UNIVERZITET U TUZLI
FAKULTET ELEKTROTEHNIKE

ZAVRŠNI PROJEKAT

Računarska grafika

Odsjek: Računarstvo i informatika

Student : Haseljić Eldar
Br. Indexa : 16116

Profesor:
Dr.sc. Skejić Emir
vanr.prof.

Tuzla, juni/lipanj 2019.

Sadržaj

Popis slika

1. Opis problema

NaOpenGL program koji, uz minimalnu upotrebu OpenGL API-ja, zadovoljava sljedeće specifikacije:

• Prikazati Mjesec koji kruži oko Zemlje koja kruži oko Sunca. Mjesec, Zemlja i Sunce trebaju biti predstavljeni kao sfere. (5 bod.)

• Zemlja bi se trebala vrtiti oko Sunca jedanput svakih 365 dana, a Mjesec bi se trebao vrtiti oko Zemlje jednom svakih 30 dana. Zemlja bi se trebala rotirati oko svoje ose jednom dnevno. Dan je interval koji ćete Vi odabrati. (15 bod.)

• Korisnik bi trebao biti u mogućnosti udvostručiti dužinu dana klikom na lijevo dugme miša i prepoloviti dužinu dana klikom na desno dugme miša. Imajte na umu da se to može postići promjenom ugla rotacije svake iteracije. (5 bod.)

• Koristiti barem dvije različite non-default specifikacije koordinata kamere. (Kontrolu "situacije" možete vršiti pomoću ulaza s tastature za odabir postavki.) (5 bod.)

• Pomoću tastature ili menija program bi trebao omogućiti ili onemogućiti osvjetljenje. Kada je osvjetljenje onemogućeno, Zemlja, Mjesec i Sunce trebaju biti tretirani kao plava, siva i sivobijela puna sfera. Kada je osvjetljenje omogućeno, Sunce treba biti tretirano kao sivobijeli difuzni izvor svjetlosti s bijelom spekularnom komponentom, dok Zemlja i Mjesec trebaju biti tretirani kao pune sfere. Svojstva materijala treba prilagoditi tako da Zemlja izgleda plavo-zelena s bijelim blještavilima, a Mjesec siv s bijelim blještavilima. (10 bod.)

• Konačno, na Zemlju i Mjesec primijeniti proizvoljno odabrane mape tekstura koje će njihov izgled učiniti realističnim. (10 bod.)

2. Rjesenje problema

2.1. Postavke programskog alata

Za ovaj projekat je koristen program Code::Blocks u kojem su prethodno postavljene postavke za kreiranje GLUT projekata tj.

1. Ubaciti fajl glut32.dll u C:\Windows\system
2. Ubaciti fajl glut.h u C:\Program Files (x86)\CodeBlocks\MinGW\include\GL
3. Ubaciti fajl glut32.lib u C:\Program Files (x86)\CodeBlocks\MinGW\lib
4. U svakom novom kreiranom projektu potrebno je postaviti putanju do glut datoteka na C:\Program Files (x86)\CodeBlocks\MinGW
5. U svaki main.cpp\main.c potrebno je staviti #include <windows.h>

Ukoliko je upitanju Linux OS potrebno je uraditi sljedece komande u terminalu

$ sudo apt-get update
$ sudo apt-get install libglu1-mesa-dev freeglut3-dev mesa-common-dev

Prilikom kompajliranja potrebno je navesti biblioteke kao u nastavku

$ g++ main.cpp -o firstOpenGlApp -lglut -lGLU –lGL (ako je upitanju .cpp program)
$ gcc main.c -o firstOpenGlApp -lglut -lGLU –lGL (ako je u pitanju .c program)

Ovako se kompajlira samo u slučaju ako se sve funkcije nalaze u istom cpp ili c fajlu, a ako ne onda je potrebno naavesti sve cpp ili c fajlove na mjestu main.cpp\main.c fajla

2.2. Prikaz planeta

Prikaz planeta je izvrsen u fajlu **planets.hpp**, u kojem se nalazi funkcija draw(). Ta funkcija nam poziva tri druge funkcije koje sluze za iscrtavanje Sunca, Zemlje i Mjeseca. Nad planetama smo vršili odredjene translacije i rotacije (glTranslatef(), glRotatef()) da bi ih doveli u odgovarajući položaj.Funkcija koja je korištena za iscrtavanje sfera je glSolidSphere(), dok je prilikom upotrebe textura korištena druga funkcija koja će biti navedena u nastavku. Unutar planets.hpp se nalaze varijable kao sto su LightOn i neke druge o kojim će se pričati u nastavku.Unutar ovog fajla imamo i globalne varijable koje ćemo koristiti u cijelom projektu.

/* planets.hpp */

```
#ifndef PLANETS_HPP
#define PLANETS_HPP value
#include <math.h>
#include "texture.hpp"

float  ScreenWidth;
float  ScreenHeight;

float HourOfDay  = -1.0;
float DayOfYear  =  0.0;
float DayOnSun   =  0.0;
float Increment  =  1.0;
int   printBool  =  1;
int   ViewBool   =  1;
int   lightBool  =  0;
int   texBool    =  0;
float x = 0.0;
float z = -10.0;

//Funkcija za crtanje Sunca
void drawSun ()
{
        //Instrukcije koje sluze prilikom postavke svjetlosti
        if(lightBool == 1)
        {
                GLfloat diffuseLight  [] = { 0.658824 , 0.658824 , 0.658824 , 1.0 };
                GLfloat specularLight [] = { 1.0 , 1.0 ,  1.0 , 1.0 };
                GLfloat position      [] = { 0.0 , 0.0 , z , 1.0 };
                GLfloat position2     [] = { 0.0 , 0.0 , z/2 , 1.0 };
```

```
                glLightfv   ( GL_LIGHT1 , GL_POSITION  , position     );
                glLightfv   ( GL_LIGHT1 , GL_DIFFUSE   , diffuseLight  );
                glLightfv   ( GL_LIGHT1 , GL_SPECULAR  , specularLight );
                glLightfv   ( GL_LIGHT0 , GL_POSITION  , position2    );
    if(texBool == 0)
      glMaterialfv ( GL_FRONT  , GL_SPECULAR  , specularLight );

                glMaterialf  ( GL_FRONT  , GL_SHININESS , 150.0 );
                glColor3f    ( 1.0 , 1.0 , 0.0 );
        }
        else
        {
                glColor3f ( 0.658824  , 0.658824 ,  0.658824 );
        }


        //Postavke pozicije sunca i instrukcija za crtanje sunca
        glTranslatef    ( 0.0  , 0.0 , z );
        glRotatef       ( 20.0 , 1.0 ,  0.0 , 0.0 );
        glPushMatrix ();

        //Postavke rotacije sunca oko svoje ose
        glRotatef ( 360.0 * DayOnSun / 240.0, 0.0 , 1.0 , 0.0);

        if(texBool == 1)
        {
           glEnable(GL_TEXTURE_2D);
    tex(1);
        }
        else
  {
    glDisable(GL_TEXTURE_2D);
    glutSolidSphere ( 1.8  , 120  ,  60  );
  }
  glPopMatrix();
        glPushMatrix    ();
}

//Funkcija za crtanje Zemlje
void drawEarth ()
{
        //Instrukcije koje sluze prilikom postavke svjetlosti
        if(lightBool == 1)
        {
                glDisable (GL_LIGHT0);
                glColor3f ( 0.0 , 0.38 , 0.478);
        }
        else
        {
                glColor3f ( 0.0 , 0.0 , 1.0 );
        }

        //Postavke rotacije zemlje oko sunca

        glRotatef    ( 360.0 * DayOfYear / 365.0  , 0.0 , 1.0 , 0.0 );
```

```
        glTranslatef ( 5.0 , 0.0 , 0.0 );
        glPushMatrix ();

        //Postavke rotacije zemlje oko svoje ose
        glRotatef ( 360.0 * HourOfDay / 24.0, 0.0 , 1.0 , 0.0);

        //Crtamo Zemlju
        if(texBool == 1)
        {
           glRotatef(-90.0,1.0,0.0,0.0);
     glEnable(GL_TEXTURE_2D);
     tex(2);
        }
        else
   {
     glDisable(GL_TEXTURE_2D);
     glutSolidSphere ( 0.6 , 60  , 30);
   }
        glPopMatrix     ();
}

//Funkcija za crtanje mjeseca
void drawMoon ()
{
        //Instrukcije koje sluze prilikom postavke svjetlosti
        if(lightBool == 1)
        {
                glDisable ( GL_LIGHT0 );
                glColor3f ( 0.658824  , 0.658824 ,  0.658824 );
        }
        else
        {
                glColor3f    ( 0.2 , 0.4 , 0.4 );
        }

        //Postavke rotacije mjeseca oko zemlje
        glRotatef    ( 360.0 * 12.0 * DayOfYear / 365.0  , 0.0 , 1.0 , 1.0);
        glTranslatef ( 1.0 , 0.0 , 0.0);

        //Crtamo mjesec
        if(texBool == 1)
        {
     glEnable(GL_TEXTURE_2D);
     tex(3);
        }
        else
   {
     glDisable(GL_TEXTURE_2D);
     glutSolidSphere ( 0.18 , 30 , 15);
   }
        glPopMatrix ();
}

//Funkcija za crtanje elemenata
void draw ()
```

```
{
        //Instrukcije koje sluze prilikom postavke svjetlosti
        if(lightBool == 1)
        {
                glShadeModel ( GL_SMOOTH );
                glEnable ( GL_LIGHT1 );
                glEnable ( GL_LIGHT0 );
                glEnable ( GL_LIGHTING );
                if(texBool!=1)
       glEnable ( GL_COLOR_MATERIAL );
        }
        else
        {
                glDisable ( GL_LIGHT0 );
                glDisable ( GL_LIGHT1 );
                glDisable ( GL_LIGHTING );
                glDisable ( GL_COLOR_MATERIAL );
        }

        //Pozivamo funkcije koje sluze za crtanje sunca, zemlje i mjeseca
   drawSun   ();
   drawEarth ();
   drawMoon  ();
        glutSwapBuffers ();
}
#endif // PLANETS_HPP
```

2.3. Animacija

Animacija tj kretanje Zemlje oko sunca, mjeseca oko Zemlje , te okretanje Zemlje oko svoje ose se vrši unutar fajla **animate.hpp**. Unutar tog fajla se nalazi funkcija koja se naziva idle () u kojoj se vrši animacija našeg sistema. Ona vrši inkrementiranje varijable HourOfDay za default – nu vrijednost varijable Increment koja je postavljena na 1.0 tj povećava stanje sata za vrijednost inkrementa. Tu imamo uslov koji je postavljen tako da kada je HourOfDay >= 24 da se on sam vrati na vrijednost 0.0. Pored te varijable imamo I varijablu DayOfYear koja predstavlja dan u godini ona se povecava za Increment / 24 zato sto prati vrijeme tj ta varijabla nam je potrebna prilikom rotacije Zemlje oko sunca te mjeseca oko Zemlje .Uz pomoc idle() funkcije te funkcije draw() u kojoj u rotaciji se koriste ove dvije varijable sto se može vidjeti u dole priloženom kodu vrši se sva magija animacije našeg sistema.O promjeni vrijednosti Increment varijable biće više u nastavku.Pored navedenih rotacija u kod je dodan način i da sunce se rotira oko svoje ose sto ujedno daje relalističniji izgled sistemu.

```
/* animate.hpp */

#ifndef ANIMATE_HPP
#define ANIMATE_HPP value

//Funkcija koja nam sluzi za animaciju
void idle ()
{
```

```
        if ( DayOfYear >= 366 )
                DayOfYear = 0.0;

        HourOfDay += Increment;
        DayOfYear += Increment / 24.0;
        DayOnSun  += Increment / 24.0;

        if ( HourOfDay >= 24 )
                HourOfDay = 0.0;

        if ( DayOfYear >= 366.0 )
                DayOfYear = 0.0;
        //printf("%f\n", DayOfYear);
        //printf("%d\n", texBool);
        //printf("%f\n",x);
        glutPostRedisplay();
}
#endif // ANIMATE_HPP
```

2.4. Promjena Increment varijable

Kao što je navedeno gore u tekstu promjna same Increment varijable utiče na brzinu kretanja svih planeta .U samom zadatku se zahtijevalo sljedeće : "*Korisnik bi trebao biti u mogućnosti udvostručiti dužinu dana klikom na lijevo dugme miša i prepoloviti dužinu dana klikom na desno dugme miša".*

Kao što je i bio zahtjev, promjenu brzine kretanja vršimo u funkciji mouse() koja se nalazu u fajlu **IO.hpp**.Korisnik klikom na lijevo dugme miša povećava duplo trenutnu vrijednost Increment varijable sto utiče da se sve rotacije odvijaju brže, klikom na desno dugme miša duplo smanjuje trenutnu vrijednost varijable Increment te se sve rotacije odvijaju sporije.

```
/* IO.hpp */

#ifndef IO_HPP
#define IO_HPP value

//Funkcija za mis
void mouse ( int button, int state , int x , int y )
{
        switch ( button )
        {
                case GLUT_LEFT_BUTTON :
                        Increment *= 2;
                        break;
                case GLUT_RIGHT_BUTTON :
                        Increment /= 2;
                        break;
        }
}
#endif // IO_HPP
```

2.5. Postavke kamere

Što se tiče postavki kamere korištena su tri načina pogleda kao što us glProjection(), glFrustum() glOrtho().Te postavke uključujemo uz pomoć sljedećih tipki na tastaturi:

- 'p' ili 'P' → postavljaju default-ni pogled tj poziva se funkcija resetView() koja postavlja defaultne vrijednosti pozicije kamere
- 'o' ili 'O' → poziva se funkcija Ortho() koja unutar sebe obavi određene postavke i poziva funkciju glOrtho()
- 'f' ili 'F' → poziva se funkcija Frustum() koja unutar sebe obavi određene postavke i poziva funkciju glFrustum()
- '+' → vrši ZOOM IN prema planetama
- '-' → vrši ZOOM OUT prema planetama

Prikaz svih ovih funkcija će biti u nastavku , a sve funkcije se nalaze u fajlu **window.hpp.** Svaka od navedenih opcija se može promijeniti i unapijediti po želji korisnika.

2.6. Svijetlo

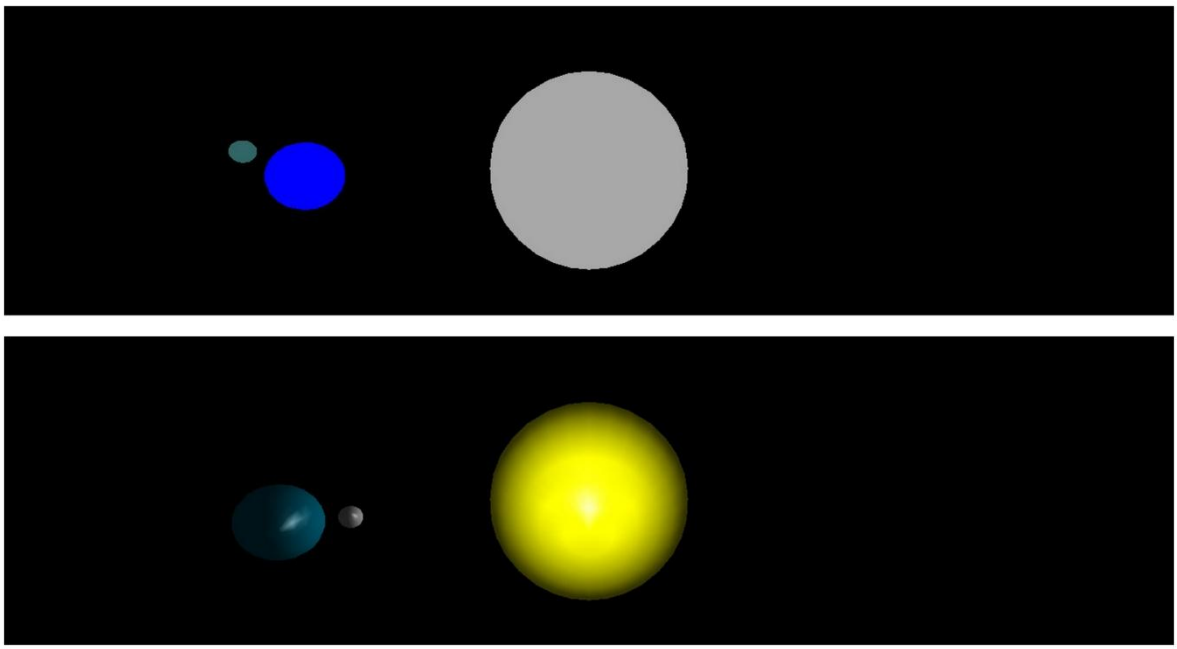
Po zahtjevu samog projekta imamo sljedeće:*″ Pomoću tastature ili menija program bi trebao omogućiti ili onemogućiti osvjetljenje. Kada je osvjetljenje onemogućeno, Zemlja, Mjesec i Sunce trebaju biti tretirani kao plava, siva i sivobijela puna sfera. Kada je osvjetljenje omogućeno, Sunce treba biti tretirano kao sivobijeli difuzni izvor svjetlosti s bijelom spekularnom komponentom, dok Zemlja i Mjesec trebaju biti tretirani kao pune sfere. Svojstva materijala treba prilagoditi tako da Zemlja izgleda plavo-zelena s bijelim blještavilima, a Mjesec siv s bijelim blještavilima.″*.

Sve postavke su vršene uz pomoć tastature koja je definisana kao funkcija kbd() i definisana je u **window.hpp**.Svjetlo je po defaultu isključeno, a može se i ručno isključiti pritiskom na dugme '0' na tastaturi. Kada se stisne dugme '0' varijabla lightBool se postavi na 0 te se izvršava određeni dio koda koji je naveden gore u fajlu **planets.hpp.** Kada je isključeno svjetlo sfrere su ofarbane u sivobijelu , bijelu i plavu boju , uz pomoć funkcije glColor().

Kada je svjetlo uključeno tj kada stisnemo dugme '1' na tastaturi funkcija lightOn() postavlja vrijednost varijable lightBool na 1 te se unutar draw() funkcije pocinje izvršavati drugi dio koda.U ovom slučaju koriste se funkije glLight() i glMaterial() koje služe za postavke svjetlosti i materijala.

Pored njih koristimo glEnable() da bi smo omogućili određene stvari . Također uz pomoć navedene funkcije smo omogućili GL_COLOR_MATERIAL bez kojeg ne bismo mogli dobiti sfere kakve želimo.

Ovdje su korištena dva svjetla ali drugo svjetlo je korišteno samo da bi se dobio relastičniji izgled samog sunca. Sunce,Zemlja i mjesec su prikazani u željenim bojama sa odgovarajućim bješatavilima. Pored navedenih funkcija postoje i funkcija reset() koja resetuje simulaciju i vraća je u početno stanje, a nju pokrećemo klikom na 'R' ili 'r'.U ovom fajlu također se nalazi i glavna funkcija za prikaz tj display(). Ona ispisuje sve komande koje su nam na raspolaganju te poziva funkciju draw(). Tu se nalazi i funkijca reshape() koja služi prilikom promjene veličine prozora.



Slika 1. Prikaz planeta bez upotrebe i sa upotrebom svjetlosti

/*window.hpp*/

```
#ifndef WINDOW_HPP
#define WINDOW_HPP

//Display funkcija
void display()
{
	glClear ( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
	glLoadIdentity ();

	if( printBool )
	{
		printf ( "-------------------------------" );
		printf ( "---------------------------------------\n");
		printf ( "Double the length of the day by" );
		printf ( " clicking the left button on the mouse.\n" );
		printf ( "-------------------------------" );
		printf ( "---------------------------------------\n");
		printf ( "Halve  the length of the day by" );
		printf ( " clicking the left button on the mouse.\n" );
		printf ( "-------------------------------" );
		printf ( "---------------------------------------\n");
		printf ( "Click 'o' or 'O' for Ortho projection.       \n" );
		printf ( "Click 'f' or 'F' for Frustum projection.     \n" );
		printf ( "Click 'p' or 'P' for Perspective projection. \n" );
		printf ( "-------------------------------" );
		printf ( "---------------------------------------\n");
		printf ( "Click '1' to turn ON  the lighting.\n" );
		printf ( "Click '0' to turn OFF the lighting.\n" );
		 printf ( "-------------------------------" );
		printf ( "---------------------------------------\n");
```

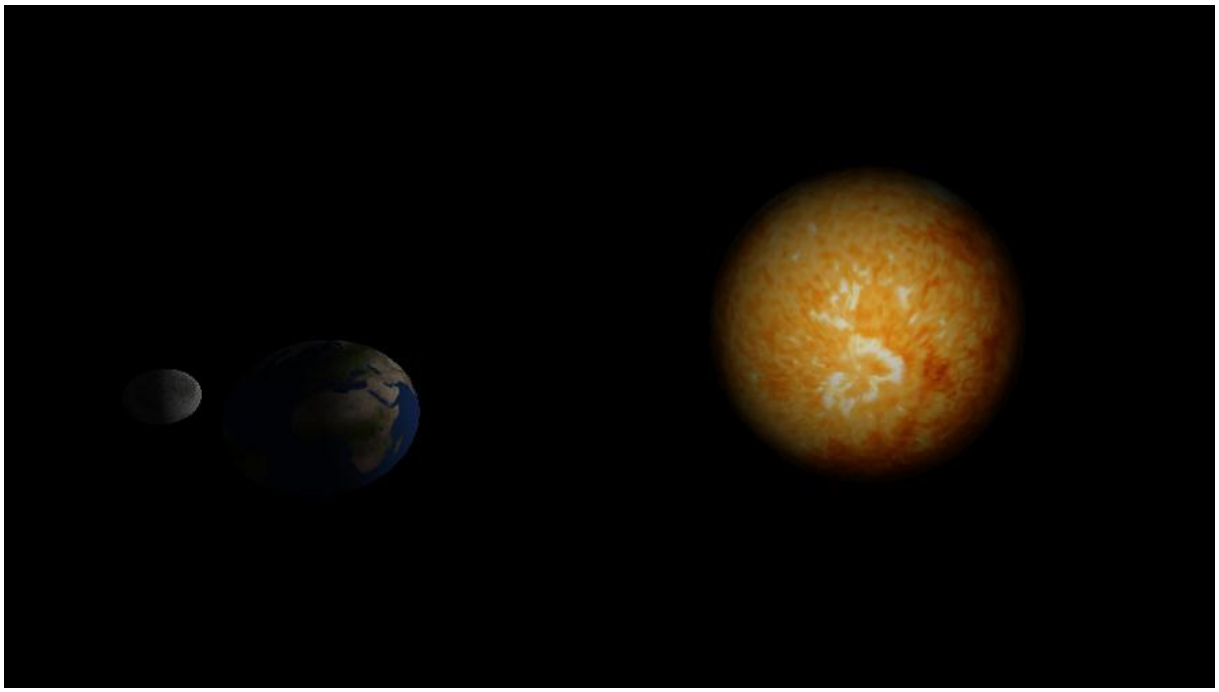
```
                printf ( "Click '+' to ZOOM IN .\n" );
                printf ( "Click '-' to ZOOM OUT.\n" );
                printf ( "-------------------------------" );
                printf ( "---------------------------------------\n" );
                printf ( "Click 'T' to turn ON  the textures.\n" );
                printf ( "Click 't' to turn OFF the textures.\n" );
                printf ( "-------------------------------" );
                printf ( "---------------------------------------\n" );
                printf ( "Click 'r' or 'R' to restart simulation.\n" );
                printf ( "Click 'ESC' to end the simulation.\n");
                printf ( "-------------------------------" );
                printf ( "---------------------------------------\n");

                printBool = 0;
        }
        //Poziv funkcije za crtanje elemenata
        draw ();
}

void init ()
{
        glClearColor ( 0.0 , 0.0 , 0.0 , 0.0 );
        glEnable ( GL_DEPTH_TEST );
}

//Funkcija reshape
void reshape ( int w, int h )
{
        if ( h == 0 ) h = 1;
        if ( w == 0 ) w = 1;
        glViewport( 0 , 0 , w , h);

        ScreenWidth  = w;
        ScreenHeight = h;
        if ( ViewBool )
        {
                glMatrixMode( GL_PROJECTION );
                glLoadIdentity ();
                gluPerspective ( 70.0 ,(float)w/(float)h,  1.0  , 30.0);
                glMatrixMode ( GL_MODELVIEW );
        }
}

//Funkcija za povratak na pocetne vrijednosti
void reset ()
{
        Increment =  1.0;
        ViewBool  =  1;
        lightBool =  0;
        texBool   =  0;
         z = -10.0;
        reshape ( ScreenWidth, ScreenHeight );
}

void Ortho ()
```

```
{
        glMatrixMode   ( GL_PROJECTION );
        glLoadIdentity ();
        glOrtho        ( -6.0 , 8.0 , -6.0 , 8.0 , -100.0 , 100.0 );
        glMatrixMode   ( GL_MODELVIEW );
        ViewBool = 0;
}

void Frustum ()
{
        glMatrixMode   ( GL_PROJECTION );
        glLoadIdentity ();
        glFrustum      ( -3.0 , 4.0 , -4.0 , 2.0 , 2.5 , 100.0 );
        glMatrixMode   ( GL_MODELVIEW );
        ViewBool = 0;
}

void resetView ()
{
        ViewBool =  1;
        reshape ( ScreenWidth , ScreenHeight );
}

void LightOff ()
{
        lightBool = 0;
        texBool = 0;
        glDisable (GL_COLOR_MATERIAL);
}

void LightOn ()
{
        lightBool = 1;
        texBool = 0;
        glEnable ( GL_COLOR_MATERIAL );
}

void TextureOn ()
{
        texBool = 1;
        lightBool = 1;
        reshape ( ScreenWidth , ScreenHeight );
}

void TextureOff ()
{

        if(lightBool == 1 && texBool == 0)
     lightBool = 1;
   else
     lightBool = 0;
        texBool = 0;
        reshape ( ScreenWidth , ScreenHeight );
}
```

```
//Funkcija za tastaturu
void kbd (unsigned char key , int x , int y)
{
	switch( key )
	{
		case 27  :
			 exit ( 0 );
			 break;
		case 'o' :
		case 'O' :
			Ortho ();
			break;
		case 'R' :
		case 'r' :
			reset ();
			break;
		case 'f' :
		case 'F' :
			Frustum ();
			break;
		case 'P' :
		case 'p' :
			resetView ();
			break;
		case '0' :
			LightOff ();
			break;
		case '1' :
			LightOn ();
			break;
		case 'T' :
			TextureOn ();
			break;
		case 't' :
			TextureOff ();
			break;
		case '+' :
			z += 0.1;
			break;
		case '-' :
			z -= 0.1;
			break;
		}
}

#endif // WINDOW_HPP
```

2.7. Texture

Zahtjev ovog projekta je bio da se uz pomoć odogvrajućih tekstura, planete učine realističnim. Texture se nalaze u folderu Textures, a postavljaju se u fajlu **textures.hpp**. Slike se nalaze u bmp formatu pored tog formata u tom folderu imamo i .jpg format ali njega ne koristimo u ovom slučaju.



Slika 2.Prikaz planeta sa uključenom texturom

/*texture.hpp*/

```
#ifndef TEXTURE_HPP
#define TEXTURE_HPP value
#include <fstream>
#include <assert.h>

using namespace std;

struct Image {

        char* pixel_data;
        int width;
        int height;

        Image(char* data, int w, int h)
        {
           pixel_data = data;
           width  = w;
           height = h;
        }

        ~Image()
```

```
        {
            delete [] pixel_data;
        }

};

GLuint Texture;
Image* planet;
GLUquadricObj *ball;

template<class T>
struct auto_array
{
   T* array;
   bool isReleased;

   auto_array(T* array_ = NULL) :
      array(array_), isReleased(false) {}

   ~auto_array()
   {
      if (!isReleased && array != NULL) delete[] array;
   }

   T* release()
   {
      isReleased = true;
      return array;
   }
};

int readInt(ifstream &input)
{
   char buffer[4];
   input.read(buffer, 4);
   return (int)(((unsigned char)buffer[3] << 24) |
      ((unsigned char)buffer[2] << 16) |
      ((unsigned char)buffer[1] << 8) |
      (unsigned char)buffer[0]);
}

Image* loadBMP(const char* filename)
{
        ifstream input;
        input.open(filename, ifstream::binary);
   char buffer[2];
        input.read(buffer, 2);
        input.ignore(8);

        int dataOffset = readInt(input);
        int headerSize = readInt(input);
        int width= readInt(input);
   int height = readInt(input);
   input.ignore(2);
```

```
        int bytesPerRow = ((width * 3 + 3) / 4) * 4 - (width * 3 % 4);
        int size = bytesPerRow * height;

        auto_array<char> pixels(new char[size]);
        input.seekg(dataOffset, ios_base::beg);
        input.read(pixels.array, size);

        auto_array<char> pixels2(new char[width * height * 3]);
        for (int y = 0; y < height; ++y)
   {
                for (int x = 0; x < width; ++x)
                {
                        for (int c = 0; c < 3; ++c)
                        {
                                pixels2.array[3 * (width * y + x) + c] =
     pixels.array[bytesPerRow * y + 3 * x + (2 - c)];
                        }
                }
        }
        input.close();
        return new Image(pixels2.release(), width, height);
}

void tex (int p)
{
   if(p == 1)
      planet =
loadBMP("C:/Users/Haselja/Desktop/eldar_haseljic_zavrsniprojekat/Projekat/Textures/sun.bmp");
   if(p == 2)
      planet =
loadBMP("C:/Users/Haselja/Desktop/eldar_haseljic_zavrsniprojekat/Projekat/Textures/earth.bmp");
   if(p == 3)
      planet =
loadBMP("C:/Users/Haselja/Desktop/eldar_haseljic_zavrsniprojekat/Projekat/Textures/moon.bmp");

   glGenTextures ( 1, &Texture );
   glBindTexture ( GL_TEXTURE_2D, Texture);
   glTexImage2D  ( GL_TEXTURE_2D, 0, GL_RGB,
            planet -> width , planet -> height ,
            0 , GL_RGB , GL_UNSIGNED_BYTE , planet -> pixel_data );
   delete planet;
   glColor3f(1,1,1);
   ball=gluNewQuadric();
   gluQuadricNormals(ball, GLU_SMOOTH);
   glBindTexture(GL_TEXTURE_2D, Texture);
   glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
   glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
   gluQuadricTexture(ball, 1);
   if ( p == 1)
      gluSphere(ball,1.8,120,60);
   if ( p == 2)
      gluSphere(ball,0.6,60,30);
   if ( p == 3)
      gluSphere(ball,0.18,30,15);
   gluDeleteQuadric(ball);
```

```
}

#endif // TEXTURE_HPP
```

U ovom fajlu prvo imamo Strukturu koja se naziva Image, ima tri polja to su pixel_data,height,width, odnosno to su osvnovne karakteristike svake slike. Nakon toga imamo tri globalne varijable koje se koriste za postavljane texture na sferu, moramo koristiti njih posto se na sferu ne može staviti textura ako se koristi gluSolidSphere() funkcija za crtanje sfere.Tako da kada su texture uključene texBool se postavi na 1 te se u planets.hpp sfera crta uz pomoć gluSphere().Unutar texture.hpp imamo i pomoćnu strukturu niza koja nam služi za snimanje sadržaja svakog pixela slike.

Pored te strukture tu je funkcija loadBMP() koja predstavlja funkciju za učitavanje .bmp fajla. T u je i famozna funkcija tex () koja sadži proces dodavanja texture na sferu.
NAPOMENA: Potrebno je dodati ogovarajuću putanju do svake slike, u tex funkciji te zbog toga ona i postoji da se odgovarajući dio koda ne bi ponavljao.

U nastavku se nalazi još i dio koda vezan za glavni progmram.

/*main.cpp*/

```
#include <windows.h>
#include <stdlib.h>
#include <GL/glut.h>
#include <GL/gl.h>
#include <stdio.h>

#include "planets.hpp"
#include "window.hpp"
#include "animate.hpp"
#include "IO.hpp"
#include "texture.hpp"

int main (int argc, char * argv [])
{
	//Inicijaliziramo glut
	glutInit	( &argc , argv );
	glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);

	//Inicijalizacija pozicije i velicine prozora
	glutInitWindowPosition (  0  ,  0  );
	glutInitWindowSize     ( 600 , 400 );
	glutCreateWindow ("Zavrsni projekat");

	init ();

	//Inicijalizacija funkcija
	glutReshapeFunc  (reshape);
	glutKeyboardFunc (kbd);
	glutDisplayFunc  (display);
	glutMouseFunc    (mouse);
	glutIdleFunc     (idle);
	glutMainLoop ();
	return 0;
}
```