ِبسمِ اللهِ الرَحمَن الرَّحِیمِ

منظومه شمسی

نویسنده: رامین اجلال

OpenGl یک امکان در c++ برای امکانات گرافیکی است که کاربر را قادر می سازد تا بتواند محیط بازی ها را شبیه سازی کند. از جمله اشکالات این محیط این است که شما باید فایلهایی را به محیط c++ اضافه کنید. این که این فایلها چه هستند اکنون به خاطر ندارم. ولی می توانید آنرا در کتابهای مربوط به این موضوع پیدا کنید.

در این یک منظومه شمسی بدون امکانات قمرها بجز ماه زمین را مشاهده می کنید. شما می توانید آنرا در محیط خود ویراست کرده و قمرهایی را در اندازه و اشل واقعی به آن اضافه کنید.

این برنامه داری کپی رایت بوده و بدون مجوز نویسنده نمی توانید استفاده تجاری کنید. اگر خواستید آنرا در فضای کمتری فشرده کنید در office 2010 آنرا ویرایش کنید.

```
//Sun set Open Gl .Computer Grpahics Lesson Studing.

//Ramin Edjlal. Urmia. Iran.۲۰۱۱

#include <GL/glut.h>

#include <string.h>


//Initiate Global Varibles.

static float year0 = 0, day0 = 0;

static float year1 = 0, day1 = 0;

static float year2 = 0, day2 = 0;

static float year3 = 0, day3 = 0;

static float year4 = 0, day4 = 0;

static float year5 = 0, day5 = 0;

static float year6 = 0, day6 = 0;

static float year7 = 0, day7 = 0;

static float year8 = 0, day8 = 0;

GLfloat light_diffuse[] =

{1.0, 1.0, 1.0, 1.0};

GLfloat light_position[] =

{0.0, 0.0, 0.0, 0.0};
```

```
float x=45.0;
float y=1.0;
float z=5.0;

//Out a text to graphics
void output(GLfloat x, GLfloat y, char *text)
{

		//Iniatate local varibles.
  char *j;

  //Push Matrix state.
  glPushMatrix();
  glTranslatef(x, y, 0);
  for (j = text; *j; j++)
		 //Out every text character staring.
    glutStrokeCharacter(GLUT_STROKE_MONO_ROMAN, *j);

  //Pop Martix.
  glPopMatrix();
}
void init(void)
{
		//Initiate OpenGl.
   glClearColor (0.0, 0.0, 0.0, 0.0);
   glEnable(GL_LIGHTING);

   glEnable(GL_LIGHT0);

   glEnable(GL_DEPTH_TEST);

}


void display(void)

{

            //Display Function.

   glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

  //Push Matrix.

   glPushMatrix();

   //Lighting varibles.

   GLfloat light_ambient[] = { 1.0, 1.0, 0.0, 1.0};


   //Lightining varibles.

   glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);


   //Create Solid Sphere.

   glutSolidSphere(0.25, 20, 16);   /* draw sun */


   //Rotate For year period.

   glRotatef ((GLfloat) year0, 0.0, 1.0, 0.0);

   //Trasfer Sphre.

   glTranslatef (0.579, 0.0, 0.0);
```

```
//Rotate for day.

glRotatef ((GLfloat) day0, 0.0, 1.0, 0.0);


//Sphere Drawing.

glutSolidSphere(0.00756, 10, 8);    /* draw Atarod planet */


//Pop Matrix.

glPopMatrix();

//Push Matrix.

glPushMatrix();


//Lightining varibles.

light_ambient[0]= 1.0;

light_ambient[1]= 0.0;

light_ambient[2]= 0.0;

light_ambient[3]= 1.0;


//Lightining function.

glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);


//Rotate for year.

glRotatef ((GLfloat) year1, 0.0, 1.0, 0.0);


//Transfer function.

glTranslatef (1.082, 0.0, 0.0);
```

```
glRotatef ((GLfloat) day1, 0.0, 1.0, 0.0);


//Create Sphere.

glutSolidSphere(0.02039, 10, 8);    /* draw Zohreh planet */


//Pop Matrix.

glPopMatrix();

//Push Matrix.

glPushMatrix();


//Lightining variables.

light_ambient[0]= 0.0;

light_ambient[1]= 1.0;

light_ambient[2]= 0.0;

light_ambient[3]= 1.0;


//Lightining function.

glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);


//RoatetRoatet for year.

glRotatef ((GLfloat) year2, 0.0, 1.0, 0.0);


//Transfer function.

glTranslatef (1.496, 0.0, 0.0);
```

```
//Roatet for day.
glRotatef ((GLfloat) day2, 0.1, 1.0, 0.0);

//Create Sphere.
glutSolidSphere(0.02, 10, 8);    /* draw Erthe planet */

//Roatet for year.
glRotatef ((GLfloat) year2, 0.0, 1.0, 0.0);

//Trasfer for day.
glTranslatef (0.02, 0.0, 0.0);

//Roatet for year.
glRotatef ((GLfloat) day2, 0.0, 1.0, 0.0);

//Create Sphere.
glutSolidSphere(0.052, 10, 8);    /* draw Moon planet */

//Pop Matrix.
glPopMatrix();
//Push Matrix.
glPushMatrix();

//Lightining varibales.
light_ambient[0]= 1.0;
```

```
light_ambient[1]= 1.0;
light_ambient[2]= 0.5;
light_ambient[3]= 1.0;

//Lightining function.
glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);

//Roatet function.
glRotatef ((GLfloat) year3, 0.0, 1.0, 0.0);

//Trasfer function.
glTranslatef (2.279, 0.0, 0.0);

//Rotate function.
glRotatef ((GLfloat) day3, 0.0, 1.0, 0.0);

//Create Sphere.
glutSolidSphere(0.01065, 10, 8);    /* draw Merikh planet */

//Pop Matrix.
glPopMatrix();
//Pop Matrix.
glPushMatrix();

//Lightining variables.
```

```
light_ambient[0]= 1.0;
light_ambient[1]= 0.5;
light_ambient[2]= 0.5;
light_ambient[3]= 1.0;

//Lightining function.
glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);

//Rotate function.
glRotatef ((GLfloat) year4, 0.0, 1.0, 0.0);

//Trasfr function.
glTranslatef (7.783, 0.0, 0.0);

//Roatet function.
glRotatef ((GLfloat) day4, 0.0, 1.0, 0.0);

glutSolidSphere(0.224112, 10, 8);    /* draw Zohal planet */

//Pop Matrix.
glPopMatrix();
//Push Matrix.
glPushMatrix();

//Lightinig variables.
```

```
light_ambient[0]= 0.5;
light_ambient[1]= 1.0;
light_ambient[2]= 0.5;
light_ambient[3]= 1.0;

//Lightining function.
glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);

//Roatet function.
glRotatef ((GLfloat) year5, 0.0, 1.0, 0.0);

//Trsfer Matrix.
glTranslatef (14.27, 0.0, 0.0);

//Roatet function.
glRotatef ((GLfloat) day5, 0.0, 1.0, 0.0);

//Create Sphrer.
glutSolidSphere(0.188986, 10, 8);    /* draw Suturn planet */

//Pop Matrix.
glPopMatrix();

//Push Matrix.
glPushMatrix();
```

```
//Lightinig variables.
light_ambient[0]= 0.5;
light_ambient[1]= 1.0;
light_ambient[2]= 0.5;
light_ambient[3]= 1.0;

//Lightining function.
glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);

//Roatet function.
glRotatef ((GLfloat) year6, 0.0, 1.0, 0.0);

//Trsfer Matrix.
glTranslatef (28.71, 0.0, 0.0);

//Roatet function.
glRotatef ((GLfloat) day6, 0.0, 1.0, 0.0);

//Create Sphrer.
glutSolidSphere(0.080148, 10, 8);    /* draw Oranus planet */

//Pop Matrix.
glPopMatrix();
        //Push Matrix.
```

```
            //Push Matrix.
glPushMatrix();

 //Lightinig variables.
light_ambient[0]= 0.5;
light_ambient[1]= 1.0;
light_ambient[2]= 0.5;
light_ambient[3]= 1.0;

//Lightining function.
glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);

//Roatet function.
glRotatef ((GLfloat) year7, 0.0, 1.0, 0.0);

//Trsfer Matrix.
glTranslatef (44.971, 0.0, 0.0);

//Roatet function.
glRotatef ((GLfloat) day7, 0.0, 1.0, 0.0);

//Create Sphrer.
glutSolidSphere(0.077654, 10, 8);    /* draw Nepton planet */

//Pop Matrix.
```

```
  glPopMatrix();
//Push Matrix.
  glPushMatrix();

   //Lightinig variables.
  light_ambient[0]= 0.5;
  light_ambient[1]= 1.0;
  light_ambient[2]= 0.5;
  light_ambient[3]= 1.0;

  //Lightining function.
  glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);

  //Roatet function.
  glRotatef ((GLfloat) year8, 0.0, 1.0, 0.0);

  //Trsfer Matrix.
  glTranslatef (59.13, 0.0, 0.0);

  //Roatet function.
  glRotatef ((GLfloat) day8, 0.0, 1.0, 0.0);

  //Create Sphrer.
  glutSolidSphere(0.00713, 10, 8);    /* draw Ploto planet */
```

```
 //Pop Matrix.
 glPopMatrix();
  //Push Matrix.
 glPushMatrix();



 //Push Atrrbute of Function.
glPushAttrib(GL_ENABLE_BIT);

//Disable some feature
glDisable(GL_DEPTH_TEST);
glDisable(GL_LIGHTING);

//Configure Matrix function.
glMatrixMode(GL_PROJECTION);

 //Pash Matrix.
glPushMatrix();

//Load Identity.
glLoadIdentity();
//Some function.
gluOrtho2D(0, 1500, 0, 1500);

//Configure Matrix Mode.
```

```
glMatrixMode(GL_MODELVIEW);


//Push Mtrix.
glPushMatrix();


//Load Identity.
glLoadIdentity();


//Configure #D Function.
glColor3f(0.0,1.0,0.0);
/* Rotate text slightly to help show jaggies. */
glRotatef(4, 0.0, 0.0, 1.0);


//Aouther name.
output(200, 225, "Ramin Edjlal.");


//Disable some feature.
glDisable(GL_LINE_SMOOTH);
glDisable(GL_BLEND);


//Student number outpurt.
output(160, 100, "880879004");


//Pop Matrix.
glPopMatrix();
```

```
 //Matrix Mode.
 glMatrixMode(GL_PROJECTION);

 //Pop Matrix.
 glPopMatrix();

 //Pop Attriburt.
 glPopAttrib();

 //Matrix Mode.
 glMatrixMode(GL_MODELVIEW);

 //Swap Buffer function.
 glutSwapBuffers();
}

void reshape (int w, int h)
{
        //Reshape function.
  glViewport (0, 0, (GLsizei) w, (GLsizei) h);
  glMatrixMode (GL_PROJECTION);
  glLoadIdentity ();
  gluPerspective(45.0, (GLfloat) w/(GLfloat) h, 1.0, 20.0);
  glMatrixMode(GL_MODELVIEW);
```

```
   glLoadIdentity();

   gluLookAt (3.5,3.5,7.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

}


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

{

             //KeayBoard.

   switch (key) {

      case 'd':

         day0 = (int)(day0 + 1) % 360;

                            day1 = (int)(day1 + 2) % 360;

                            day2 = (int)(day2 + 3) % 360;

                            day3 = (int)(day3 + 4) % 360;

                            day4 =(int) (day4 + 5) % 360;

                            day5 = (int)(day5 + 6) % 360;

                            glutPostRedisplay();

         break;

      case 'D':

         day0 = (int)(day0 - 1) % 360;

                            day1 = (int)(day1 - 2) % 360;

                            day2 = (int)(day2 - 3) % 360;

                            day3 = (int)(day3 - 4) % 360;

                            day4 = (int)(day4 - 5) % 360;

                            day5 = (int)(day5 - 6) % 360;

                            glutPostRedisplay();
```

```
  break;

case 'a':

  year0 = (int)(year0 + 36) % 360;

         year1 = (int)(year1 + 7) % 360;

                year2 = (int)(year2 + 5) % 360;

                year3 = (int)(year3 + 3) % 360;

                year4 = (int)(year4 + 2) % 360;

                year5 = (int)(year5 + 1) % 360;

  glutPostRedisplay();

  break;

case 'A':

  year0 = (int)(year0 - 36) % 360;

         year1 = (int)(year1 - 7) % 360;

                year2 = (int)(year2 - 5) % 360;

                year3 = (int)(year3 - 3) % 360;

                year4 = (int)(year4 - 2) % 360;

                year5 = (int)(year5 - 1) % 360;

  glutPostRedisplay();

  break;

       case 'f':

  day0 = (int)(day0 + 1) % 360;

                day1 = (int)(day1 + 2) % 360;

                day2 = (int)(day2 + 3) % 360;

                day3 = (int)(day3 + 4) % 360;

                day4 = (int)(day4 + 5) % 360;
```

```
                    day5 = (int)(day5 + 6) % 360;

                    year0 = (int)(year0 + 36) % 360;

             year1 = (int)(year1 + 7) % 360;

                    year2 = (int)(year2 + 5) % 360;

                    year3 = (int)(year3 + 3) % 360;

                    year4 = (int)(year4 + 2) % 360;

                    year5 = (int)(year5 + 1) % 360;

      glutPostRedisplay();

      break;

    default:

      break;

  }

}

GLvoid Timer( int value )

{

          //Timer function.

  if( value ) glutPostRedisplay();

  glutTimerFunc(40,Timer,value);

}

static void _Timer(int value)

{

 /* increment angle */

      day0 = (int)(day0 + 1) % 360;

                    day1 = (int)(day1 + 2) % 360;

                    day2 = (int)(day2 + 3) % 360;
    day3 = (int)(day3 + 4) % 360;
    day4 = (int)(day4 + 5) % 360;
    day5 = (int)(day5 + 6) % 360;
    day6 = (int)(day5 + 6) % 360;
    day7 = (int)(day5 + 8) % 360;
    year0 = (year0 + 44.51 ) ;
year1 = (year1 + 27.38 ) ;
    year2 = (year2 + 14.55) ;
    year3 = (year3 + 2.31)  ;
    year4 = (year4 + 0.93)  ;
    year5 = (year5 + 0.33 ) ;
    year6 = (year6 + 0.18)  ;
    year7 = (year7 + 0.01)  ;
    year8 = (year8 + 0.01)  ;
    if(year0>360)
         year0=(int)year0%360;

    if(year1>360)
         year1=(int)year1%360;

    if(year2>360)
         year2=(int)year2%360;

    if(year3>360)
         year3=(int)year3%360;
```

```
        if(year4>360)

                        year4=(int)year4%360;


                if(year5>360)

                        year5=(int)year5%360;


          if(year6>360)

                        year6=(int)year6%360;


          if(year7>360)

                        year7=(int)year7%360;


        if(year8>360)

                        year8=(int)year8%360;




  /* send redisplay event */

  glutPostRedisplay();


  /* call this function again in 10 milliseconds */

  glutTimerFunc(50, _Timer, 0);

}


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

```
{
        //Main functinon.
  glutInit(&argc, argv);

  //Initate Display Mode.
  glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);

  //Initate Windows size.
  glutInitWindowSize (1000, 1000);

  //Initiate Windows position.
  glutInitWindowPosition (100, 100);

  //Create a window.
  glutCreateWindow (argv[0]);

  //Initiate.
  init ();

  //Displaye function call.
  glutDisplayFunc(display);

  //Reshape function.
  glutReshapeFunc(reshape);
  //Timer function.
   glutTimerFunc(10, _Timer, 0);


   //Main loop.
   glutMainLoop();
   return 0;
}
```