VISVESVARAYA TECHNOLOGICAL UNIVERSITY
Belagavi-590018, Karnataka

A

MINI PROJECT REPORT

ON

"SOLAR SYSTEM"

Submitted in partial fulfillment for the award of the degree in **Bachelor of Engineering in Computer Science & Engineering**

Submitted by

SHISHEER OLI	**[1BH19CS101]**
KHEM RAJ BAJGAIN	**[1BH20CS016]**
PRABIN KUMAR SAH	**[1BH20CS023]**

Under the Guidance of

Mrs. MONICA C
Assistant Professor
Department of Computer Science and Engineering

BANGALORE TECHNOLOGICAL INSTITUTE
(ISO 9001:2015 Certified Institute) Sarjapur Road, Kodathi, Bangalore-560035

Department of Computer Science & Engineering

2022-2023

BANGALORE TECHNOLOGICAL INSTITUTE
Bengaluru-35
(An ISO 9001:2015 Certified Institute)
Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Mini project report on "**SOLAR SYSTEM**" carried out by **SHISHEER OLI [1BH19CS101]**, **KHEM RAJ BAJGAIN [1BH20CS016]** and **PRABIN KUMAR SAH [1BH20CS023]** the Bonafide students of **Bangalore Technological Institute**, **Bangalore** in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of **Visvesvaraya Technological University, Belagavi** during the year **2022-2023**. Thus, it is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report. The mini project report has been approved, as it satisfies the academic requirement in the respect of the mini project report prescribed for the said degree.

Mrs. Monica C
Assistant Professor
Department of CSE

Dr. Sohan Kumar Gupta

H.O.D, Department of CSE

External Viva

Name of the Examiners	**Signature with date**
1..........................	....
2..........................	....

BANGALORE TECHNOLOGICAL INSTITUTE
Bengaluru-35
(An ISO 9001:2015 Certified Institute)
Department of Computer Science and Engineering

DECLARATION

We, the students of sixth semester **B.E, COMPUTER SCIENCE AND ENGINEERING, BANGALORE TECHNOLOGICAL INSTITUTE, BANGALORE,** hereby declare that the Mini project on "**SOLAR SYSTEM**" has been independently carried out by me at **Bangalore Technological Institute, Bengaluru** and submitted in partial fulfillment of the requirements for the award of the degree in **Bachelor of Engineering in Computer Science & Engineering** of the **Visvesvaraya Technological University, Belagavi** during the academic year **2022-2023**.

We also declare that, to the best of my knowledge and believe the work reported here does not form or part of any other dissertation on the basis of which a degree or award was conferred on an early occasion of this by any other students.

PLACE: BENGALURU

DATE:

SHISHEER OLI [1BH19CS101]
KHEM RAJ BAJGAIN [1BH20CS016]
PRABIN KUMAR SAH [1BH20CS023]

ACKNOWLEDGEMENT

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals and elders. I would like to take this opportunity to thank them all.

We heartily extend my words of gratitude to our Mini project Coordinator , **Mrs. Monica C**, for her valuable advice, encouragement and suggestion given to me in the course of my Mini project work. We convey gratitude to her for having constantly monitored the development of the seminar report and setting up precise deadlines.

We would like to express my immense gratitude to the Head of Department **Dr. Sohan Kumar Gupta**, for his unfailing encouragement and suggestions given to me in the course of the work.

We would like to take this opportunity to express my gratitude to the Principal, **Dr. H S Nanda**, for giving me this opportunity to enrich knowledge. We am grateful to the President **Dr. A Prabhakara Reddy** and Secretary, **Sri. C L Gowda** for having provided us with a great infrastructure and well-furnished labs.

Finally, a note of thanks to the Department of Computer Science and Engineering,both teaching and non-teaching staff for their cooperation extended.

Last but not the least, We acknowledge the support and feedback of my parents, guardians and friends, for their indispensable help always.

SHISHEER OLI [1BH19CS101]
KHEM RAJ BAJGAIN [1BH20CS016]
PRABIN KUMAR SAH [1BH20CS023]

ABSTRACT

"SOLAR SYSTEM" is a 2D representation of planets. The solar system is a group of objects that revolve around the Sun. It includes the Sun, eight planets, moons, asteroids, and comets. The planets have different characteristics. Earth is our home and has life. Some planets have moons. Asteroids are rocky objects, and comets are icy objects. Scientists study the solar system to learn about the universe. It's important to understand and appreciate the interconnectedness of these celestial bodies. **Solar System** is created using OpenGL.

CONTENTS

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

Computer graphics is conscerned with all aspects of producing picture or an image using computers and, more generally, the representation and manipulation of pictorial data by a computer. The development of computer graphics has made computers easier to interact with and better for understanding and interpreting many types of data. Developments in computer graphic had a profound impact on many types of media and have revolutionized the animation and video game industry. Today computers and computer-generated images touch many aspects of our daily life. Computer imagery is found on television, in newspapers, in weather reports, and during surgical procedures. A well-constructed graph can present complex statistics in a form that is easier to understand and interpret. Such graphs are used to illustrate papers, reports, theses, and other presentation material. A range of tools and facilities are available to enable users to visualize their data, and computer graphics are used in many disciplines. We implement computer graphics using the OpenGL API. 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.

OpenGL is a low-level graphics library specification. It makes available to the programmer a small set of geometric primitives - points, lines, polygons, images, and bitmaps. OpenGL provides a set of commands that allow the specification of geometric objects in two or three dimensions, using the provided primitives, together with commands that control how these objects are rendered (drawn). Since OpenGL drawing commands are limited to those that generate simple geometric primitives (points, lines, and polygons), the OpenGL Utility Toolkit (GLUT) has been created to aid in the development of more complicated three-dimensional objects such as a sphere, a torus, and even a teapot. GLUT may not be satisfactory for full- featured OpenGL applications, but it is a useful starting point for learning OpenGL. GLUT is designed to fill the need for a window system independent programming

Interface for OpenGL programs. The interface is designed to be simple yet still meet the needs of useful OpenGL programs. Removing window system operations from OpenGL is a sound decision because it allows the OpenGL graphics system to be retargeted to various Systems including powerful but expensive graphics workstations as well as mass-production graphics systems like video games, set-top boxes for interactive television, and PCs.

The GLUT application-programming interface (API) requires very few routines to display a graphics scene rendered using OpenGL. The GLUT routines also take relatively few parameters.

1.1COMPUTER GRAPHICS

The term computer graphics includes almost everything on computers that is not text or sound. Today nearly all computers use some graphics and users expect to control their computer through icons and pictures rather than just by typing. The term Computer Graphics has several meanings the representation

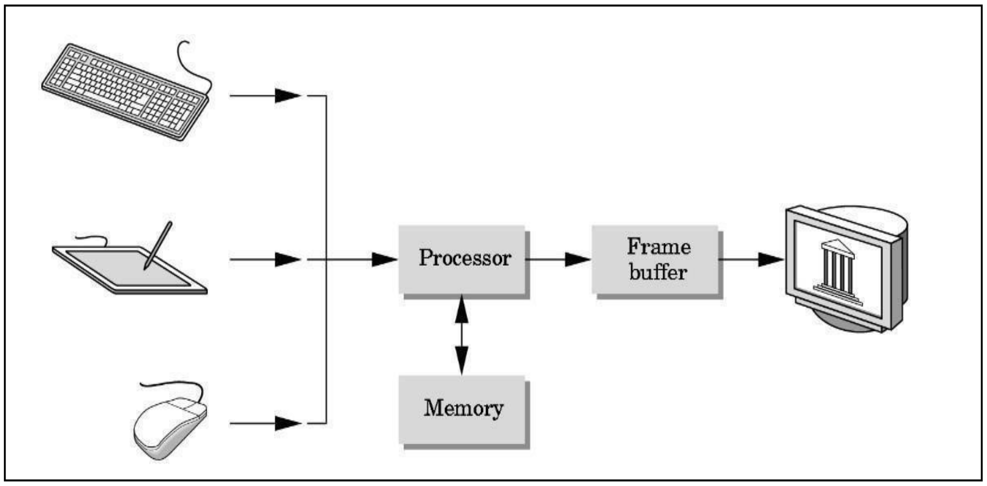

Fig 1.1 A Graphics System

and manipulation of pictorial data by a computer the various technologies used to create and manipulate such pictorial data the images so produced, and the sub-field of computer science which studies methods for digitally synthesizing and manipulating visual content today computers and computer-generated images touch many aspects of our daily life.

Computer imagery is found on television, in newspapers, in weather reports, and during surgical procedures. A well-constructed graph can present complex statistics in a form that is easier to understand and interpretation fig 1.1 inputs go to processor from there to frame buffer and then it displays the pixel. It illustrates the graphics system. A graphics system is just the graphics part of the software. that sits in between the hardware and application programs.

CHAPTER 2

LITERATURE SURVEY

2.1 OPENGL TECHNOLOGY

OpenGL is strictly defined as “a software interface to graphics hardware.” In essence, it is a 3D graphics and modeling library that is highly portable and very fast. Using OpenGL, you can create elegant and beautiful 3D graphics with exceptional visual quality. The greatest advantage to using OpenGL is that it is orders of magnitude faster than a ray tracer or software- rendering engine. Initially, it used algorithms carefully developed and optimized by Silicon Graphics, Inc. (SGI), an acknowledged world leader in computer graphics and animation.

1. Vertex processing: Much of the work in the pipeline is in converting object representations from one co-ordinate system to another. Every vertex is processed independently.

The co-ordinate system include the following:

- Object co-ordinates
- Camera co-ordinates
- Screen co-ordinates.

2. Clipping and Primitive assembly: clipping is necessary, because of the limitation that no imagining system can see the whole object at once. Cameras have film of limited size. Image outside the clipping volume is clipped out. Clipping is done by vertex basis.
3. Rasterization: generation of pixels in the frame buffer. Rasterizer determines which pixels in the frame buffer are inside the polygon. The output of the Rasterizer isa set of fragments for each primitive. Fragments carry color, location, depth and other information.
4. Fragment processing: Updates the pixel in the frame buffer. Color of the pixels that correspond to fragment can be read from the frame buffer. Color fragment may bealtered by texture mapping.

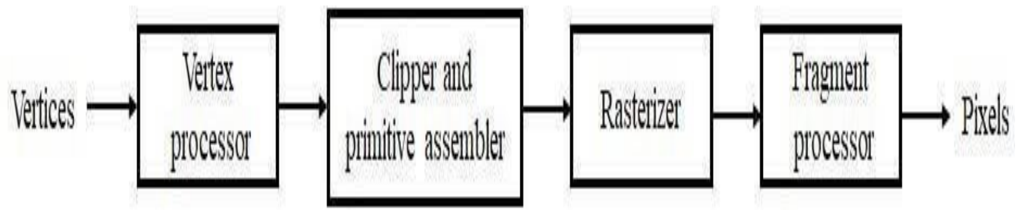

Fig 2.1 Graphics pipeline.

A computer graphics pipeline, rendering pipeline, is a conceptual model that describes what steps a graphics system needs to perform to render a 3D scene to a 2D screen. The graphics pipeline as four stages. These stages include vertex processing, clipping, rasterization.

2.2 OPENGL FUNDAMENTALS

The section explains some of the concepts inherent in OpenGL. Primitives and Commands OpenGL draws primitives-points, line segments, or polygons-subject to several selectable modes. You can control modes independently of each other; that is, setting one mode doesn't affect whether other modes are set (although many modes may interact to determine what eventually ends up in the frame buffer).Primitives are specified, modes are set, and other OpenGL operations are described by issuing commands in the form of function calls. Primitives are defined by a group of one or more vertices. The interface consists in more than 250 different functions, which can be used to draw complex tridimensional scenes with simple primitives. It consists of many functions that help to create a real world object and a particular existence for an object can be given.

2.3 PROJECT DEFINITION

The project “SOLAR SYSTEM” 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!”

Keys used

Mouse is used to rotate the whole solar system. Double tab is used to rotate the solar system and releasing the mouse stops the movement/rotation of the solar system.

2.4 ADVANTAGES

2.2.1 User-friendly.

2.2.2 Simple and interactive.

2.2.3 Multiple actions can be performed at once.

2.2.4 OpenGL is more functional than any other API

2.2.5 Reliability and Portability.

CHAPTER 3

SYSTEM REQUIREMENTS SPECIFICATION

3.1 Minimum hardware requirements

- **Microprocessor:** 1.0 GHz and above CPU based on either AMD or INTEL Microprocessor Architecture.
- **Main memory:** 512 MB RAM.
- **Hard Disk :** 100MB.
- **Hard disk speed in RPM:** 5400 RPM.
- **Keyboard:** QWERTY Keyboard.
- **Mouse:** 2 or 3 Button mouse.
- **Monitor:** 1024 x 768 display resolution.

3.2 Minimum software requirements

- Operating system: Windows XP and later.
- Visual C++ 2005 and later.
- OPENGL Library.
- X86.
- X64(WOW).
- Mouse Driver.
- Graphics Driver.
- C Libraries.

CHAPTER 4

DESIGN

4.1 SYSTEM DESIGN

The mini project is designed to how graphics can be used to represent data. It is a graphics package encoded in Microsoft Visual C++ with OpenGL and the mini project basically deals with providing the graphical representation of data that a user provides as statistics to the system.

The above mini project can be analyzed as follows:

- The mini project is implemented using ‘C++’ and ‘OpenGL’ built in functions.
- In the beginning the user is asked to enter the statistical data through the windows command line.
- Once the data is input, the computations are done by program to compute various co- ordinates etc. and appropriate graph as per users choice is rendered on the display window.
- This project is developed using Microsoft Visual C++ with OpenGL as an academic project using keyboard and mouse. This project is implemented by making use of extensive use of library functions offered by graphic package of ‘OpenGL’. A list of standard library functions that are used are given below. A number of user defined functions also have been used and a summary of those functions follows this list of standard library functions.
- The aim of the project is to show how the a space shooter can be implemented using computer graphics. The various functions used to implement it are attractive.

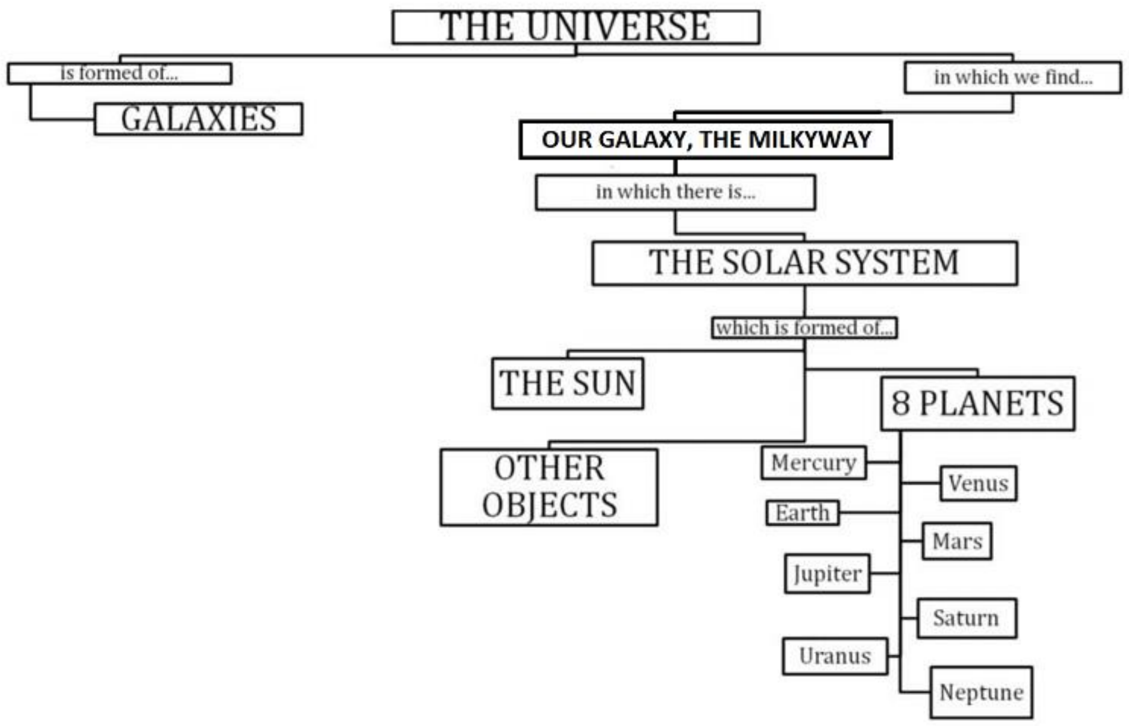

Fig 4.1 DataFlow Diagram.

The dataflow diagram which shows the flow of data within the program from the start to the different functions within the program, where each function will perform its specified actions for the proper display of the graphics and functionality defined within the program.

CHAPTER 5

IMPLEMENTATION

5.1 INTRODUCTION TO VISUAL STUDIO

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs for Microsoft Windows super family of operating systems, as well as websites, web applications and web services. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code.

Visual Studio includes a code editor supporting IntelliSense as well as code refactoring. The integrated debugger works both as a source- level debugger and a machine-level debugger. Other built-in tools include a forms designer for building GUI applications, web designer, class designer, and database schema designer. It accepts plug-ins that enhance the functionality at almost every level—including adding support for source-control systems (like Subversion) and adding new toolsets like editors and visual designers for domain-specific languages or toolsets for other aspects of the software development lifecycle (like the Team Foundation Server client: Team Explorer).

Visual Studio supports different programming languages and allows the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages include C, C++ and C++/CLI (via Visual C++), VB.NET (via Visual Basic .NET), C# (via Visual C#), and F# (as of Visual Studio 2010).Support for other languages such as M, Python, and Ruby among others is available via language services installed separately.

5.2 ALGORITHM

1. Initialize the display mode;
2. Initialize the window position and size;
3. Create display window;
4. Initialize the keyboard function;
5. Call display function;

5.3 OPENGL FUNCTIONS

The functions used in the programs are as follows:

- **main( ):** The control will first enter into this function and in this function it will create a window and it will initialize the window and then this function calls display function.
- **glutInit() :** interaction between the windowing system and OPENGL is initiated
- **glutInitDisplayMode() :** used when double buffering is required and depthinformation isrequired
- **glutCreateWindow() :** this opens the OPENGL window and displays the title at topof thewindow
- **glutInitWindowSize() :** specifies the size of the window
- **glutInitWindowPosition() :** specifies the position of the window in screen co-ordinates
- **glutKeyboardFunc() :** handles normal ASCII symbols
- **glutSpecialFunc() :** handles special keyboard keys
- **glutReshapeFunc() :** sets up the callback function for reshaping the window
- **glutIdleFunc() :** this handles the processing of the background
- **glutDisplayFunc() :** this handles redrawing of the window
- **glutMainLoop() :** this starts the main loop, it never returns

- **glViewport() :** used to set up the viewport
- **glVertex3fv() :** used to set up the points or vertices in three dimensions
- **glColor3fv() :** used to render color to faces
- **glFlush() :** used to flush the pipeline
- **glutPostRedisplay() :** used to trigger an automatic redrawal of the object
- **glMatrixMode() :** used to set up the required mode of the matrix
- **glLoadIdentity() :** used to load or initialize to the identity matrix

5.4 Source Code

```
#include<stdio.h>
#include<stdlib.h>
#include<GL/glut.h>
#include<math.h>
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);
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);
glutSolidSphere(0.1,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(8.0,0.0,0.0);
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(-8.0,-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(6.0,4.0,0.0);
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(-6.0,4.0,0.0);
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(5.0,-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(-7.0,3.0,0.0);
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(-7.0,2.0,0.0);
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(7.0,-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(7.0,-3.0,0.0);
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(-7.0,-3.0,0.0);
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(7.0,2.0,0.0);
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(1.0,-7.0,0.0);
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(2.0,-5.0,0.0);
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(5.0,-1.0,0.0);
gluLookAt(0.0,10.0,0.0,1.0,0.0,0.0,0.0,0.0,1.0);
glRotatef((GLfloat) b,0.0,0.0,0.0);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.07,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(-6.0,7.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.07,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(-0.5,3.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.07,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(-1.5,4.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.07,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(-9.0,3.0,0.0);
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);
```

```
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.07,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(9.0,-5.9,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.1,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(6.5,8.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.1,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(5.0,7.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.1,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(-9.0,6.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.1,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(-10.5,9.0,0.0);
```

```
gluLookAt(0.0,10.0,2.0,1.0,0.0,0.0,0.0,0.0,1.0);
glRotatef((GLfloat) b,0.0,0.0,0.0);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.07,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(-11.0,-7.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.07,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(-11.0,5.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.05,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(-7.0,-5.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(-10.0,3.7,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);
glPopMatrix();
```

```
glPushMatrix();
glTranslatef(-7.0,-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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(-8.0,5.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.03,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(-8.0,-5.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.05,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(-11.0,-4.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(6.0,-5.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
```

```
glutSolidSphere(0.04,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(-6.9,7.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(5.0,-4.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.05,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(6.0,4.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(3.0,-4.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(4.0,-7.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.05,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(-4.0,-3.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);
glPopMatrix();
 glPushMatrix();
glTranslatef(4.0,-7.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);
glPopMatrix();
glPushMatrix();
 glTranslatef(11.0,-4.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.05,20,8);
glPopMatrix();
glPushMatrix();
 glTranslatef(9.0,-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,0.0,0.0,0.0);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);
glPopMatrix();
glPushMatrix();
```

```
glTranslatef(8.0,-4.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.05,20,8);
glPopMatrix();
glPushMatrix();
 glTranslatef(9.0,5.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef (7.0,7.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(0.9,7.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(1.0,6.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);
```

```
glPopMatrix();
glPushMatrix();
glTranslatef(0.8,-5.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(3.0,-7.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.05,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(1.0,5.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(2.0,4.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(0.0,7.0,0.0);
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);
```

```
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.05,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(0.0,5.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(-9.0,0.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.05,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(-10.0,4.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(9.0,3.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.05,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(-10.0,-6.0,0.0);
```

```
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(10.0,0.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.05,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(-9.0,-7.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(-3.0,4.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.05,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(-9.9,-7.5,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);
glPopMatrix();
```

```
glPushMatrix();
glTranslatef(1.0,5.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.05,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(3.0,6.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(-2.0,-5.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.05,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(-3.0,-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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(-4.0,-8.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
```

```
glutSolidSphere(0.05,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(8.3,-7.1,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(-10.0,7.6,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.05,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(-3.0,7.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(-1.4,7.5,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.05,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(3.0,6.5,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(-6.0,4.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(-6.0,-6.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.05,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(0.7,4.0,0.0);
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);
glColor3f(4.3,3.5,1.0);
glutSolidSphere(0.04,20,8);
glPopMatrix();
glPushMatrix();
glTranslatef(2.0,2.0,0.0);
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(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(0.0,0.0,5.0,0.0,0.0,0.0,0.0,1.0,0.0);
}
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;
case '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;
}
```

CHAPTER 6

SAMPLE OUTPUT

6.1 SNAPSHOTS

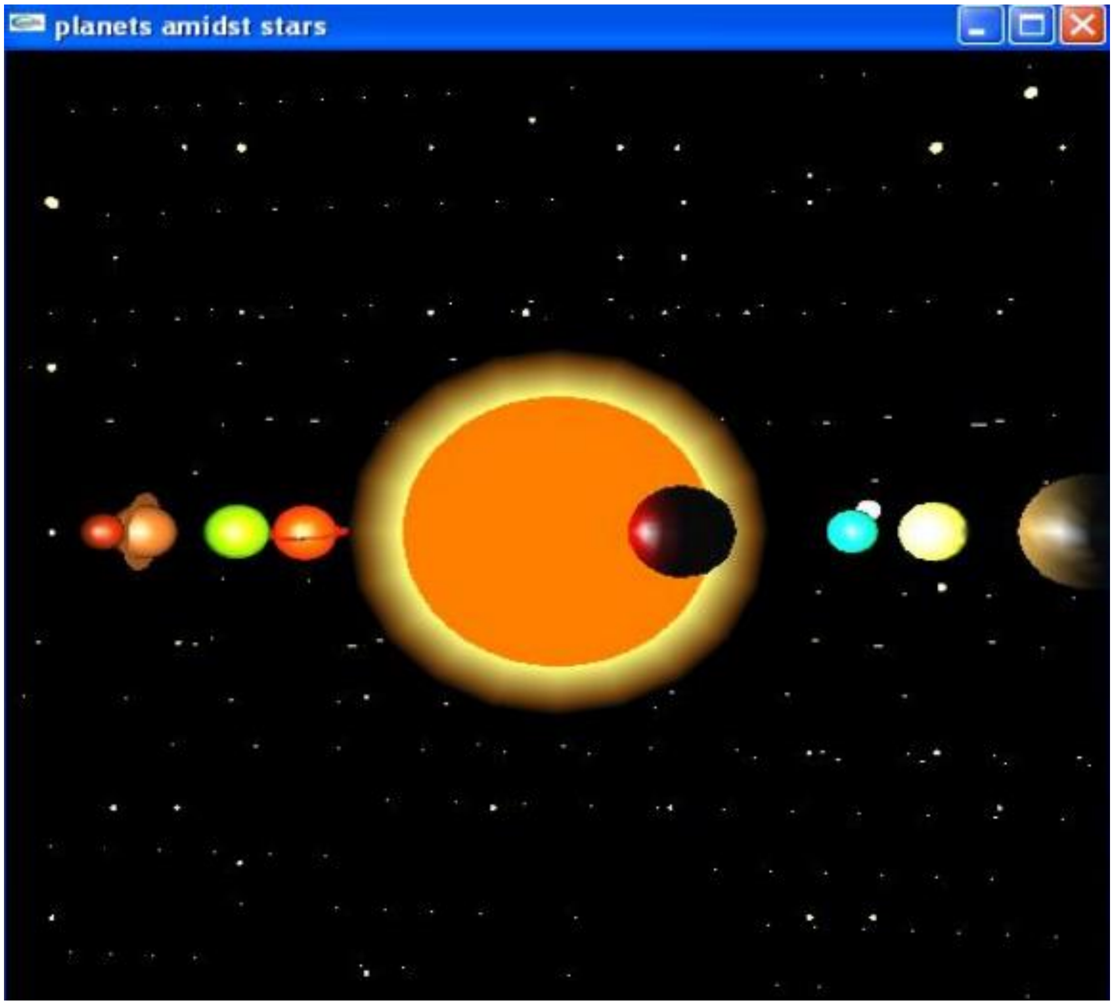

Fig 6.1 Solar system with revolution of mercury.

Here we can see the sun at the center and all planets revolving around the sun.

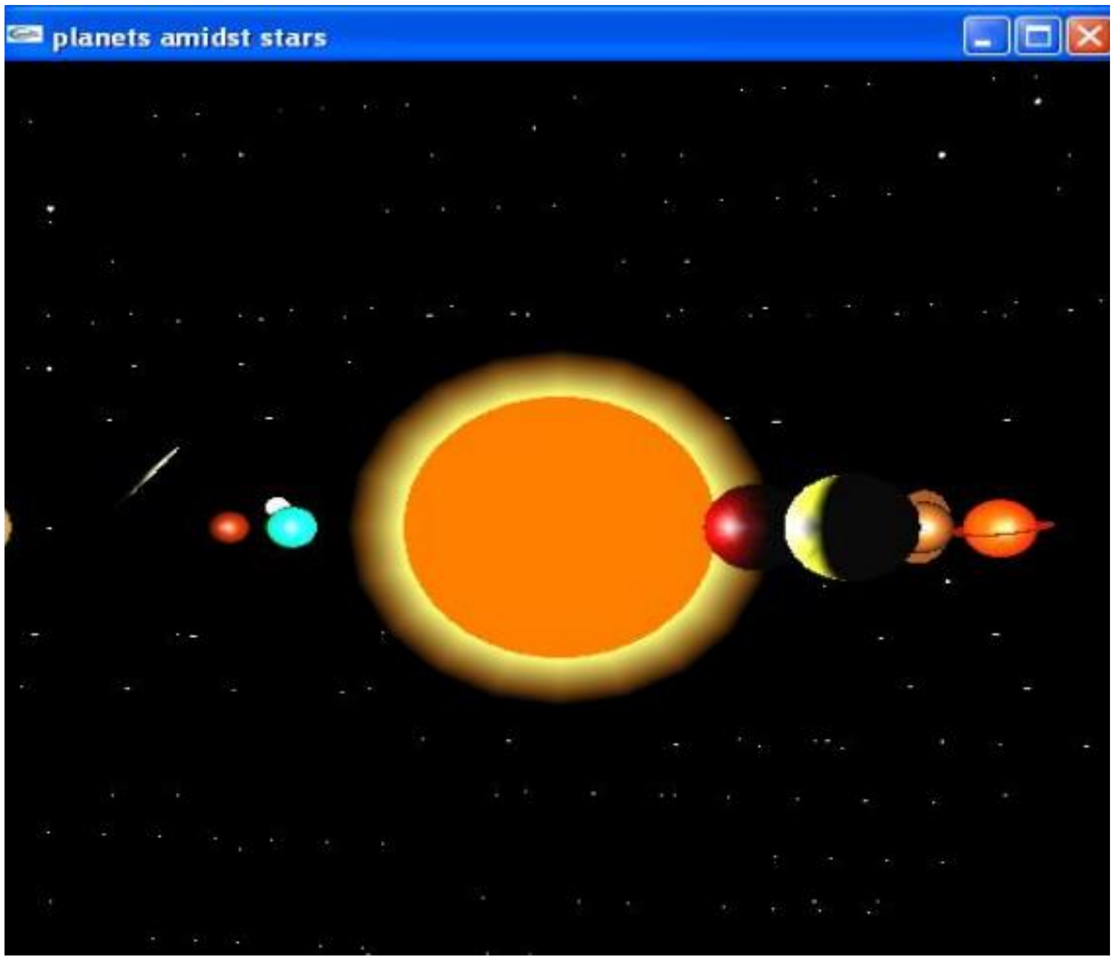

Fig 6.2 Solar system with revolution of planets and comet.

In this snapshot, we can see sun at the center and eight planets revolving around the sun.

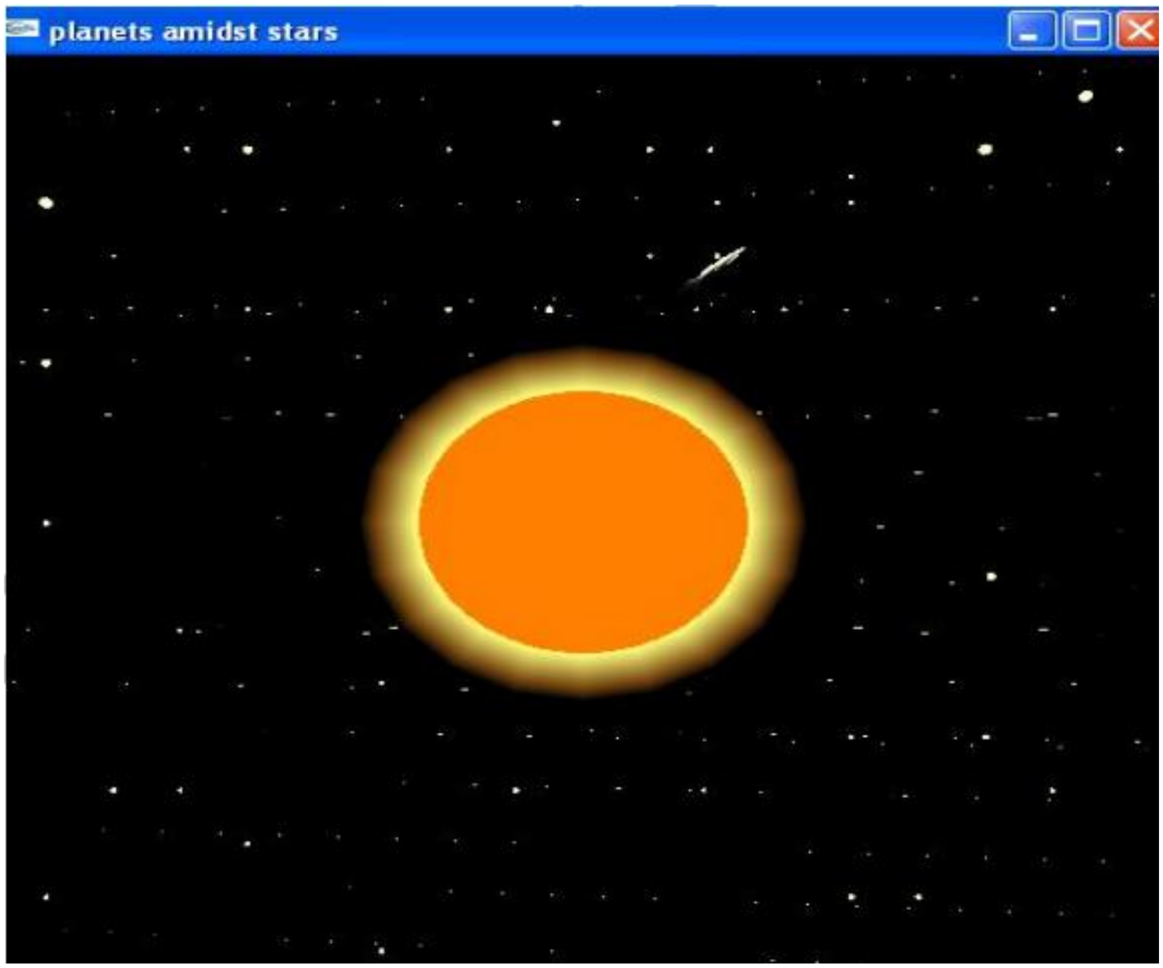

Fig 6.3 Sun twinkling stars and comet

In this snapshot we can the sun that is imagined to be placed at the center and stars.

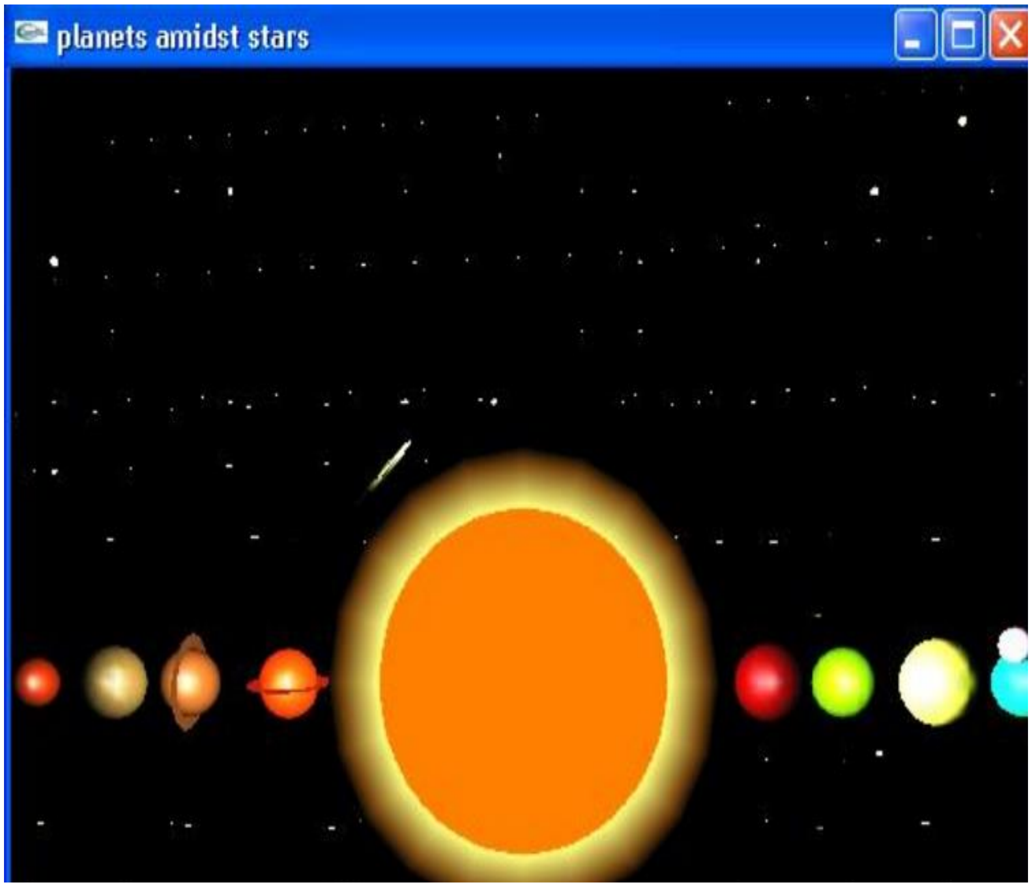

Fig 6.4 View of solar system.

In this snapshot, sun is placed at the center and eight planets are placed in the sun’s orbit.

CONCLUSION AND FUTURE ENHANCEMENTS

CONCLUSION

"**SOLAR SYATEM**" is designed and implemented using a graphics software system called as **OpenGL** which has became a widely accepted standard for developing graphic application. Using OpenGL functions user can create geometrical objects and can use **translation, rotation, scaling** with respect to the co-ordinate system. The development of this project has enabled us to improve accuracy, problem solving skills while providing a fun and interactive experience to the player.

FUTURE ENHANCEMENT

The project can be further enhanced by adding some of the features using OpenGL in future. The project can be further implemented by including the background effect and translate command along with the Mouse interactions been given. The project can further be implemented by other modification if required.

BIBLIOGRAPHY

BOOK

- Edward Angel's Interactive Computer Graphics Pearson Education 5th Edition
- Interactive computer Graphics --A top down approach using open GL—by EdwardAngel.
- Jackie.L.Neider,MarkWarhol,Tom.R.Davis,"OpenGLRed Book",Second. Revised Edition, 2005
- Donald D Hearn and M.PaulineBaker,"Computer Graphics with OpenGL", 3rd Edition.

WEBSITES

- www.OpenGL Redbook.com
- www.OpenGL simple examples.
- www.OpenGL programming guide.
- http://www.wikipedia.com
- http://basic4gl.wikispaces.com
- http://www.opengl.org