VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama ", Belgaum -590 018, Karnataka State, India

A MINI PROJECT REPORT ON

"Celestial Exploratory"

Submitted on partial fulfilment of academic requirement of 6th semester

COMPUTER GRAPHICS AND VISUALIZATION LAB

BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE AND ENGINEERING

Submitted by

NAME: NIKITHA MD
USN: 1SJ19CS109

NAME: PALLAVI N
USN: 1SJ19CS112

Under the guidance of

SRINATH G M
Assistant Professor
Department of CSE, SJCIT

CHANDANA K R
Assistant Professor
Department of CSE, SJCIT

Carried out
at

COMPUTER GRAPHICS AND VISUALIZATION LABORATORY, SJCIT

S J C INSTITUTE OF TECHNOLOGY

Department of Computer Science and Engineering
Chickaballapur – 562 101 2021-2022

|| Jai Sri Gurudev ||

Sri Adichunchanagiri Shikshana Trust ®

S.J.C INSTITUTE OF TECHNOLOGY

Department of Computer Science and Engineering

Chickballapur-562101

CERTIFICATE

This is to certify that the Project work entitled "**Celestial Exploratory**" is a bona fide work carried out at Computer Graphics And Visualization Laboratory by **NIKITHA MD (1SJ19CS109)** and **PALLAVI N (1SJ19CS112)** in partial fulfilment for the award of Bachelor of Engineering in Computer Science and Engineering in Sixth semester of the Visvesvaraya Technological University, Belgaum during the year 2021-22. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the department library. The project report has been approved as it satisfies the academic requirements in respect to Sixth Semester Mini Project work prescribed for the said degree.

.....	.....	.....
Signature of the Guide	**Signature of the Guide**	**Signature of the HOD**
Srinath G M	Chandana K R	Dr. Manjunatha Kumar B H
Assistant Professor	Assistant Professor	Professor and HOD,
Department of CSE, SJCIT	Department of CSE, SJCIT	Department of CSE, SJCIT

External Viva:
Name and signature of the Examiners

1. **.....**
2. **.....**

ACKNOWLEDGEMENT

With reverential pranam, we express our sincere gratitude and salutations to the feet of his holiness **Paramapoojya Jagadguru Byravaikya Padmabhushana Sri Sri Sri Dr. Balagangadharanatha Maha Swamiji**, his holiness **Paramapoojya Jagadguru Sri Sri Sri Dr. Nirmalanandanatha Maha Swamiji** and **Pramapoojya Sri Sri Mangalnatha Swamiji**, Sri Adichunchanagiri Mutt for their unlimited blessings.

First and foremost, we wish to express our deep sincere feelings of gratitude to our institution, **Sri Jagadguru Chandrashekaranatha Swamiji Institute of Technology**, for providing me an opportunity for completing my Computer Graphics and Visualization Lab mini-project successfully.

We extend deep sense of sincere gratitude to **Dr. G T Raju, Principal, S J C Institute of Technology, Chickballapur**, for providing an opportunity to complete the Computer Graphics and Visualization Lab mini project.

We extend special in-depth, heartfelt, and sincere gratitude to HOD **Dr. Manjunatha Kumar B H, Head of the Department, Computer Science and Engineering, S J C Institute of Technology, Chickballapur**, for his constant support and valuable guidance of the Computer Graphics and Visualization Lab mini-project.

We convey my sincere thanks to Project Guide **SRINATH G M and CHANDANA K R, Assistant Professor, Department of Computer Science and Engineering, S J C Institute of Technology**, for their constant support, valuable guidance and suggestions of the Computer Graphics and Visualization Lab mini project.

We also thank all those who extended their support and co-operation while bringing out this Computer Graphics and Visualization Lab mini-project.

NAME : NIKITHA MD
USN : 1SJ19CS109

NAME : PALLAVI N
USN : 1SJ19CS112

ABSTRACT

Computer graphics is the process of making the design, 2D, 4D and animation of an object. Computer graphics can do many things, including modelling, simulation and visualization of an object or a problem.

Visualization can also be done to facilitate the delivery of a material in a formal classroom or school.Solar system is a set of celestial bodies bound by gravitational forces. The movement of celestial bodies like the sun, stars, planets and the other will be more easily understood if taught through visualization movement through computer animation. This visualization shows the solar system planetary motion, or we can call it a revolution, that is, when the planets move around the sun, and remain in orbit each using OpenGL API to represent the solar system as a visual.

LIST OF FIGURES

FIGURES

INDEX

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Computer Graphics

Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D pattern recognition abilities allow us to perceive and process pictorial data rapidly and efficiently. Interactive computer graphics is the most important means of producing pictures since the invention of photography and television. It has the added advantage that, with the computer, we can make pictures not only of concrete real world objects but also of abstract, synthetic objects, such as mathematical surfaces and of data that have no inherent geometry, such as survey results.

1.2 OpenGL

OpenGL (Open Graphics Library) is a standard specification defining a cross language cross platform API for writing applications that produce 2D and 3D computer graphics. The interface consists of over 250 different function calls which can be used to draw complex 3D scenes from simple primitives. OpenGL was developed by Silicon Graphics Inc. (SGI) in 1992 and is widely used in CAD, virtual reality, scientific visualization, information visualization and flight simulation. It is also used in video games, where it competes with direct 3D on Microsoft Windows Platforms. OpenGL is managed by the non profit technology consortium, the Khronos group Inc.

OpenGL serves two main purposes :

* To hide the complexities of interfacing with different 3D accelerators, by presenting programmer with a single, uniform API

* To hide the differing capabilities of hardware platforms , by requiring that all implementations support the full OpenGL feature set

OpenGL has historically been influential on the development of 3D accelerator, promoting a base level of functionality that is now common in consumer level hardware:

* Rasterized points, lines and polygons are basic primitives.
* A transform and lighting pipeline .
* Z buffering .
*Texture Mapping.
* Alpha Blending.

1.2.1 OpenGL Graphics Architecture :

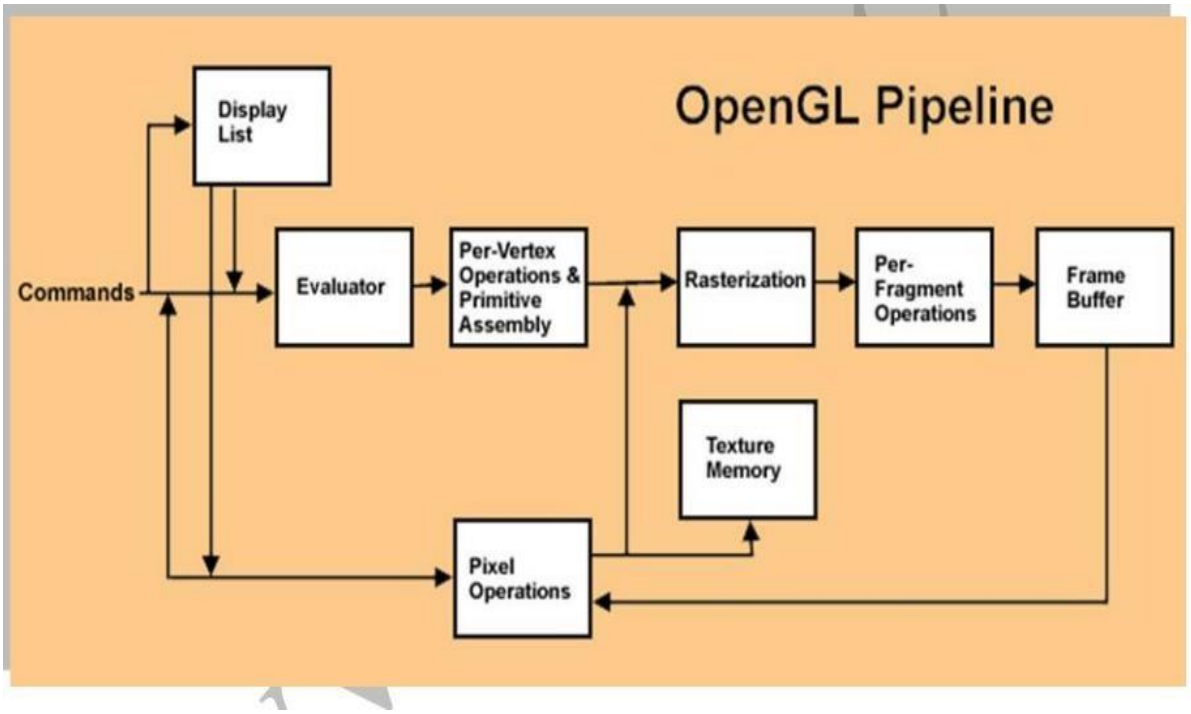

Figure 1.1 Opengl Graphics Architecture

Display Lists :

All data, whether it describes geometry or pixels, can be saved in a display list for current or later use. When a display list is executed, the retained data is sent from the display list just as if it were sent by the application in immediate mode.

Evaluators :

All geometric primitives are eventually described by vertices. Parametric curves and surfaces may be initially described by control points and polynomial functions called basis functions.

Per Vertex Operations :

For vertex data, next is the "per-vertex operations" stage, which converts the vertices into primitives. Some vertex data are transformed by 4 x 4 floating-point matrices. Spatial coordinates are projected from a position in the 3D world to a position on your screen.

Pixel Operation:

While geometric data takes one path through the OpenGL rendering pipeline, pixel data takes a different route. Pixels from an array in system memory are first unpacked from one of a variety of formats into the proper number of components. Next the data is scaled, biased, and processed by a pixel map. The results are clamped and then either written into texture memory or sent to the rasterization step.

Rasterization:

Rasterization is the conversion of both geometric and pixel data into fragments. Each fragment square corresponds to a pixel in the framebuffer. Color and depth values are assigned for each fragment square

Fragment Operations :

Before values are actually stored into the framebuffer, a series of operations are performed that may alter or even throw out fragments. All these operations can be enabled or disabled.

1.3 Project Goal

The aim of this project is to show the shadow implementation using OPENGL which include Movement, Light properties also transformation operations like translation, rotation, scaling etc on objects. The package must also have a user friendly interface .

1.4 Scope

It is developed in ECLIPSE. It has been implemented on UBUNTU platform. The 3-D graphics package designed here provides an interface for the users for handling the display and manipulation of Celestial Exploratory. The Keyboard is the main input device used

CHAPTER 2

LITERATURE SURVEY

2.1 SOLAR SYSTEM :

The Solar System consists of the Sun and the astronomical objects bound to it by gravity, all of which formed from the collapse of a giant molecular cloud approximately 4.6 billion years ago. Of the many objects that orbit the Sun, most of the mass is contained within eight relatively solitary planets[e] whose orbits are almost circular and lie within a nearly flat disc called the ecliptic plane. The four smaller inner planets, Mercury, Venus, Earth and Mars, also called the terrestrial planets, are primarily composed of rock and metal. The four outer planets, the gas giants, are substantially more massive than the terrestrials. The two largest, Jupiter and Saturn, are composed mainly of hydrogen and helium; the two outermost planets, Uranus and Neptune, are composed largely of ices, such as water, ammonia and methane, and are often referred to separately as "ice giants".

The Solar System is also home to two regions populated by smaller objects. The asteroid belt, which lies between Mars and Jupiter, is similar to the terrestrial planets as Six of the planets and three of the dwarf planets are orbited by natural satellites, usually termed "moons" after Earth's Moon. Each of the outer planets is encircled by planetary rings of dust and other particles.

The principal component of the Solar System is the Sun, a main sequence G2 star that contains 99.86 percent of the system's known mass and dominates it gravitationally. The Sun's four largest orbiting bodies, the gas giants, account for 99 percent of the remaining mass, with Jupiter and Saturn together comprising more than 90 percent.

Most large objects in orbit around the Sun lie near the plane of Earth's orbit, known as the ecliptic. All the planets and most other objects also orbit with the Sun's rotation (counter clockwise, as viewed from above the Sun's north pole).

The overall structure of the charted regions of the Solar System consists of the Sun, four relatively small inner planets surrounded by a belt of rocky asteroids, and four gas giants surrounded by the outer Kuiper belt of icy objects. Astronomers sometimes informally divide this structure into separate regions. The inner Solar System includes the four terrestrial planets and the main asteroid belt. The outer Solar System is beyond the asteroids, including the four gas giant planets. Since the discovery of the Kuiper belt, the outermost parts of the Solar System are considered a distinct region consisting of the objects beyond Neptune.

Kepler's laws of planetary motion describe the orbits of objects about the Sun. According to Kepler's laws, each object travels along an ellipse with the Sun at one focus. Objects closer to the Sun (with smaller semi-major axes) travel more quickly, as they are more affected by the Sun's gravity. On an elliptical orbit, a body's distance from the Sun varies over the course of its year. The orbits of the planets are nearly circular.

Due to the vast distances involved, many representations of the Solar System show orbits the same distance apart. In reality, with a few exceptions, the farther a planet or belt is from the Sun, the larger the distance between it and the previous orbit. Most of the planets in the Solar System possess secondary systems of their own, being orbited by planetary objects called natural satellites, or moons (two of which are larger than the planet Mercury) or in the case of the four gas giants, by planetary rings; thin bands of tiny particles that orbit them in unison.

Most of the largest natural satellites are in synchronous rotation, with one face permanently turned toward their parent. The objects of the inner Solar System are composed mostly of rock, the collective name for compounds with high melting points, such as silicates, iron or nickel, that remained solid under almost all conditions in the protoplanetary nebula.

Jupiter and Saturn are composed mainly of gases, the astronomical term for materials with extremely low melting points and high vapor pressure such as molecular hydrogen, helium, and neon, which were always in the gaseous phase in the nebula.

2.2 Proposed System:

To achieve three dimensional effects, OpenGL software is proposed . It is software which provides a graphical interface. It is a interface between application program and graphics hardware. the advantages are:

1. OpenGL is designed as a streamlined.
2. It is a hardware independent interface i.e. it can be implemented on many different hardware platforms.
3. With OpenGL, we can draw a small set of geometric primitives such as points, lines and polygons etc.
4. It provides double buffering which is vital in providing transformations.
5. It is event driven software.
6. It provides call back function.

CHAPTER 3

SYSTEM REQUIREMENTS

3.1 Hardware Requirements:

- Processor : intel 486/Pentium processor and above
- Processor speed : 500MHz or above
- RAM : 64 MB or above
- Storage space : 2 MB or above
- Monitor Resolution : colour monitor with minimum resolution of 640*480

3.2 Software Requirements:

- An MS_DOS based operating system like windows XP
- A visual C/C++ compiler is required for compiling the source code to make a executable file which can be directly executed.• The built in graphics and dynamic link libraries like glut and glut32, and header file like glut.h to create 2D &3D layout.

3.3 Functional Requirements:

The whole mini project is divided into many small parts known as functions and these functions would take care of implementing particular parts of this mini project which makes the programming easy and effective. Each function takes the responsibility of designing the assigned parts hence we combined all the functions at a particular stage to get the final required design.

The developed mini project also displays the statements of services the animation should provide, how the system should react to particular inputs and how the system should behave in particular situations.

3.4 Non-Functional Requirements:

The developed mini project also displays the constraints on the services or functions offered by the system. They include constraints on the development process, standards, etc.

CHAPTER 4

SYSTEM DESIGN AND IMPLEMENTATION

4.1 SOFTWARE DESIGN

The project “CELESTIAL EXPLORATORY” is meant as a source of recreation where one can sit in front of the computer and have the vision of a plant in space. This package is developed to provide opportunities to climb aboard the earth for the adventure of the lifetime. It is aimed to create starts and planets and give constant motion to these objects. The sun and its family of eight planets are imagined to be placed in a background of bright twinkling starts along with a comet in constant motion. The lighting effect in the background appears as though the planet is rotating and revolving around the sun in the galaxy. The most important aspect of this project is that ,one can sit back ,relax and watch constantly occurring motion of the planet and the stars just depicting the fact that “as passengers of the earth our voyage never ends!”

This chapter documents a detailed description of the implementation of our project. We have incorporated several inbuilt OpenGL function in this project.

The following code snippet enables the easy rendering of solid sphere with different colors and makes them to rotate and translate.

```
{                                                                                    .
. . . .
glRotatef(s. . .);
glTranslatef(. . .);
glRotatef(. . .);
glColor3f(. . .);
glutSolidSphere(. . .);
. . . . .
}
```

The header files used are :

1. #include:This is C library function for standard input and output.

2. #include:This header is included to read the glut.h, gl.h and glu.h.

3. #include:This is a C library function for performing certain mathematical operations.

In the Init() we have made use of the following functions:

1. **glClearColor(. . .):**

Whenever we wish to draw a new frame,the window must be cleared by using the 4dimensional(RGBA)color system.The above function must be enabled to make the window on screen solid and white.

2. **glshadeModel(. . .):**

To enable the smooth shading we must set the shade as follows glShadeModel(GL_SMOOTH);

3. **glEnable(. . .):**

The z_buffer is one of the buffers that make up the frame buffer.The depth buffer must be cleared whenever we wish to redraw the display.This is done as follows glEnable(GL_DEPTH_TEST);

4. **glMaterial(. . .):**

We can specify different material properties for the front and back faces of a surface through the following functions glMaterialfv(GLenum face, GLenum type,GLfloat *pointer_to_array); glMaterialfv(GLenum face,GLenum type,GLfloat value);

5. **glLight(. . .):**

This function is used to enable a light source.The following function specifies the required vector and scalar parameters to enable a light source. glLightfv(GLenum source,GLenum parameter,GLfloat*pointer_to_arrary)) glLightf(GLenum source,GLenum parameter,GLfloat value)

6. **glColorMaterial(. . .):**

This function is used to change a single material property.

myinit();

Here we initialize the color buffer, set the point size and set window co-ordinate values.

display();

This function creates and translates all the objects in a specified locatioN in a particular order.

Translated(. . .);

In this function the variables are components of displacement vector.

glutPostRedisplay();

It ensures that display will be drawn only once each time program goes through the event loop.

glutMainLoop();

This function whose execution will cause the program to begin an event processing loop.

User Interface:

Keyboard Based Interface

Using the keyboard user can make the planets to rotate on their own axis and revolve round the Sun. The stars are made to twinkle and the Comet is made to revolve round the Sun.

1. The keys **m, v, e, r, j, s, u, n** are used to rotate the planets.
2. The keys **M, V, E, R, J, S, U, N** are used to revolve the planets around the Sun.
3. The key **z** rotates the sun, **B** gives both the rotation and revolution of the planets around the rotating Sun with a Comet revolution and Stars twinkle.
4. Pressing the key **A** revolves all the planets and comet and the key a rotates all the planets around the rotating Sun with Stars twinkling in the background.
5. The key **b** is used to make the stars twinkle and c for the revolution of the Comet.

Mouse Interface

Using the mouse user can make the planets to rotate and revolve round the Sun and Comet to revolve round the Sun.

Left Button: Rotates and revolves the planets and Comet in anticlockwise direction.
Middle Button: Rotates and revolves the planets and Comet in clockwise direction.
Right Button: Rotates and revolves the planets and Comet in clockwise direction.

4.2 IMPLEMENTATION :

The implementation of the different objects in this project is divided into different module.

MODULE 1:

SUN:

The sun is drawn by using the following lines of code.

```
{                                                                    glPushMatrix();
glRotatef(…);
glLightfv(GL_LIGHT0,GL_POSITION,position);
glDisable(GL_LIGHTING);
glutSolidSphere(…);
glPopMatrix();
}
```

MODULE 2:

PLANETS WITH RINGS:

The planets Saturn and Uranus are the 2 planets in our solar system with rings. They are implemented using the following codes.

```
 {
 glPushMatrix();
 glRotatef(…);
glTranslatef(…);
 gluLookAt(0.0,10.0,2.0,1.0,0.0,0.0,0.0,0.0,1.0);
 glutSolidSphere(…);
 int i=0;
 glBegin(GL_QUAD_STRIP);
for(i=0;i<=360;i++)
{
glVertex3f(sin(i*3.1416/180)*0.5,cos(i*3.1416/180)*0.5,0);
glVertex3f(sin(i*3.1416/180)*0.7,cos(i*3.1416/180)*0.7,0);
}                                                                    glEnd();
glPopMatrix();
}
```

MODULE 3:

EARTH:

The earth is drawn along with its natural satellite, moon which revolve round the earth. The following lines of codes are used to implement the earth and the moon.

```
{                                                                    glPushMatrix();
glRotatef(…);
glTranslatef(…);
glRotatef(…);
glColor3f(…);
glutSolidSphere(…); /*draw planet earth*/
```

```
glRotatef(…);
glTranslatef(…);
glColor3f(…);
glutSolidSphere(…); /*draw moon*/
glPopMatrix();                                                    }
```

MODULE 4:

OTHER PLANETS:

The remaining planets are Mercury, Venus, Mars, Jupiter and Neptune. All these planets are implemented using the same set of codes by changing the values and colors.

```
{
glPushMatrix();
glRotatef(…);
glTranslatef(…);
glRotatef(…);
glColor3f(…);
glutSolidSphere(…); /*draw smaller planet mercury*/
glPopMatrix();
}
```

MODULE 5:

STARS:

The stars are implemented in the background using the following lines of codes.

```
{
glPushMatrix();                                                   glTranslatef(…);
gluLookAt(0.0,10.0,0.0,1.0,0.0,0.0,0.0,0.0,3.0);
glRotatef(…);
glScalef(…);
glColor3f(…);
glutSolidSphere(…);
glPopMatrix();                                                    }
```

MODULE 6:

COMET:

The comet is made to revolve round the sun.
The following codes are used to implement the comet.

```
{
glPushMatrix();
glRotatef((GLfloat)c,6.0,-14.0,-6.0);
```

```
glTranslatef(5.0,3.0,-1.0);
glScalef(0.60,0.0,2.5);
glColor3f(7.5,9.5,2.0);
glutSolidSphere(0.2,12,6); /*draw comet*/
glPopMatrix();
}
```

CHAPTER 5

TESTING

5.1Introduction to Testing:

Software testing is a critical element of the ultimate review of specification design and coding. Testing of software leads to the uncovering of errors in the software functional and performance requirements are met. Testing also provides a good indication of software reliability and software quality as a whole. The result of different phases of testing are evaluated and then compared with the expected results. If the errors are uncovered they are debugged and corrected. A strategy approach to software testing has the generic characteristics:

- Different testing techniques are appropriate at different points of time.
- Testing and debugging are different activities, but debugging must be accommodated in the testing strategy.

Following three approaches of debugging were used:

- Debugging by Induction
- Debugging by Deduction
- Backtracking

5.2 Test Plan:

In this test plan all major activities are described below:

- Unit testing.
- Integration testing.
- Validation testing.
- System testing.

CHAPTER 6

SNAPSHOTS

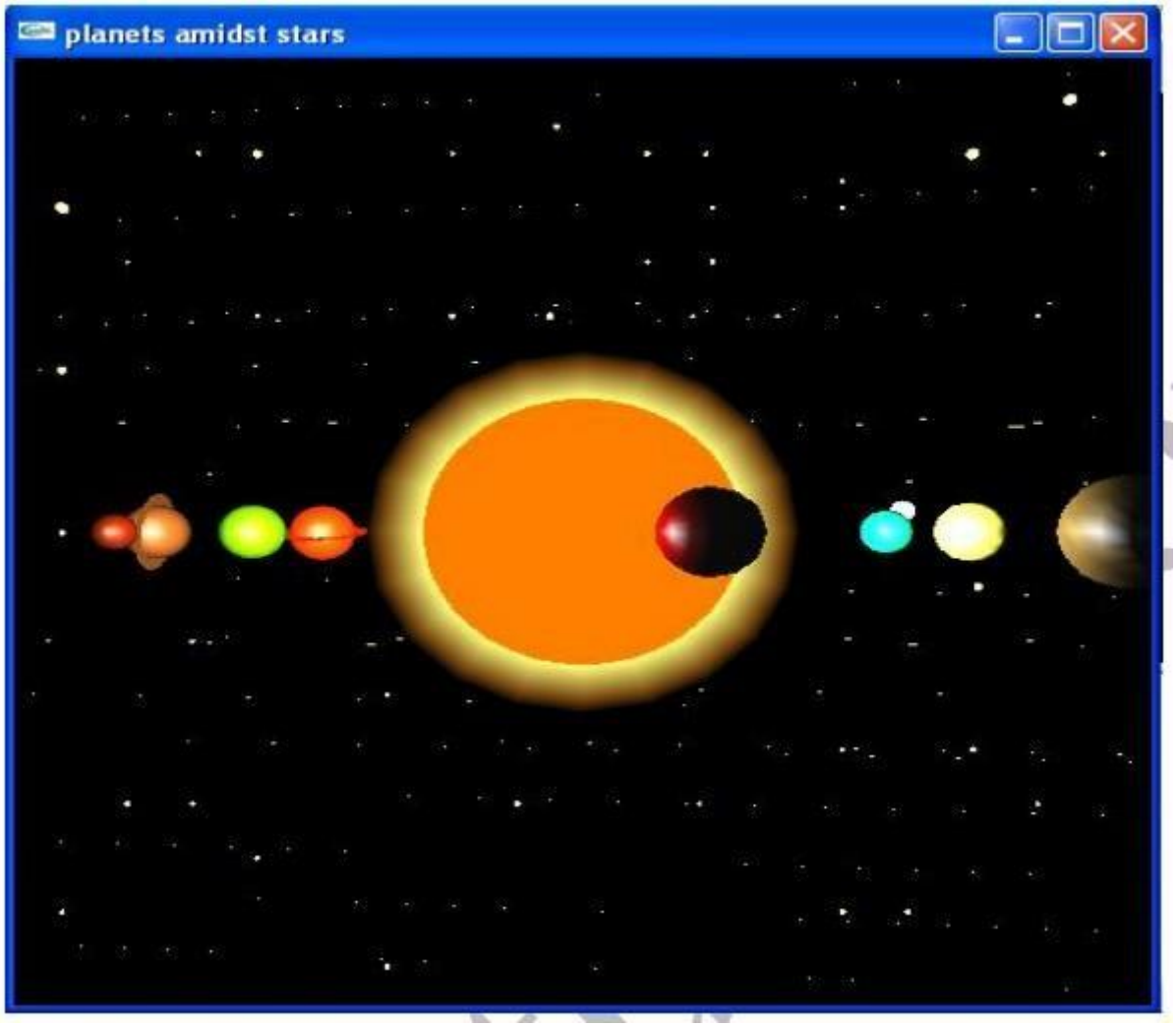

Figure 6.1: Solar system with revolution of mercury

Here we can see the sun at the center and all planets revolving around the sun.In this we have mainly highlighted planet mercury revolving around the sun.It appears dark because the light is falling on the front face of the planet mercury but the viewer is viewing the back face

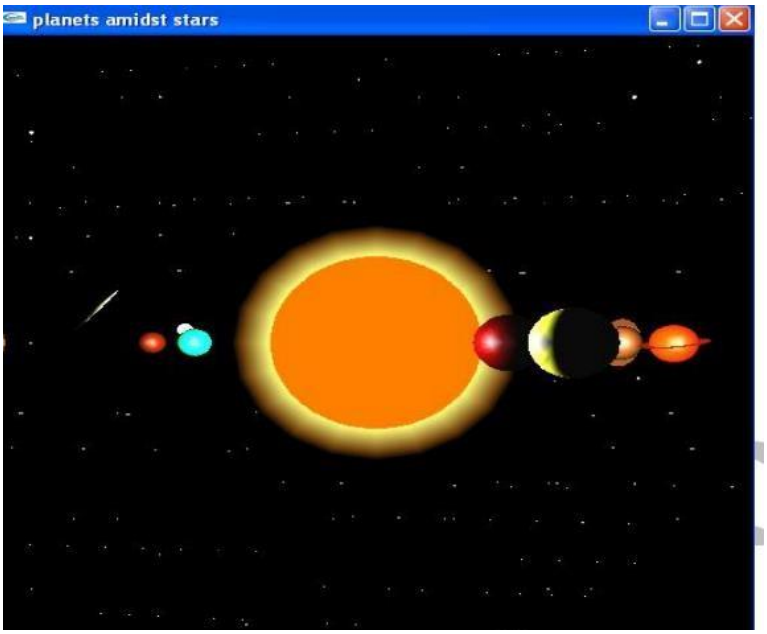

Fig 6.2 Solar system with revolution of planets and comet

In this snapshot, we can see sun at the center and all eight planets Revolving around the sun and are placed in the background of bright twinkling stars with the comet in a constant motion

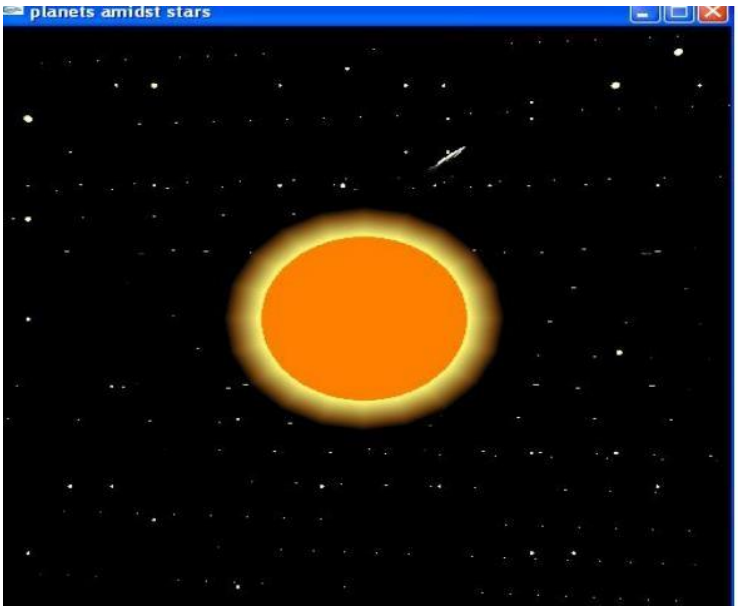

Fig 6.3 Sun ,twinkling stars and comet

In this snapshot we can the sun that is imagined to be placed at the center and is also olaced in the background of bright twinkling stars and the comet passing with a constant motion through the sun from the left bottom end to right top end

Fig 6.5 Final view of solar system

In this snapshot ,sun is placed at the center and its eight planets are placed in the sun's orbit. These eight planets are shown to be rotating around the sun .the planets and sun are placed in The background of bright twinkling stars and we can also see the comet passing through these planets and sun with constant motion

CHAPTER 7 CONCLUSION

An attempt has been made to develop an openGL which meets necessary requirements of the user successfully. Since it is user friendly it enables user to interact efficiently and easily.

The development of mini project has given us a good exposure to openGL by which we have learnt some of the techniques which helps in the development of animation and gaming.

Hence it is helpful for us to even take this field as our career too and develop some other features in openGL and provide as token of contribution to the graphics world.

BIBLIOGRAPHY

1. Edward Angel: Interactive Computer Graphics: A Top-Down Approach with 5th Edition, Addison-wesley,2008.

2. James D Foley, Andries Van Dam, Steven K Feiner, Jhon F Hughes: - Computer Graphics, Addison-Wesley 1997.

3. Yashavanth Kanetkar, **"Graphics under opengl"**, Published by BPB Publications.

4. Athul P.Godse, **"Computer Graphics"**,Technical Publications Pune, 2 nd Revised Edition.

5. James D.foley, Andries Van Dam, Steven K.Feiner, John F.Hughes, **"Computer Graphics Principle & Practice"** Published by Pearson Education Pte.ltd, 2 nd Edition.

WEBSITES

[1] www.OpenGLRedbook.com

[2] www.OpenGLsimpleexamples.com

[3] www.OpenGLprogrammingide.com

[4] www.wikipedia.com

APPENDIX A- SOURCE CODE

```
#include
#include  #include  static int m=0,M=0,v=0,V=0,E=0,e=0,r=0,R=0,j=0,J=0,s=0,S=0,U=0,u=0,n=0,N=0,X=0,z=0,B=0,b=0, c=0;
static GLint axis=2;  GLfloat diffuseMaterial[4]={0.5,0.5,0.5,1.0}; /*initialize material property,light soure,lighting model,and
depth buffer*/  void myinit(void)
{
 glClearColor(0.0,0.0,0.0,0.0);
glShadeModel(GL_SMOOTH);
 glEnable(GL_DEPTH_TEST);
GLfloat mat_specular[]={1.0,1.0,1.0,1.0};  GLfloat
light_position[]={1.0,1.0,1.0,0.0};
glMaterialfv(GL_FRONT,GL_DIFFUSE,diffuseMaterial);
glMaterialfv(GL_FRONT,GL_SPECULAR,mat_specular);
glMaterialf(GL_FRONT,GL_SHININESS,25.0);
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0); glLightfv(GL_LIGHT0,GL_POSITION,light_position);  glColorMaterial(GL_FRONT,GL_DIFFUSE);
glEnable(GL_COLOR_MATERIAL);
}
void display(void)
 {
GLfloat position[]={0.0,0.0,1.5,1.0};
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
glColor3f(1.0,0.5,0.0); glPushMatrix();
 glRotatef((GLfloat)z,1.0,1.0,1.0);
glLightfv(GL_LIGHT0,GL_POSITION,position);
glDisable(GL_LIGHTING);
glutSolidSphere(0.8,40,16); /*draw sun*/
glPopMatrix();  glPushMatrix();
glLightfv(GL_LIGHT0,GL_POSITION,position);
glDisable(GL_LIGHTING);
 glEnable(GL_LIGHTING);
glColor3f(1.5,0.5,0.0);
glutSolidTorus(0.2,0.9,6,20);
glPopMatrix();  glPushMatrix();
glRotatef((GLfloat)M,0.0,1.0,0.0);
glTranslatef(1.5,0.0,0.0);
glRotatef((GLfloat)m,0.0,1.0,0.0);
glColor3f(1.0,0.0,0.0);
glutSolidSphere(0.2,20,8); /*draw smaller planet mercury*/
glPopMatrix();  glPushMatrix();
glRotatef((GLfloat)V,0.0,1.0,0.0);
glTranslatef(2.0,0.0,1.0);
glRotatef((GLfloat)v,0.0,1.0,0.0);  glColor3f(7.5,9.5,1.0);
 glutSolidSphere(0.2,20,8); /*draw smaller plant venus*/
glPopMatrix();  glPushMatrix();
 glRotatef((GLfloat)E,0.0,1.0,0.0);
glTranslatef(3.5,0.0,0.0);
glRotatef((GLfloat)e,0.0,1.0,0.0);  glColor3f(0.1,6.5,2.0);
 glutSolidSphere(0.2,20,8); /*draw smaller plant earth*/
glRotatef((GLfloat)X,0.0,1.0,0.0);
glTranslatef(0.3,0.2,0.0);  glColor3f(4.3,3.5,8.0);
glutSolidSphere(0.1,20,14); /*draw moon*/
 glPopMatrix();  glPushMatrix();
 glRotatef((GLfloat)R,0.0,1.0,0.0);
glTranslatef(5.0,0.0,3.0);
glRotatef((GLfloat)r,0.0,1.0,0.0);  glColor3f(1.0,0.2,0.0);
 glutSolidSphere(0.2,20,8); /*draw smaller planet mars*/
glPopMatrix();  glPushMatrix();
glRotatef((GLfloat)J,0.0,1.0,0.0);
glTranslatef(-2.5,0.0,1.0);
glRotatef((GLfloat)j,0.0,1.0,0.0);  glColor3f(0.9,0.7,0.3);
glutSolidSphere(0.2,20,8);/*draw smaller planet Jupiter*/
glPopMatrix();  glPushMatrix();
 glRotatef((GLfloat)S,0.0,1.0,0.0);  glTranslatef(-
5.0,0.0,0.0);
 gluLookAt(0.0,10.0,2.0,1.0,0.0,0.0,0.0,0.0,1.0);
glRotatef((GLfloat)s,0.0,0.0,5.0);  glColor3f(4.5,0.5,0.0);
 glutSolidSphere(0.5,20,16); /*draw smaller plant Saturn*/
int i=0;
glBegin(GL_QUAD_STRIP);  for(i=0;i<=360;i++)  {
glVertex3f(sin(i*3.1416/180)*0.5,cos(i*3.1416/180)*0.5,0);
glVertex3f(sin(i*3.1416/180)*0.7,cos(i*3.1416/180)*0.7,0);
```

```
 }  glEnd();  glPopMatrix();  glPushMatrix();
glRotatef ((GLfloat) U, 0.0, 1.0,0.0);  glTranslatef (-
6.5, 0.0, 0.0);  gluLookAt (10.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 10.0, 1.0);  glRotatef((GLfloat) u, 0.0, 0.0, 5.0);
glColor3f( 1.2, 0.6,0.2);  glutSolidSphere (0.5, 20,
16);  /* draw smaller planet Uranus*/
glBegin(GL_QUAD_STRIP);
 for(i=0; i<=360; i++)
{  glVertex3f(sin(i*3.1416/180)*0.5,cos(i*3.1416/180)*0.5,
0);
 glVertex3f(sin(i*3.1416/180)*0.7, cos(i*3.1416/180)*0.7,0);
}  glEnd();  glPopMatrix();  glPushMatrix();
glRotatef ((GLfloat) N,0.0, 1.0, 0.0);
glTranslatef (-8.0, 0.0, 0.0);  glRotatef
((GLfloat) n, 0.0, 1.0, 0.0);  glColor3f(1.0, 2.0,
0.0);  glutSolidSphere(0.4, 20, 8);
glPopMatrix();/* draw smaller planet Neptune */
glPushMatrix();  glRotatef ((GLfloat) c, 6.0, -
14.0,-6.0);  glTranslatef (5.0,3.0,-1.0);
glScalef(0.60,0.0,2.5);  glColor3f (7.5, 9.5, 2.0);
glutSolidSphere (0.2, 12, 6);
glPopMatrix();/*draw comet*/ //to put the stars as a background
glPushMatrix();  glTranslatef(0.0,-2.0,0.0);
gluLookAt(0.0,10.0,0.0,1.0,0.0,0.0,0.0,0.0,3.0);
glRotatef((GLfloat)b,0.0,0.0,0.0); glScalef(200.0,0.0,0.0);
glColor3f(4.3,3.5,1.0);  glutSolidSphere(0.04,20,8);
glPopMatrix();  glPushMatrix();  glTranslatef(0.0,2.0,0.0);
gluLookAt(0.0,10.0,0.0,1.0,0.0,0.0,0.0,0.0,3.0);
glRotatef((GLfloat)b,0.0,0.0,0.0);  glScalef(200.0,0.0,0.0);
glColor3f(4.3,3.5,1.0);  glutSolidSphere(0.04,20,8);
glPopMatrix();  glPushMatrix();  glTranslatef(0.0,-4.0,0.0);
gluLookAt(0.0,10.0,0.0,1.0,0.0,0.0,0.0,0.0,3.0);
glRotatef((GLfloat)b,0.0,0.0,0.0);  glScalef(200.0,0.0,0.0);
glColor3f(4.3,3.5,1.0);  glutSolidSphere(0.04,20,8);
glPopMatrix();  glPushMatrix();  glTranslatef(0.0,4.0,0.0);
gluLookAt(0.0,10.0,0.0,1.0,0.0,0.0,0.0,0.0,3.0);
glRotatef((GLfloat) b,0.0,0.0,0.0);
glScalef(200.0,0.0,0.0);  glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.1,20,8);  glPopMatrix();
glPushMatrix();  glTranslatef(0.0,-6.0,0.0);
gluLookAt(0.0,10.0,0.0,1.0,0.0,0.0,0.0,0.0,3.0);
glRotatef((GLfloat) b,0.0,0.0,0.0);
glScalef(200.0,0.0,0.0);  glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.1,20,8);  glPopMatrix();
glPushMatrix();  glTranslatef(0.0,6.0,0.0);
gluLookAt(0.0,10.0,0.0,1.0,0.0,0.0,0.0,0.0,3.0);
glRotatef((GLfloat) b,0.0,0.0,0.0);
glScalef(200.0,0.0,0.0); glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.1,20,8);  glPopMatrix();
glPushMatrix();  glTranslatef(0.0,-8.0,0.0);
gluLookAt(0.0,10.0,0.0,1.0,0.0,0.0,0.0,0.0,3.0);
glRotatef((GLfloat) b,0.0,0.0,0.0);
glScalef(200.0,0.0,0.0);  glColor3f(4.3,3.5,1.0);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);  glPopMatrix();
glPushMatrix();  glTranslatef(0.0,0.0,0.0);
gluLookAt(0.0,10.0,2.0,1.0,0.0,0.0,0.0,0.0,1.0);
glRotatef((GLfloat)b,0.0,0.0,0.0);
glScalef(200.0,0.0,0.0);  glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);  glPopMatrix();
glPushMatrix();  glTranslatef(0.0,-1.0,0.0);
gluLookAt(0.0,10.0,2.0,1.0,0.0,0.0,0.0,0.0,1.0);
glRotatef((GLfloat)b,0.0,0.0,0.0);
glScalef(200.0,0.0,0.0);  glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);  glPopMatrix();
glPushMatrix();  glTranslatef(0.0,1.0,0.0);
gluLookAt(0.0,10.0,2.0,1.0,0.0,0.0,0.0,0.0,1.0);
glRotatef((GLfloat)b,0.0,0.0,0.0);
glScalef(200.0,0.0,0.0);  glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);  glPopMatrix();
glPushMatrix();  glTranslatef(0.0,2.0,0.0);
```

```
gluLookAt(0.0,10.0,2.0,1.0,0.0,0.0,0.0,0.0,1.0);
glRotatef((GLfloat)b,0.0,0.0,0.0);
glScalef(200.0,0.0,0.0);  glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);  glPopMatrix();
glPushMatrix();  glTranslatef(8.7,9.0,0.0);
gluLookAt(0.0,10.0,2.0,1.0,0.0,0.0,0.0,0.0,1.0);
glRotatef((GLfloat)b,1.0,7.0,5.0);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);  glPopMatrix();
 glutSwapBuffers();
 }  void reshape(int
w,int h)
{
glViewport(0,0,(GLsizei)w,(GLsizei)h);
glMatrixMode(GL_PROJECTION);
 glLoadIdentity();
gluPerspective(60.0,(GLfloat)w/(GLfloat)h,1.0,20.0);
glMatrixMode(GL_MODELVIEW);
 glLoadIdentity();
gluLookAt(0.0,0.0,5.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0);
}
 void keyboard(unsigned char key,int x,int y)
 {
switch(key)
{  case 'z':
z=(z+50)%360;
glutPostRedisplay();
break;  case 'm':
m=(m+3)%360;
glutPostRedisplay();
break;  case 'M':
M=(M+12)%360;
glutPostRedisplay();
break;  case 'v':
v=(v+2)%360;
 glutPostRedisplay();
break;  case 'V':
V=(V+10)%360;
 glutPostRedisplay();
break;  case 'e':
e=(e+5)%360;
 glutPostRedisplay();
break;  case 'E':
E=(E+8)%360;
 glutPostRedisplay();
break;  case 'r':
r=(r+6)%360;
 glutPostRedisplay();
break;  case 'R':
R=(R+6)%360;
glutPostRedisplay();
break;
'j':j=(j+10)%360;
glutPostRedisplay
break;  case 'J':
J=(J+4)%360;
 glutPostRedisplay();
break;  case
's':s=(s+9)%360;
glutPostRedisplay();
break;  case
'S':S=(S+3)%360;
glutPostRedisplay();
break;  case
'u':u=(u+8)%360 ;
glutPostRedisplay();
break;  case
'U':U=(U+2)%360;
glutPostRedisplay();
break;  case
'n':n=(n+7)%360;
glutPostRedisplay();
```

```
 break;  case
'N':N=(N+1)%360
glutPostRedisplay();
break;  case
'b':b=(b+10)%360
glutPostRedisplay();
break;  case
'c':c=(c+1)%360;
b=(b+10)%360;
glutPostRedisplay();
break;  case
'X':X=(X+5)%360;
glutPostRedisplay();
break;
 case 'a':z=(z+50)%360;  b=(b+10)%360;
m=(m+3)%360;
v=(v+2)%360;
e=(e+5)%360;
r=(r+6)%360;
j=(j+10)%360;
s=(s+9)%360;
u=(u+8)%360;
n=(n+7)%360;
c=(c+1)%360;
glutPostRedisplay();
break;  case
'A':z=(z+50)%360;
 b=(b+10)%360;  M=(M+12)%360;
 V=(V+10)%360;
E=(E+8)%360;
R=(R+6)%360 J=(J+4)%360;
S=(S+3)%360;
 U=(U+2)%360;
N=(N+1)%360;
c=(c+1)%360;
glutPostRedisplay();
break;  case
'B':z=(z+50)%360;
b=(b+10)%360;
c=(c+1)%360;
m=(m+3)%360
;M=(M+12)%360;
v=(v+2)%360;
V=(V+10)%360;
e=(e+5)%360;
E=(E+8)%360;
r=(r+6)%360;
R=(R+6)%360;
j=(j+10)%360;
J=(J+4)%360;
s=(s+9)%360;
S=(S+3)%360;
u=(u+8)%360;
U=(U+2)%360;
n=(n+7)%360;
N=(N+1)%360;
 glutPostRedisplay();
break;  case
27:exit(0);  break;
default:break;  }
void mouse(int btn ,int state,int x,int y)
 {
 if(btn==GLUT_LEFT_BUTTON && state==GLUT_DOWN)
{
z=(z+50)%360; b=(b+10)%360;
c=(c+1)%360;  m=(m+3)%360;
M=(M+12)%360;
v=(v+2)%360;
V=(V+10)%360;
e=(e+5)%360; E=(E+8)%360;
r=(r+6)%360; R=(R+6)%360;
```

```
j=(j+10)%360; J=(J+4)%360;
s=(s+9)%360; S=(S+3)%360;
u=(u+8)%360; U=(U+2)%360;
n=(n+7)%360; N=(N+1)%360;
 glutPostRedisplay();
}
if(btn==GLUT_MIDDLE_BUTTON && state==GLUT_DOWN)
 {
z=(z+50)%360;
b=(b+10)%360;  c=(c+1)%360;
m=(m+3)%360;
M=(M+12)%360;  v=(v-
2)%360; V=(V-10)%360;
e=(e+5)%360; E=(E+8)%360;
r=(r-6)%360; R=(R-6)%360;
j=(j+10)%360; J=(J+4)%360;
s=(s-9)%360; S=(S-3)%360;
u=(u+8)%360; U=(U+2)%360;
n=(n-7)%360; N=(N-1)%360;
glutPostRedisplay();
 }
 if(btn==GLUT_RIGHT_BUTTON && state==GLUT_DOWN)
 {
 z=(z-50)%360;  b=(b-
10)%360;  c=(c+1)%360;
m=(m-3)%360; M=(M-
12)%360;  v=(v-2)%360;
V=(V-10)%360;  e=(e-
5)%360; E=(E-8)%360;
r=(r-6)%360; R=(R-
6)%360;  j=(j-10)%360;
J=(J-4)%360;  s=(s-
9)%360; S=(S-3)%360;
u=(u-8)%360; U=(U-
2)%360;  n=(n-7)%360;
N=(N-1)%360;
 glutPostRedisplay();
 }
 }
 int main(int argc,char **argv)
 {
glutInit(&argc,argv);
 glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
glutInitWindowSize(500,500);  glutInitWindowPosition(100,100);
glutCreateWindow("planets amidst stars");
 myinit();
glutDisplayFunc(display);
glutReshapeFunc(reshape);
glutKeyboardFunc(keyboard);
glutMouseFunc(mouse);
glEnable(GL_DEPTH_TEST);
glutMainLoop();  return 0; }
```