**LAPORAN**

**PEMBUATAN OBJEK 3 DIMENSI**

**ILUSTRASI PEMODELAN ORBIT TATA SURYA**

**TUGAS BESAR : GRAFIKA KOMPUTER**

**DOSEN : Hendri Karisma, S.Kom.**

Disusun oleh:

Asep Maryana : 10108909

Hilda Istiqomah : 10109931

Sandi Lesmana Jaya : 10108997

Kelas : IF-17K



**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK DAN ILMU KOMPUTER**

**UNIVERSITAS KOMPUTER INDONESIA**

**2013**

**BAB I**

**PENDAHULUAN**

* 1. **Rumusan Masalah**

Rumusan masalah dalam penyusunan tugas besar grafika computer ini adalah sebagai berikut :

1. Bagaimana penggambaran objek 3D untuk hasil output?
2. Bagaimana mensimulasikan grafis OpenGL yang didukung pemrograman C/C++ untuk membuat animasi perputaran planet dalam orbit tata surya?
3. Bagaimana membuat objek planet dengan dengan memanfaatkan objek-objek yang ada dalam OpenGL?
   1. **Maksud dan Tujuan**

Maksud dari penulisan tugas besar ini adalah membangun aplikasi ilustrasi pemodelan planet dalam orbit tata surya.

Adapun tujuan yang ingin dicapai dalam penulisan tugas besar ini adalah :

1. Dapat menampilkan outputilustrasi pemodelan objek 3 dimensi.
2. Dapat mensimulasikan grafis pada OpenGL yang didukung bahasa pemrograman C/C++ untuk menghasilkan animasi perputaran planet dalam garis orbit tata surya terhadap matahari.
3. Dapat menampilkan sebagian planet yang terdapat pada garis orbit tata surya, sehingga dapat dijadikan media pembelajaran atau edukasi.
   1. **Batasan Masalah**

Dalam pembangunan aplikasi ini, perlu diberikan beberapa batasan masalah agar mempermudah saat melakukan perancangan, batasan-batasan itu antara lain:

1. Perputaran planet terhadap matahari pada garis orbitnya dengan animasi 3D.
2. Objek pada aplikasi ini menampilkan matahari serta 8 (delapan) planet (merkurius, venus, bumi, mars, yupiter, saturnus, uranus dan neptunus) yang berputar mengelilingi matahari, bulan yang berputar mengelilingi bumi dan garis sebagai tempat lintasan dari ke 8 (delapan) planet tersebut.
3. Menggeser sudut pandang aplikasi dapat menggunakan pergeseran mouse.
4. Batasan perangkat yang ada di aplikasi ini adalah :
5. Aplikasi dibangun dengan menggunakan bahasa pemrograman C/C++.
6. Pengembangan aplikasi menggunakan library OpenGL.
7. Perangkat lunak yang digunakan adalah menggunakan Sistem Operasi Windows 7, Netbeans IDE, Library OpenGL.
   1. **Objek Pembangun Yang Digunakan**
8. Point (GL\_POINTS)

Point digunakan untuk menggambar bintang-bintang.

1. Line (GL\_LINE\_LOOP)

Line digunakan untuk menggambar garis orbit.

1. Texture 2D (GL\_TEXTURE\_2D)

Texture 2D digunakan untuk menggambar planet.

* 1. **Teknik Yang Digunakan**

1. Perspective
2. Color
3. Lighting (Diffuse, Ambient dan Specular)
4. Translation, Rotation dan Scalling
5. Mapping Texture

**BAB II**

**PEMBAHASAN**

**2.1 Kontribusi Anggota Kelompok**

Dalam pembuatan tugas besar grafika komputer yang bertema pembuatan objek 3D untuk pemodelan orbit tata surya dibagi dalam beberapa tugas yang masing-masing harus diimplementasikan. Berikut penjelasan dari masing-masing kontribusi dari setiap anggota kelompok yang beranggotakan :

1. Asep Maryana

Adalah selaku ketua kelompok 3 yang mengatur dan menentukan semua konsep yang telah didiskusikan sebelumnya, membuat coding dan konsep awal pemodelan 3D tatasurya mulai dari mendefinisikan gambar sebagai objek yang akan dibentuk dalam texture 3D, menulis coding main.cpp , mf.h untuk proses texture binding, menentukan tugas anggota kelompok supaya lebih terstruktur.

1. Hilda Istiqomah

Tugas yang diberikan dari asep selaku ketua kelompok adalah membuat file draw.h yang berguna sebagai fungsi untuk menggambar objek dasar bola planet. Selain itu Hilda diberikan tugas untuk membuat laporan dari hasil pembuatan tugas besar pembuatan objek 3D pemodelan orbit tatasurya.

1. Sandi Lesmana Jaya

Sandi diberikan tugas untuk berkintribusi membuat file type.h dan bmp.h. selain itu sandi membuat bahan untuk materi yang akan dipresentasikan.

**2.2. Komponen Pembangun Objek**

Komponen pada pembangunan aplikasi pemodelan orbit tatasurya dalam bentuk 3D terhadap matahari ini banyak menggunakan fungsi*gl\_points* digunakan untuk menggambar bintang-bintang. *Gl\_line\_loop* digunakan untuk menggabungkan titik-titik menjadi gambar garis orbit yang nantinya berfungsi sebagai orbit perputaran planet. *Gl\_textture\_2D* digunakan untuk menggambar planet-planet.

**2.3 Cara Pembuatan**

Dalam pembuatan objek 3D pemodelan orbit tatasurya ini menggunakan beberapa teknik , yaitu :

1. *Perspective*

Teknik ini digunakan sebagai proyeksi.

1. *Color*

Teknik ini digunakan untuk memberikan warna pada objek yang akan dibuat.

1. *Lighting (Diffuse, Ambient dan Specular)*

Teknik ini digunakan untuk mengatur instensitas cahaya pada objek 3D, menggunakan teknik pencahayaan *diffuse, ambient dan specular*.

1. *Translation, Rotaion, Scalling*

* *Translation* digunakan untuk menggambar kembali permukaan polygon pada sisi baru dari objek tersebut pada setiap permukaan. Sebuah objek dalam 3 dimensi ditranslasikan dengan mengubah setiap point yang mendifinisikan objek.
* *Rotation* digunakan untuk membangkitkan rotasi pada objek 3D, dengan membuat aksis dari rotasi dan jumlah sudut rotasi.
* *Scalling* digunakan sebagai penskalaan pada objek 3D dengan transformasi mengubah ukuran dan posisi objek relative terhadap koordinat asli.

1. *Mapping Texture*

Teknik pemetaan sebuah tekstur pada pola gambar *wireframe,* dimana *wireframe* yang telah dibuat akan ditampilkan memiliki kulit luar seperti *texture* yang diinginkan. Dalam pemberian tekstur dilakukan beberapa hal yaitu menentukan tekstur gambar, menandai koordinat tekstur pada *vertex* dan menentukan parameter *texture.*

**BAB III**

**IMPLEMENTASI**

* 1. **Source Code**
* **Main.cpp**

#include <GL/glut.h>

#include <windows.h>

#include <mmsystem.h>

#include <math.h>

#include "type.h"

#include "bmp.h"

#include "mf.h"

#include "draw.h"

#define DEG\_TO\_RAD 0.017453

static GLuint texName;

bool g\_bOrbitOn = true;

float g\_fSpeedmodifier = 1.0f;

float g\_fElpasedTime;

double g\_dCurrentTime;

double g\_dLastTime;

static GLdouble viewer[]= { -sin(10\*DEG\_TO\_RAD), 10\*cos(10\*DEG\_TO\_RAD), -25 };

static int year =0 ,day = 0;

float angle=0.0;

TEXTUREIMAGE earthImg;

TEXTUREIMAGE sunImg;

TEXTUREIMAGE moonImg;

TEXTUREIMAGE venusImg;

TEXTUREIMAGE spaceImg;

TEXTUREIMAGE saturnImg;

TEXTUREIMAGE jupiterImg;

TEXTUREIMAGE neptuneImg;

TEXTUREIMAGE uranusImg;

TEXTUREIMAGE marsImg;

TEXTUREIMAGE mercuryImg;

#define NUM\_TEXTURES 20

GLuint textureObjects[NUM\_TEXTURES];

MATERIAL material[NUM\_TEXTURES];

LIGHT light =

{

{0.0, 0.0, 0.0, 1.0},

{1.0, 1.0, 1.0, 1.0},

{1.0, 1.0, 1.0, 1.0},

{0.0, 0.0, 0.0, 1.0}

};

MATERIAL materialSpace =

{

{0.4, 0.4, 0.4, 1.0},

{0.0, 0.0, 0.0, 1.0},

{0.0, 0.0, 0.0, 1.0},

0.0

};

MATERIAL materialSun =

{

{100.0, 100.0, 100.0, 20.0},

{0.8, 0.8, 0.8, 1.0},

{0.0, 0.0, 0.0, 1.0},

0.0

};

MATERIAL materialEarth =

{

{0.1, 0.1, 0.1, 1.0},

{1.0, 1.0, 1.0, 1.0},

{0.0, 0.0, 0.0, 1.0},

100.0

};

MATERIAL materialMoon =

{

{0.1, 0.1, 0.1, 1.0},

{1.0, 1.0, 1.0, 1.0},

{0.0, 0.0, 0.0, 1.0},

0.0

};

MATERIAL materialSaturn =

{

{0.1, 0.1, 0.1, 1.0},

{1.0, 1.0, 1.0, 1.0},

{0.0, 0.0, 0.0, 1.0},

100.0

};

MATERIAL materialJupiter =

{

{0.1, 0.1, 0.1, 1.0},

{1.0, 1.0, 1.0, 1.0},

{0.0, 0.0, 0.0, 1.0},

100.0

};

MATERIAL materialNeptune =

{

{0.1, 0.1, 0.1, 1.0},

{1.0, 1.0, 1.0, 1.0},

{0.0, 0.0, 0.0, 1.0},

100.0

};

MATERIAL materialMars =

{

{0.1, 0.1, 0.1, 1.0},

{1.0, 1.0, 1.0, 1.0},

{0.0, 0.0, 0.0, 1.0},

100.0

};

MATERIAL materialMercury =

{

{0.1, 0.1, 0.1, 1.0},

{1.0, 1.0, 1.0, 1.0},

{0.0, 0.0, 0.0, 1.0},

100.0

};

MATERIAL materialUranus =

{

{0.1, 0.1, 0.1, 1.0},

{1.0, 1.0, 1.0, 1.0},

{0.0, 0.0, 0.0, 1.0},

100.0

};

void init()

{

glClearColor(0.0,0.0,0.0,0.0);

glEnable(GL\_DEPTH\_TEST);

glShadeModel(GL\_FLAT);

material[0]=materialSun;

material[1]=materialEarth ;

material[2]=materialMoon ;

material[3]=materialSpace ;

material[4]=materialSaturn;

material[5]=materialJupiter;

material[6]=materialNeptune;

material[7]=materialUranus;

material[8]=materialMars;

material[9]=materialMercury;

GLfloat light\_ambient[]={1.0, 1.0, 1.0, 1.0};

GLfloat light\_diffuse[]={1.0, 1.0, 1.0, 1.0};

GLfloat light\_specular[]={1.0, 1.0, 1.0, 1.0};

glLightfv(GL\_LIGHT0, GL\_AMBIENT, light\_ambient);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light\_diffuse);

glLightfv(GL\_LIGHT0, GL\_SPECULAR, light\_specular);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

}

void reshape(int w,int h)

{

glViewport(0,0,(GLint) w, (GLint) h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(60.0,(GLfloat)w/(GLfloat)h,1.0,200.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

gluLookAt(viewer[0],viewer[1],viewer[2],0,0,0,0,0,1);

}

void ShutdownRC(void)

{

glDeleteTextures(NUM\_TEXTURES, textureObjects);

}

void display(void)

{

glLoadIdentity();

gluLookAt(viewer[0],viewer[1],viewer[2],0,0,0,0,0,1);

g\_dCurrentTime = timeGetTime();

g\_fElpasedTime = (float)((g\_dCurrentTime - g\_dLastTime) \* 0.001);

g\_dLastTime = g\_dCurrentTime;

GLfloat light\_position[] = { 0.0, 0.0, 0.0, 1.0 };

static float fSunSpin = 0.0f;

static float fMercurySpin = 0.0f;

static float fMercuryOrbit = 0.0f;

static float fVenusSpin = 0.0f;

static float fVenusOrbit = 0.0f;

static float fEarthSpin = 0.0f;

static float fEarthOrbit = 0.0f;

static float fMoonSpin = 0.0f;

static float fMoonOrbit = 0.0f;

static float fMarsSpin = 0.0f;

static float fMarsOrbit = 0.0f;

static float fJupiterSpin = 0.0f;

static float fJupiterOrbit = 0.0f;

static float fNeptuneSpin = 0.0f;

static float fNeptuneOrbit = 0.0f;

static float fUranusSpin = 0.0f;

static float fUranusOrbit = 0.0f;

static float fSaturnSpin = 0.0f;

static float fSaturnOrbit = 0.0f;

float spaceWidth =18.50;

float spacedepth =8.0;

static float MercuryToSun = -3.0;

static float VenToSun = -5.0;

static float earthToSun = -7.0;

static float moonToEarth = -0.9f;

static float MarsToSun = -9.0;

static float JupiterToSun = -11.0;

static float NeptuneToSun = -14.0;

static float SaturnToSun = -12.0;

static float UranusToSun = -15.0;

if( g\_bOrbitOn == true )

{

fSunSpin -= g\_fSpeedmodifier \* (g\_fElpasedTime \* 10.0f);

fEarthSpin -= g\_fSpeedmodifier \* (g\_fElpasedTime \* 100.0f);

fEarthOrbit -= g\_fSpeedmodifier \* (g\_fElpasedTime \* 20.5f);

fMoonSpin -= g\_fSpeedmodifier \* (g\_fElpasedTime \* 50.0f);

fMoonOrbit -= g\_fSpeedmodifier \* (g\_fElpasedTime \* 200.0f);

fVenusSpin -= g\_fSpeedmodifier \* (g\_fElpasedTime \* 10.0f);

fVenusOrbit -= g\_fSpeedmodifier \* (g\_fElpasedTime \* 20.8f);

fSaturnSpin -= g\_fSpeedmodifier \* (g\_fElpasedTime \* 10.0f);

fSaturnOrbit -= g\_fSpeedmodifier \* (g\_fElpasedTime \* 20.2f);

fJupiterSpin -= g\_fSpeedmodifier \* (g\_fElpasedTime \* 10.0f);

fJupiterOrbit -= g\_fSpeedmodifier \* (g\_fElpasedTime \* 20.0f);

fNeptuneSpin -= g\_fSpeedmodifier \* (g\_fElpasedTime \* 10.0f);

fNeptuneOrbit -= g\_fSpeedmodifier \* (g\_fElpasedTime \* 19.8f);

fUranusSpin -= g\_fSpeedmodifier \* (g\_fElpasedTime \* 10.0f);

fUranusOrbit -= g\_fSpeedmodifier \* (g\_fElpasedTime \* 19.5f);

fMarsSpin -= g\_fSpeedmodifier \* (g\_fElpasedTime \* 10.0f);

fMarsOrbit -= g\_fSpeedmodifier \* (g\_fElpasedTime \* 19.2f);

fMercurySpin -= g\_fSpeedmodifier \* (g\_fElpasedTime \* 10.0f);

fMercuryOrbit -= g\_fSpeedmodifier \* (g\_fElpasedTime \* 19.0f);

}

glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT);

glMatrixMode( GL\_MODELVIEW );

// Set up texture maps

glEnable(GL\_TEXTURE\_2D);

glTexEnvi(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_MODULATE);

glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position);

glMaterialfv(GL\_FRONT, GL\_SPECULAR,material[3].specular);

glMaterialfv(GL\_FRONT, GL\_AMBIENT, material[3].ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, material[3].diffuse);

glMaterialf(GL\_FRONT, GL\_SHININESS, material[3].shininess);

glPushMatrix();

mf(spaceImg,& textureObjects[0]);

glRotatef(90,1,1.0,0.0);

gltDrawSphere(40,100,100);

glPopMatrix();

glDeleteTextures(1, &textureObjects[0]);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* matahari \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

glMaterialfv(GL\_FRONT, GL\_SPECULAR,material[0].specular);

glMaterialfv(GL\_FRONT, GL\_AMBIENT, material[0].ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, material[0].diffuse);

glMaterialf(GL\_FRONT, GL\_SHININESS, material[0].shininess);

glColor3f(1,1,1);

glPushMatrix();

glRotatef(10,1,0.0,0.0);

glRotatef(fSunSpin,0.0,1.0,0.0);

glRotatef(90,1,0.0,0.0);

mf(sunImg,& textureObjects[0]);

gltDrawSphere(2,200,200);

glPopMatrix();

glDeleteTextures(1, &textureObjects[0]);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* akhir matahari \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* bumi \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

glMaterialfv(GL\_FRONT, GL\_SPECULAR,material[1].specular);

glMaterialfv(GL\_FRONT, GL\_AMBIENT, material[1].ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, material[1].diffuse);

glMaterialf(GL\_FRONT, GL\_SHININESS, material[1].shininess);

glPushMatrix();

glRotatef(fEarthOrbit,0.0,1.0,0.0);

glTranslatef(0.0,0.0,earthToSun);

glRotatef(fEarthSpin,0.0,1.0,0.0);

glRotatef(10,0,0.0,1.0);

glRotatef(90,1,0.0,0.0);

mf(earthImg,& textureObjects[0]);

gltDrawSphere(0.6,100,100);

glPopMatrix();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* akhir bumi \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* orbit bumi \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

glBegin(GL\_LINE\_LOOP);

for(angle=0;angle<=360;angle++)

glVertex3f(earthToSun\*sin(DEG\_TO\_RAD\*angle),0,earthToSun\*cos(DEG\_TO\_RAD\*angle));

glEnd();

glDeleteTextures(1, &textureObjects[0]);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* akhir orbit bumi \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* bulan \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

glPushMatrix();

glRotatef( fEarthOrbit, 0.0f, 1.0f, 0.0f );

glTranslatef( 0.0f, 0.0f, earthToSun);

glRotatef( fMoonOrbit, 0.0f, 1.0f, 0.0f );

glTranslatef( 0.0f, 0.0f, moonToEarth );

glRotatef( fMoonSpin, 0.0f, 1.0f, 0.0f );

glColor4f( 1.0f, 1.0f, 1.0f, 1.0f );

glRotatef(90,1,0.0,0.0);

mf(moonImg,& textureObjects[0]);

gltDrawSphere(0.2,50,20);

glPopMatrix();

glDeleteTextures(1, &textureObjects[0]);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* akhir bulan \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* venus \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

glColor3f(192,122,24);

glPushMatrix();

glRotatef(fVenusOrbit,0.0,1.0,0.0);

glTranslatef(0.0,0.0,VenToSun);

glRotatef(fVenusSpin,0.0,1.0,0.0);

glRotatef(90,1,0.0,0.0);

mf(venusImg,& textureObjects[0]);

gltDrawSphere(0.5,150,200);

glPopMatrix();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* akhir venus \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* orbit venus \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

glBegin(GL\_LINE\_LOOP);

for(angle=0;angle<=360;angle++)

glVertex3f(VenToSun\*sin(DEG\_TO\_RAD\*angle),0,VenToSun\*cos(DEG\_TO\_RAD\*angle));

glEnd();

glDeleteTextures(1, &textureObjects[0]);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* akhir orbit venus \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* saturnus \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

glColor3f(192,122,24);

glPushMatrix();

glRotatef(fSaturnOrbit,0.0,1.0,0.0);

glTranslatef(0.0,0.0,SaturnToSun);

glRotatef(fSaturnSpin,0.0,1.0,0.0);

glRotatef(90,1,0.0,0.0);

mf(saturnImg, &textureObjects[0]);

gltDrawSphere(0.5,150,200);

glPopMatrix();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* akhir saturnus \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* orbit saturnus \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

glBegin(GL\_LINE\_LOOP);

for(angle=0;angle<=360;angle++)

glVertex3f(SaturnToSun\*sin(DEG\_TO\_RAD\*angle),0,SaturnToSun\*cos(DEG\_TO\_RAD\*angle));

glEnd();

glDeleteTextures(1, &textureObjects[0]);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* akhir orbit saturnus \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* yupiter \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

glColor3f(192,122,24);

glPushMatrix();

glRotatef(fJupiterOrbit,0.0,1.0,0.0);

glTranslatef(0.0,0.0,JupiterToSun);

glRotatef(fJupiterSpin,0.0,1.0,0.0);

glRotatef(90,1,0.0,0.0);

mf(jupiterImg, &textureObjects[0]);

gltDrawSphere(0.5,150,200);

glPopMatrix();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* akhir yupiter \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* orbit yupiter \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

glBegin(GL\_LINE\_LOOP);

for(angle=0;angle<=360;angle++)

glVertex3f(JupiterToSun\*sin(DEG\_TO\_RAD\*angle),0,JupiterToSun\*cos(DEG\_TO\_RAD\*angle));

glEnd();

glDeleteTextures(1, &textureObjects[0]);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* akhir orbit yupiter \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* neptunus \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

glColor3f(192,122,24);

glPushMatrix();

glRotatef(fNeptuneOrbit,0.0,1.0,0.0);

glTranslatef(0.0,0.0,NeptuneToSun);

glRotatef(fNeptuneSpin,0.0,1.0,0.0);

glRotatef(90,1,0.0,0.0);

mf(neptuneImg, &textureObjects[0]);

gltDrawSphere(0.5,150,200);

glPopMatrix();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* akhir neptunus \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* orbit neptunus \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

glBegin(GL\_LINE\_LOOP);

for(angle=0;angle<=360;angle++)

glVertex3f(NeptuneToSun\*sin(DEG\_TO\_RAD\*angle),0,NeptuneToSun\*cos(DEG\_TO\_RAD\*angle));

glEnd();

glDeleteTextures(1, &textureObjects[0]);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* akhir orbit neptunus \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* uranus \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

glColor3f(192,122,24);

glPushMatrix();

glRotatef(fUranusOrbit,0.0,1.0,0.0);

glTranslatef(0.0,0.0,UranusToSun);

glRotatef(fUranusSpin,0.0,1.0,0.0);

glRotatef(90,1,0.0,0.0);

mf(uranusImg,& textureObjects[0]);

gltDrawSphere(0.5,150,200);

glPopMatrix();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* akhir uranus \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* orbit uranus \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

glBegin(GL\_LINE\_LOOP);

for(angle=0;angle<=360;angle++)

glVertex3f(UranusToSun\*sin(DEG\_TO\_RAD\*angle),0,UranusToSun\*cos(DEG\_TO\_RAD\*angle));

glEnd();

glDeleteTextures(1, &textureObjects[0]);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* akhir orbit uranus \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* mars \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

glColor3f(192,122,24);

glPushMatrix();

glRotatef(fMarsOrbit,0.0,1.0,0.0);

glTranslatef(0.0,0.0,MarsToSun);

glRotatef(fMarsSpin,0.0,1.0,0.0);

glRotatef(90,1,0.0,0.0);

mf(marsImg,& textureObjects[0]);

gltDrawSphere(0.5,150,200);

glPopMatrix();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* akhir mars \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* orbit mars \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

glBegin(GL\_LINE\_LOOP);

for(angle=0;angle<=360;angle++)

glVertex3f(MarsToSun\*sin(DEG\_TO\_RAD\*angle),0,MarsToSun\*cos(DEG\_TO\_RAD\*angle));

glEnd();

glDeleteTextures(1, &textureObjects[0]);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* akhir orbit mars \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* merkurius \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

glColor3f(192,122,24);

glPushMatrix();

glRotatef(fMercuryOrbit,0.0,1.0,0.0);

glTranslatef(0.0,0.0,MercuryToSun);

glRotatef(fMercurySpin,0.0,1.0,0.0);

glRotatef(90,1,0.0,0.0);

mf(mercuryImg,& textureObjects[0]);

gltDrawSphere(0.5,150,200);

glPopMatrix();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* akhir merkurius \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* orbit merkurius \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

glBegin(GL\_LINE\_LOOP);

for(angle=0;angle<=360;angle++)

glVertex3f(MercuryToSun\*sin(DEG\_TO\_RAD\*angle),0,MercuryToSun\*cos(DEG\_TO\_RAD\*angle));

glEnd();

glDeleteTextures(1, &textureObjects[0]);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* akhir orbit merkurius \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

glutSwapBuffers();

glFlush();

}

void keyboard(unsigned char key ,int x,int y)

{

switch(key) {

case'=':

++g\_fSpeedmodifier;

glutPostRedisplay();

break;

case ' ':

g\_bOrbitOn = !g\_bOrbitOn;

glutPostRedisplay();

break;

case'-':

--g\_fSpeedmodifier;

glutPostRedisplay();

break;

case'w':

viewer[0]+= 0.1;

break;

case'e':

viewer[0]-= 0.1;

break;

case's':

viewer[1]+=0.1;

break;

case'd':

viewer[1]-=0.1;

break;

case'x':

viewer[2]+=0.1;

break;

case'c':

viewer[2]-=0.1;

break;

default:

break;

}

display();

}

void mouse(int btn, int state, int x, int y)

{

static int axis=0;

if(btn==GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)

{

axis++;

axis=axis%5;

}

if(btn==GLUT\_RIGHT\_BUTTON && state == GLUT\_DOWN)

{

axis--;

if(axis==-1) axis=4;

}

switch(axis) {

case 0:

viewer[0] = -sin(10\*DEG\_TO\_RAD);

viewer[1] = cos(10\*DEG\_TO\_RAD);

viewer[2] = -15;

break;

case 1:

viewer[0] = 1;

viewer[1] = 20;

viewer[2] = 1;

break;

case 2:

viewer[0] = -sin(10\*DEG\_TO\_RAD);

viewer[1] = 10\*cos(10\*DEG\_TO\_RAD);

viewer[2] = -13;

break;

case 3:

viewer[0] = -5;

viewer[1] = 15;

viewer[2] = -15;

break;

case 4:

viewer[0] = -sin(10\*DEG\_TO\_RAD);

viewer[1] = 10\*cos(10\*DEG\_TO\_RAD);

viewer[2] = -25;

break;

default:

break;

}

display();

}

void mouseMenu(int value)

{

switch(value) {

case 1: // start /stop

g\_bOrbitOn = !g\_bOrbitOn;

glutPostRedisplay();

break;

case 2: // zoom out

viewer[1]-=0.1;

break;

case 3: // zoom in

viewer[1]+=0.1;

break;

case 4: // move right

viewer[0]-= 0.1;

break;

case 5: // move left

viewer[0]+= 0.1;

break;

case 6:

exit(0);

break;

}

}

int main(int argc, char\*\*argv)

{

GLsizei width, height;

width = glutGet(GLUT\_SCREEN\_WIDTH);

height= glutGet(GLUT\_SCREEN\_HEIGHT);

LoadBmp("image//sun.bmp" , &sunImg);

LoadBmp("image//earth.bmp" , &earthImg);

LoadBmp("image//moon.bmp" , &moonImg);

LoadBmp("image//venus.bmp" , &venusImg);

LoadBmp("image//sky.bmp" , &spaceImg);

LoadBmp("image//saturn.bmp" , &saturnImg);

LoadBmp("image//jupiter.bmp" , &jupiterImg);

LoadBmp("image//neptune.bmp" , &neptuneImg);

LoadBmp("image//uranus.bmp" , &uranusImg);

LoadBmp("image//mars.bmp" , &marsImg);

LoadBmp("image//mercury.bmp" , &mercuryImg);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

glutInit(&argc,argv);

glutInitWindowPosition( width / 4, height / 4 );

glutInitWindowSize( width / 2, height / 2 );

glutInitDisplayMode(GLUT\_DOUBLE|GLUT\_RGB);

glutCreateWindow("Tata Surya - IF17K Kelompok 3");

// nambahkan menu jika klik kanan mouse

glutCreateMenu(mouseMenu);

glutAddMenuEntry("Start/Pause", 1);

glutAddMenuEntry("Zoom Out", 2);

glutAddMenuEntry("Zoom In", 3);

glutAddMenuEntry("Move Right", 4);

glutAddMenuEntry("Move Left", 5);

glutAddMenuEntry("Exit", 6);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

init();

glutDisplayFunc(display);

glutIdleFunc(display);

glutReshapeFunc(reshape);

glutKeyboardFunc(keyboard);

glutMouseFunc(mouse);

glutMainLoop();

ShutdownRC();

return 0;

}

* **Type.h**

#ifndef \_TYPE\_H

#define \_TYPE\_H

typedef struct {

int imgWidth;

int imgHeight;

unsigned char \*data;

} TEXTUREIMAGE;

typedef struct {

unsigned short bfType;

unsigned long bfSize;

unsigned short bfReserved1;

unsigned short bfReserved2;

unsigned long bfOffBits;

} BMPFILEHEADER;

typedef struct {

unsigned long biSize;

long biWidth;

long biHeight;

unsigned short biPlanes;

unsigned short biBitCount;

unsigned long biCompression;

unsigned long biSizeImage;

long biXPelsPerMeter;

long biYPelsPerMeter;

unsigned long biClrUsed;

unsigned long biClrImportant;

} BMPINFOHEADER;

typedef struct {

float ambient[4];

float diffuse[4];

float specular[4];

float shininess;

} MATERIAL;

typedef struct {

float lightAmbient[4];

float lightDiffuse[4];

float lightSpecular[4];

float lightPosition[4];

} LIGHT;

#endif /\* \_TYPE\_H \*/

* **Mf.h**

#ifndef \_MF\_H

#define \_MF\_H

#include <GL/glut.h>

#include "type.h"

#include "mf.h"

/\* proc untuk melakukan binding image bmp kedalam object \*/

void mf(TEXTUREIMAGE textureImg, GLuint \* texName)

{

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1);

glGenTextures(1,texName);

glBindTexture(GL\_TEXTURE\_2D , \*texName);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_REPEAT);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_REPEAT);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGB, textureImg.imgWidth,textureImg.imgHeight, 0,

GL\_RGB, GL\_UNSIGNED\_BYTE, textureImg.data);

}

#endif /\* \_MF\_H \*/

* **Draw.h**

#ifndef \_DRAW\_H

#define \_DRAW\_H

#include <stdio.h>

#include <GL/glut.h>

/\* fungsi untuk menggambar objek dasar bola planet, dibuat oleh hilda \*/

void gltDrawSphere(GLfloat fRadius, GLint iSlices, GLint iStacks)

{

//printf("gltDrawSphere - start\n");

GLfloat drho = (GLfloat)(3.141592653589) / (GLfloat) iStacks;

GLfloat dtheta = 2.0f \* (GLfloat)(3.141592653589) / (GLfloat) iSlices;

GLfloat ds = 1.0f / (GLfloat) iSlices;

GLfloat dt = 1.0f / (GLfloat) iStacks;

GLfloat t = 1.0f;

GLfloat s = 0.0f;

GLint i, j;

for (i = 0; i < iStacks; i++)

{

GLfloat rho = (GLfloat)i \* drho;

GLfloat srho = (GLfloat)(sin(rho));

GLfloat crho = (GLfloat)(cos(rho));

GLfloat srhodrho = (GLfloat)(sin(rho + drho));

GLfloat crhodrho = (GLfloat)(cos(rho + drho));

glBegin(GL\_TRIANGLE\_STRIP);

s = 0.0f;

for ( j = 0; j <= iSlices; j++)

{

GLfloat theta = (j == iSlices) ? 0.0f : j \* dtheta;

GLfloat stheta = (GLfloat)(-sin(theta));

GLfloat ctheta = (GLfloat)(cos(theta));

GLfloat x = stheta \* srho;

GLfloat y = ctheta \* srho;

GLfloat z = crho;

glTexCoord2f(s, t);

glNormal3f(x, y, z);

glVertex3f(x \* fRadius, y \* fRadius, z \* fRadius);

x = stheta \* srhodrho;

y = ctheta \* srhodrho;

z = crhodrho;

glTexCoord2f(s, t - dt);

s += ds;

glNormal3f(x, y, z);

glVertex3f(x \* fRadius, y \* fRadius, z \* fRadius);

}

glEnd();

t -= dt;

}

//printf("gltDrawSphere - end\n");

}

#endif /\* \_DRAW\_H \*/

* **Bmp.h**

#ifndef \_BMP\_H

#define \_BMP\_H

#include <stdio.h>

#include <stdlib.h>

#include <memory.h>

#include <string.h>

#include "type.h"

#include "bmp.h"

void LoadBmp(char \*filename, TEXTUREIMAGE \*image)

{

FILE \*file; // file bmp yang akan diload

unsigned long size; // ukuran image dalam bytes

unsigned long i; // standard counter

unsigned short int plane; // number of planes in image

unsigned short int bpp; // jumlah bits per pixel

char temp; // temporary color storage for var warna sementara untuk memastikan filenya ada

if ((file = fopen(filename, "rb")) == NULL) {

printf("File Not Found : %s\n", filename);

return;

}

// mencari file header bmp

fseek(file, 18, SEEK\_CUR);

// read the width

if ((i = fread(&image->imgWidth, 4, 1, file)) != 1) {

printf("Error reading width from %s.\n", filename);

return;

}

// membaca nilai height

if ((i = fread(&image->imgHeight, 4, 1, file)) != 1) {

printf("Error reading height from %s.\n", filename);

return;

}

//printf("Height of %s: %lu\n", filename, image->sizeY)

//menghitung ukuran image(asumsi 24 bits or 3 bytes per pixel)

size = image->imgWidth \* image->imgHeight \* 3;

// read the planes

if ((fread(&plane, 2, 1, file)) != 1) {

printf("Error reading planes from %s.\n", filename);

return;

}

if (plane != 1) {

printf("Planes from %s is not 1: %u\n", filename, plane);

return;

}

// read the bitsperpixel

if ((i = fread(&bpp, 2, 1, file)) != 1) {

printf("Error reading bpp from %s.\n", filename);

return;

}

if (bpp != 24) {

printf("Bpp from %s is not 24: %u\n", filename, bpp);

return;

}

// seek past the rest of the bitmap header.

fseek(file, 24, SEEK\_CUR);

// read the data.

image->data = (unsigned char \*) malloc(size);

if (image->data == NULL) {

printf("Error allocating memory for color-corrected image data");

return;

}

if ((i = fread(image->data, size, 1, file)) != 1) {

printf("Error reading image data from %s.\n", filename);

return;

}

for (i = 0; i < size; i += 3) { // membalikan semuan nilai warna (gbr - > rgb)

temp = image->data[i];

image->data[i] = image->data[i + 2];

image->data[i + 2] = temp;

}

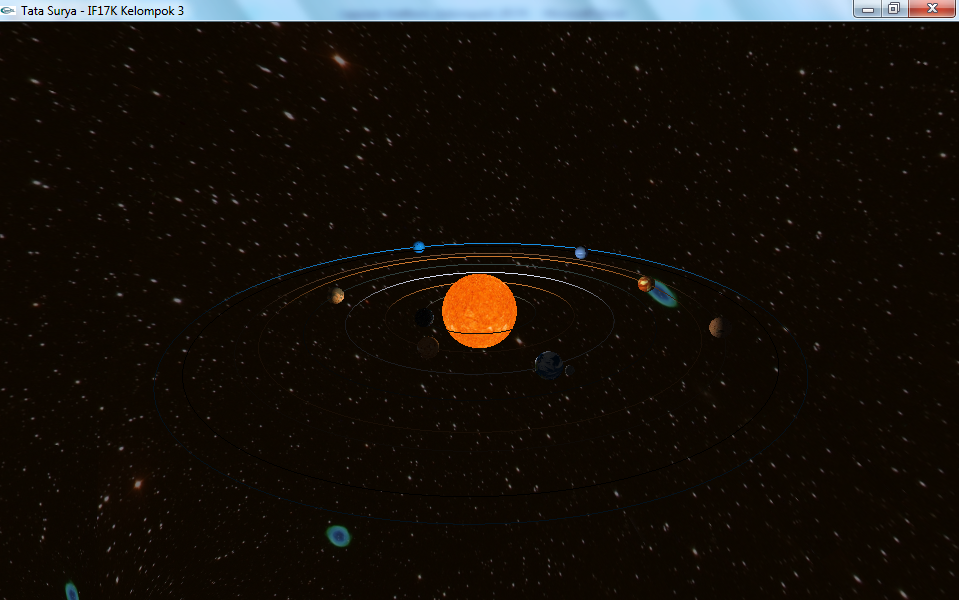
fclose(file);

}

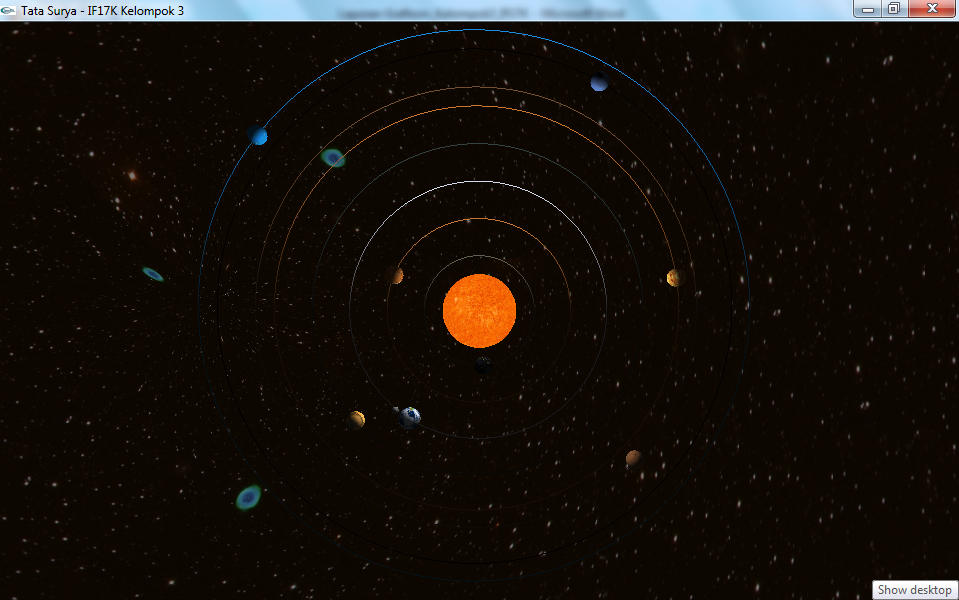
#endif /\* \_BMP\_H \*/

* 1. **Screen Shoot**

1. Tampilan awal aplikasi pemodelan objek 3D orbit tatasurya



1. Tampilan dari sudut pandang yang berbeda ketika fungsi mouse untuk merubah pergeseran sudut pandang digerakan :





* 1. **Fungsi keyboard dan Mouse**

Dalam melakukan rotasi terhadap sudut pandang aplikasi dapat dilakukan menggunakan *Mouse*  untuk menggeser sudut pandang yang diinginkan dari berbagai sisi dengan cara meng-*klik ­*kiri pada mouse sambail menggeserkan mouse ke berbeda arah. Sedangkan fungsi lain yang menggunakan tombol *keyboard* dapat dilakukan sesuai fungsi yang telah ditentukan sebagai berikut :

|  |  |
| --- | --- |
| **Keyboard** | **Fungsi** |
| Tombol Huruf **D** | Untuk membuat tampilan aplikasi dalam posisi *Zoom Out dan* |
| Tombol Huruf **S** | Untuk membuat tampilan aplikasi dalam posisi *Zoom In* |
| Tombol Huruf **E** | Untuk membuat tampilan aplikasi bergeser sudut pandang perputaran kea rah kanan |
| Tombol Huruf **W** | Untuk membuat tampilan aplikasi bergeser sudut pandang perputaran kearah kiri |
| Tombol **Spasi ( *space* )** | Untuk membuat perputaran orbit planet berhenti. Untuk menjalankan perputaran planet kembali dilakukan dengan cara menekan kembali tombol spasi ( *space )* pada *keyboard.* |

**BAB IV**

**KESIMPULAN**

Setelah melakukan perancangan objek 3D yaitu pemodelan Tatasurya, penulis mengambil kesimpulan sebagai berikut :

1. Anggota mengetahui beberapa fungsi-fungsi OpenGL dalam aplikasi 3D.
2. Anggota mengetahui beberapa teknik-teknik 3D dengan beberapa fungsi yang memberikan kesan nyata pada objek yang dibuat.
3. Anggota dapat mengaplikasikan dari materi-materi yang diberikan pada perkuliahan Grafika Komputer sebagai bahan dasar pengetahun mengenai OpenGL dan terutama oembuatan objek 3D ini.