL_FOG);
glutPostRedisplay();
}

if(id==13)
{
aflag=0;
     day=0;
flag2=2;
```

```
 glClearColor(0.1,0.1,0.1,0);
 GLfloat fogcolour[4]={0.0,0.0,0.0,1.0};

  glFogfv(GL_FOG_COLOR,fogcolour);          /* Define the fog colour */
  glFogf(GL_FOG_DENSITY,0.5);              /* How dense */
  glFogi(GL_FOG_MODE,GL_EXP);               /* exponential decay */
                    /* end */
  glHint(GL_FOG_HINT, GL_FASTEST);          /* compute per vertex */
  glEnable(GL_FOG);

  glutPostRedisplay();
      }
}

void colorMenu(int id)
{
  if (id==6)
 {
  r=g=0;
  b=1;
 glutPostRedisplay();

 }
 if(id ==7)
 {
  r=0.8;
  b=g=0;
  glutPostRedisplay();
```

```
 }
 if(id==8)
 {
    g=1;
 r=b=0;
 glutPostRedisplay();
 }
 if (id==9)
 {
  r=b=g=0;
  glutPostRedisplay();
 }
 if(id==10)
 {
 b=0;
 r=g=1;
       glutPostRedisplay();
 }
 if(id==11)
 {
 b=r=g=.7;
       glutPostRedisplay();
 }

}

void myreshape(int w,int h)
{
```

```
        glViewport(0,0,w,h);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        if(w<=h)
                glOrtho(-2.0,2.0,-2.0*(GLfloat)h/(GLfloat)w,2.0*(GLfloat)h/(GLfloat)w,-10.0,10.0);
        else
                glOrtho(-2.0*(GLfloat)w/(GLfloat)h,2.0*(GLfloat)w/(GLfloat)h,-2.0,2.0,-10.0,10.0);
        glMatrixMode(GL_MODELVIEW);
        glutPostRedisplay();
}


//*************************** Main
**********************************************************************

int main(int argc, char **argv)
{
/* Initialisation and window creation */

  glutInit(&argc, argv);              /* Initialize GLUT state. */

  glutInitDisplayMode(GLUT_RGBA |      /* RGB and Alpha */
                      GLUT_DOUBLE|     /* double buffer */
                      GLUT_DEPTH);     /* Z buffer (depth) */

  glutInitWindowSize(Xsize,Ysize);     /* set initial window size. */
  glutInitWindowPosition(0,0);         /* upper left corner of the screen. */
  glutCreateWindow("3D CAR ANIMATION"); /* Open a window with a title. */
/* Now register the various callback functions */
```

```
 glutReshapeFunc(myreshape);
 glutDisplayFunc(DrawGLScene);        /* Function to do all our OpenGL drawing. */
 glutReshapeFunc(ReSizeGLScene);
 glutKeyboardFunc(NormalKey);         /*Normal key is pressed */
 glutSpecialFunc( SpecialKeyFunc );
 InitGL(Xsize,Ysize);
 int submenu=glutCreateMenu(colorMenu);
 glutAddMenuEntry("blue", 6);
glutAddMenuEntry("red", 7);
glutAddMenuEntry("green",8);
glutAddMenuEntry("black",9);
glutAddMenuEntry("yellow",10);
glutAddMenuEntry("grey",11);
 glutCreateMenu(myMenu);
glutAddMenuEntry("car model mode", 1);
glutAddMenuEntry("car driving mode", 2);
glutAddMenuEntry("fog effect",3);
glutAddMenuEntry("wheel effect",4);
glutAddMenuEntry("toggle light",5);
glutAddSubMenu("car colors",submenu);
glutAddMenuEntry("daymode",12);
glutAddMenuEntry("Night mode",13);
glutAttachMenu(GLUT_RIGHT_BUTTON);
 /* Now drop into the event loop from which we never return */
 glutMainLoop();                  /* Start Event Processing Engine. */
 return 1;
}
```

DAFTAR PUSTAKA

[1] https://slideplayer.info/slide/12847016/

[2] https://id.scribd.com/doc/199267758/Laporan-Grafika-Komputer-Kelompok-3