Bangalore University

UNIVERSITY VISVESVARAYA COLLEGE OF ENGINEERING

K R Circle Bengaluru – 560001

Department of Computer Science and Engineering

MINI PROJECT

On

GRAPHICAL IMPLEMENTATION ON SOLAR SYSTEM

Submitted by

Kavya Bhat (19GANSE021) **Krishna Shenoy S (19GANSE023)**

5th Semester B.Tech (ISE)

Under the Guidance of

Dr. H S Vimala
Professor
Dept of CSE, UVCE

Manjula S
Guest faculty
Dept of CSE, UVCE

Jan-2022

Bangalore University

UNIVERSITY VISVESVARAYA COLLEGE OF ENGINEERING

K R Circle Bengaluru – 560001

Department of Computer Science and Engineering

CERTIFICATE

This is to certify **Kavya Bhat(19GANSE021)** and **Krishna Shenoy S(19GANSE023), 5th Sem B.Tech (ISE)** have successfully completed the Mini Project work on **"Solar Sytem"** prescribed by the **Bangalore University** during the academic year 2021-22.

Dr. H S Vimala
Professor
Dept. of CSE, UVCE,
Bangalore

Dr. Dilip Kumar S M
Professor & Chairman
Dept of CSE, UVCE,
Bangalore

Examiner 1 …….

Examiner 2…….

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned our effort with success.

We express our sincere gratitude to University Visvesvaraya College of Engineering Bangalore, for having given us the opportunity to carry out this project. We also like to thank our Honourable Vice Chancellor Dr. Venugopal K R for giving us all the support to make this project successful.

We would like to thank our Principal Dr. H. N. Ramesh, and professor & Chairman Dr. Dilip Kumar S M for providing us all the facilities to work on this project.

We express our gratitude and thanks to Manjula S, Guest Faculty, Department of Computer Science and Engineering, for her support and encouragement.

We consider it as a privilege and an honour to express our sincere gratitude to our guide, Dr. H S Vimala. Professor, Dept. of CSE for her valuable guidance throughout the tenure of this project work, and her constant support and encouragement made this work successful.

We also thank non-teaching staff of Dept of CSE, our parents and our friends for their help, encouragement and support. Last but not the least, we thank God Almighty, without whose blessings this wouldn't have been possible.

Kavya Bhat (19GANSE021)

KrishnaShenoy S (19GANSE023)

ABSTRACT

This solar system model depicts a 3Dimensional model where it shows the basic working of the solar system.

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.

Free formed shapes used and different colours for each part. To get a precise with it requires the proper scaling and animations.

This project named “Solar System” uses OpenGL software interface and develops 2D images.

The project is based on Simple window coordinates and using recursive techniques in OpenGL.

CONTENTS

CHAPTER 1

INTRODUCTION

Computer Graphics is the use of computers to create and manipulate pictures on a display device. It comprises of software techniques to create, store, modify, represents pictures. It is difficult to display an image of any size on the computer screen. This method is simplified by using Computer graphics. Graphics on the computer are produced by using various algorithms and techniques. The end product of the computer graphics is a picture it may be a business graph, drawing, and engineering.

One of the application domains of Computer Graphics is Education and Training. Computer-generated model of the physical, financial and economic system is often used as educational aids which can help to understand the operation of the system. For some training applications, particular systems are designed. For example Flight Simulator. It helps in giving training to the pilots of airplanes having the advantages of fuel saving, safety, ability to familiarize the training with a large number of the world's airport.

Representation of uses of the Computer Graphics :

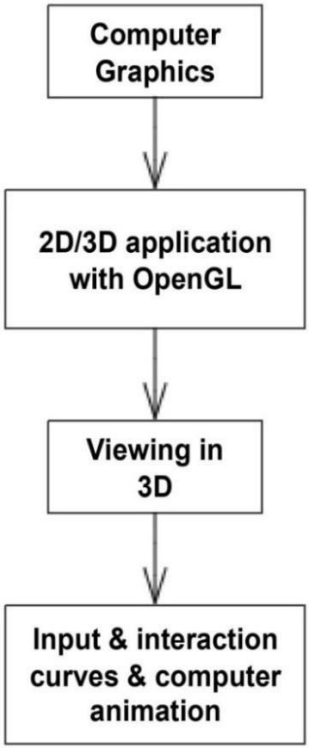


Fig 1.1 Applications of computer graphics

- Computer Graphics is the creation of pictures with the help of a computer.
- 2-Dimensional & 3-Dimensional applications can be created in Computer Graphics using OpenGL. In the 2D system, we use only two coordinates X and Y but in 3D, an extra coordinate Z is added. 3D graphics techniques and their application are fundamental to the entertainment, games, and computer-aided design industries.
- 3-Dimensional objects can be made & viewed in Computer Graphics. We can view an object from any spatial position, eg. In front of an object, Behind the object, In the middle of a group of objects, Inside an object, etc. - 3D descriptions of objects must be projected onto the flat viewing surface of the output device,
- In the field of computer graphics, interaction refers to the manner in which the application program communicates with input and output devices of the system. Interactive curves & their applications are extremely useful in a number of academic and industrial settings, and specifically play a significant role in multidisciplinary problem solving, such as in font design, designing objects, object recognition, etc. Animation refers to the movement on the screen of the display device created by displaying a sequence of still images. Animation is the technique of designing, drawing, making layouts and preparation of photographic series which are integrated into the multimedia and gaming products.

Graphics System & I/O devices

A Graphics system has 5 main elements Input Devices, Processor, Memory, Frame Buffer, Output Devices Pixels and the Frame Buffer. A picture is produced as an array (raster) of picture elements (pixels). These pixels are collectively stored in the Frame Buffer. Pixels in the frame buffer are displayed as points on the surface of the display.

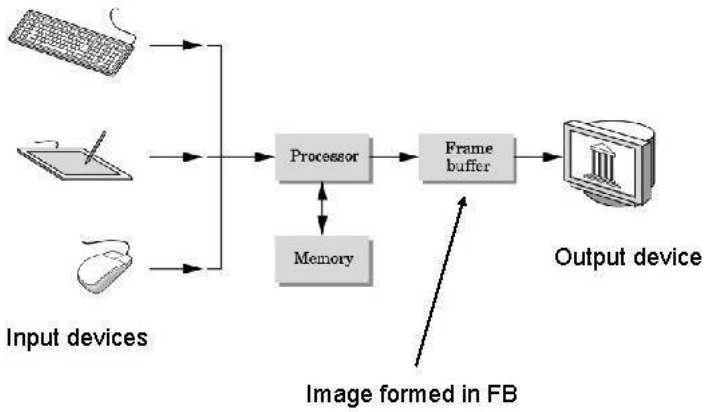


Fig 1.2 Basic Graphics system

The Input Devices are the hardware that is used to transfer transferred input to the computer. The data can be in the form of text, graphics, sound, and text. These Devices include Keyboard, Mouse, Trackball, Spaceball, Joystick, Light Pen, Digitizer, Touch Panels, Voice Recognition, Image Scanner. The output device displays data from the memory of the computer. Output can be text, numeric data, line, polygon, and other objects. The most predominant type of display has been the Cathode Ray Tube (CRT).

Motivation

- The Solar System is a beautiful model, which brings relaxing ambience.
- It has movement of planets and other space elements which provides a relaxing mind.
- It is also aesthetic and has beautiful appearance by its styling.
- All these features of Solar System took hold of our interest, out of which we decided to implement the Solar System used Computer Graphics.

Contribution

- We have created the Galaxy.
- We added an algorithm for the rotation of planets which needs to rotate around the Sun.
- Also created a menu which says what we can do in this.
- Made the change of the viewing pattern when the control buttons are pressed.

Objectives

- Developing a package using computer graphics with OpenGL.
- Migration from text editor to OpenGL.
- To show that implementation of Translation is easier with OpenGL.
- Implementing certain technical concept like Translation, motion, and use of the Idle Function.
- How to use Lightning effects used to produce computer animation.

Our CG Project

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 the constantly occurring motion of the planet and the stars just depicting the fact that "as passengers of the earth our voyage never ends!"

CHAPTER 2
HARDWARE & SOFTWARE REQUIREMENTS SPECIFICATIONS

The Hardware & Software requirements specification for this project are

Hardware requirements

- Pentium or higher processor.
- 16 MB or more RAM.
- A standard keyboard, and Microsoft compatible mouse
- VGA monitor.
- If the user wants to save the Created files a secondary storage medium can be Used.

Software requirements

- The graphics package has been designed for OpenGL;
- Machine must have CodeBlocks, minGW configured with freeglut.
- Software installed preferably 6.0 or later versions with mouse driver installed.
- Turbo c Libraries are used and hence a TC version 2 or later is required.
- Windows 10, Windows 11.

OpenGL (Open Graphics Library)

OpenGL is a software interface to graphics hardware. This interface consists of about 150 distinct commands that you use to specify the objects and operations needed to produce interactive three-dimensional applications.

OpenGL has become a widely accepted standard for developing graphics application. OpenGL is easy to learn, and it possesses most of the characteristics of

other popular graphics system. It is top-down approach. OpenGL 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 three-dimensional 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 Direct3D on Microsoft Windows platforms.

Function in the main GL library have name that begin with the letter gl and stored in the library. The second is the openGL utility Library (GLU). This library uses only GL function but contains codes for creating common object and viewing. Rather then using an different library for each system we used available library called openGL utility toolkit (GLUT). It used as #include<glut.h>

Using OpenGL

A common way to utilize the OpenGL graphics library of routines is by making calls to the routines from C programs. In order to utilize the OpenGL library, you will need to #include <GL/gl.h> in your C program. All OpenGL library functions begin with the letters **gl** and capitalize words (example functions: glClear, glClearColor, <codeglVertex2f). The OpenGL library provides a powerful but primitive set of rendering commands. Higher level libraries provide additional functionality. These include the following.

OpenGL Utility Library (GLU)

The GLU is included with OpenGL and built using low-level OpenGL commands. It contains routines for setting up viewing and projection matrices, polygonal tesselation and surface rendering. When using the GLU library you will need to #include <GL/glu.h> in your program. GLU routines begin with the prefix **glu**.

OpenGL Extension to the X Window System (GLX)

OpenGL does not include any routines for handling windowing operations so the GLX was developed to fill this gap with regard to the X Window System. Using GLX, you can enable your OpenGL programs to draw into a window under the X Window

System. GLX routines begin with the prefix **glx**.

OpenGL Utility Toolkit (GLUT)

The GLUT is a window system-independent toolkit written to hide the complexities of differing window system APIs. It is much easier to use and more portable but far less featured than the GLX library. Functions performed include window definition, window control, and monitoring of keyboard and mouse input. GLUT also has limited support for creating pop-up menus. If you are going to use the GLUT library you will need use #include <GL/glut.h> instead of #include <GL/gl.h> in your program (Note that glut.h includes gl.h, glu.h, and glx.h automatically). GLUT routines begin with the prefix **glut.**

In proposed system, the OpenGL is a graphic software system designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms. To achieve these qualities, no commands for performing windowing tasks or obtaining user input are included in OpenGL; instead, you must work through whatever windowing system controls the particular hardware you're using.

OpenGL doesn't provide high-level commands for describing models of three-dimensional objects. Such commands might allow you to specify relatively complicated shapes such as automobiles, parts of the body, airplanes, or molecules. With OpenGL, you must build up your desired model from a small set of *geometric primitives* - points, lines, and polygons.

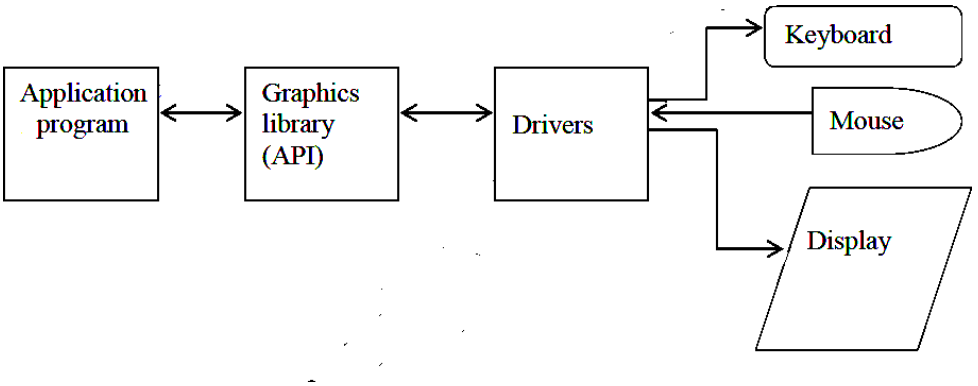


Fig 2.1 Application programmers model of graphics system

The interface between an application program and a graphics system can be

specified through a set of functions the resides in a graphics library .These specification are called the application programmer's interface (API).The application programmer see only the API and is thus shielded from the details of both the hardware and the software implementation of the graphics library. The software drivers are responsible for interpreting the output of the API and converting this data to a form that is understood by the particular hardware.

Primitives and Commands

OpenGL draws primitives-points, line segments, or polygons-subject to several selectable modes. You can control modes independently of each other i.e., 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*. A vertex defines a point, an endpoint of a line, or a corner of a polygon where two edges meet. Data (consisting of vertex coordinates, colors, normals, texture coordinates, and edge flags) is associated with a vertex, and each vertex and its associated data are processed independently, in order, and in the same way. The only exception to this rule is if the group of vertices must be *clipped* so that a particular primitive fits within a specified region; in this case, vertex data may be modified and new vertices created. The type of clipping depends on which primitive the group of vertices represents.

Commands are always processed in the order in which they are received, although there may be an indeterminate delay before a command takes effect. This means that each primitive is drawn completely before any subsequent command takes effect. It also means that state-querying commands return data that's consistent with complete execution of all previously issued OpenGL commands.

Procedural versus Descriptive

OpenGL provides you with fairly direct control over the fundamental operations of two- and three-dimensional graphics. This includes specification of such parameters as transformation matrices, lighting equation coefficients, antialiasing methods, and pixel update

operators. However, it doesn't provide you with a means for describing or modeling complex geometric objects. Thus, the OpenGL commands you issue specify how a certain result should be produced (what procedure should be followed) rather than what exactly that result should look like. That is, OpenGL is fundamentally procedural rather than descriptive. Because of this procedural nature, it helps to know how OpenGL works—the order in which it carries out its operations, for example—in order to fully understand how to use it.

Execution Model

The model for interpretation of OpenGL commands is client-server. An application (the client) issues commands, which are interpreted and processed by OpenGL (the server). The server may or may not operate on the same computer as the client. In this sense, OpenGL is network-transparent. A server can maintain several GL *contexts,* each of which is an encapsulated GL state. A client can connect to any one of these contexts. The required network protocol can be implemented by augmenting an already existing protocol (such as that of the X Window System) or by using an independent protocol. No OpenGL commands are provided for obtaining user input.

The effects of OpenGL commands on the frame buffer are ultimately controlled by the window system that allocates frame buffer resources. The window system determines which portions of the frame buffer OpenGL may access at any given time and communicates to OpenGL how those portions are structured. Therefore, there are no OpenGL commands to configure the frame buffer or initialize OpenGL. Frame buffer configuration is done outside of OpenGL in conjunction with the window system; OpenGL initialization takes place when the window system allocates a window for OpenGL rendering.

Basic OpenGL Operation

The evaluator stage of processing provides an efficient means for approximating curve and surface geometry by evaluating polynomial commands of input values. During the next stage, per-vertex operations and primitive assembly, OpenGL processes geometric primitives—points, line segments, and polygons, all of which are described by vertices. Vertices are transformed and lit, and primitives are clipped to the viewport in p**reparation for the next stage.**

Rasterization produces a series of frame buffer addresses and associated values using a two-dimensional description of a point, line segment, or polygon. Each fragment so produced is fed into the last stage, per-fragment operations, which performs the final

operations on the data before it's stored as pixels in the frame buffer. These operations include conditional updates to the frame buffer based on incoming and previously stored z-values (for z-buffering) and blending of incoming pixel colors with stored colors, as well as masking and other logical operation**s on pixel values.**

Input data can be in the form of pixels rather than vertices. Such data, which might describe an image for use in texture mapping, skips the first stage of processing described above and instead is processed as pixels, in the pixel operations stage. The result of this stage is either stored as texture memory, for use in the rasterization stage, or rasterized and the resulting fragments merged into the frame buffer just as if they were generated from geometric data.All elements of OpenGL state, including the contents of the texture memory and even of the frame buffer, can be obtained by an OpenGL application**.**

CHAPTER 3
ALGORITHM

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.

Keyboard Interface

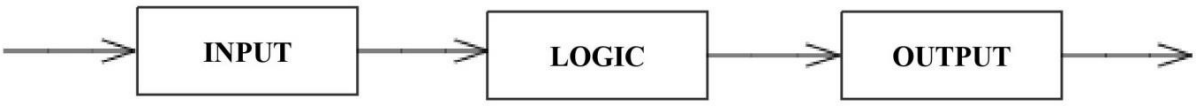
- The keys **m, v, e, r, j, s, u, n** are used to rotate the planets.
- The keys **M, V, E, R, J, S, U, N** are used to revolve the planets around the Sun.
- The key z rotates the sun, B gives both the rotation and revolution of the planets around the rotating Sun with a Comet revolution and Stars twinkle.
- 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.
- 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 rotate and revolve around the Sun and Comet to revolve around the Sun.

- **Left Button:** Rotates and revolves the planets and Comet in an anticlockwise direction.
- **Right Button:** Rotates and revolves the planets and Comet in a clockwise direction.

Flowchart



Logic

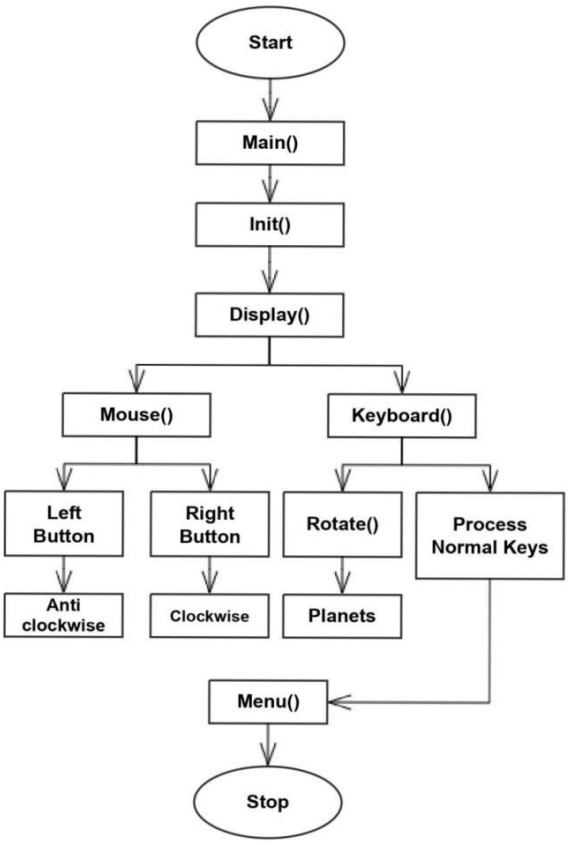


Fig 3.1 Flow chart for the Solar System project

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 :

- #include<stdlib.h>:This is C library function for standard input and output.
- #include<GL/glut.h>:This header is included to read the glut.h, gl.h and glu.h.
- #include<math.h>:This is a C library function for performing certain mathematical operations.

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

- glClearColor(. . .):

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

- glshadeModel(. . .):

To enable the smooth shading we must set the shade as follows

```
glShadeModel(GL_SMOOTH);
```

- 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);
```

- 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);

- 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)

- 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

CHAPTER 4

RESULT

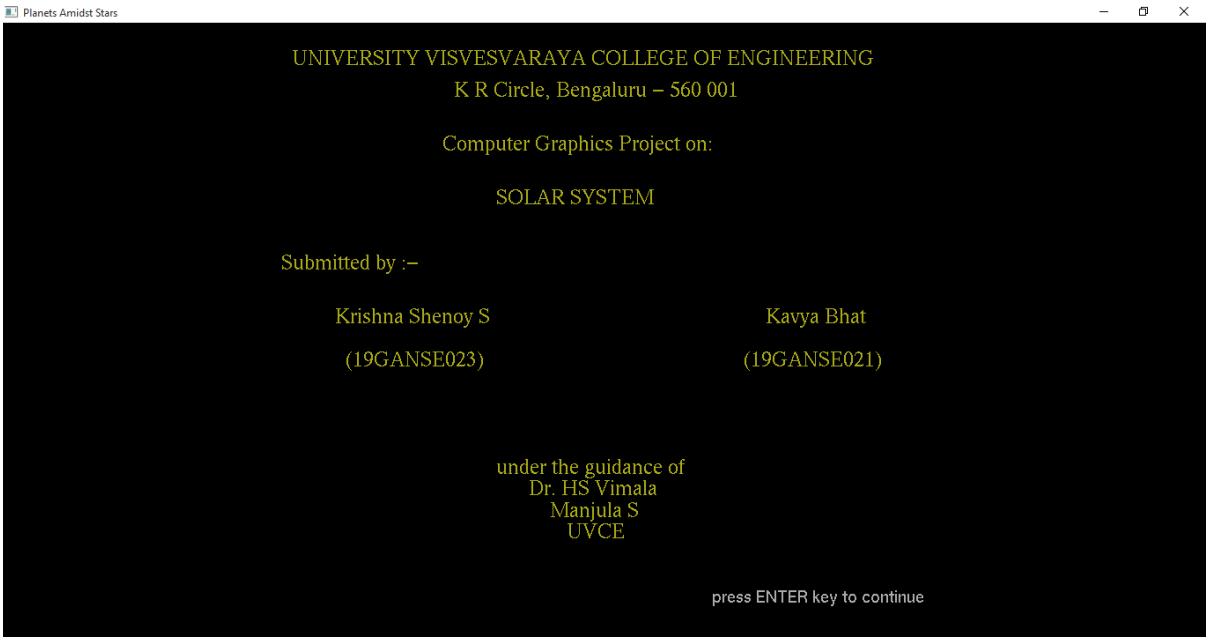


Fig 4.1 The start screen of the output appear as shown above

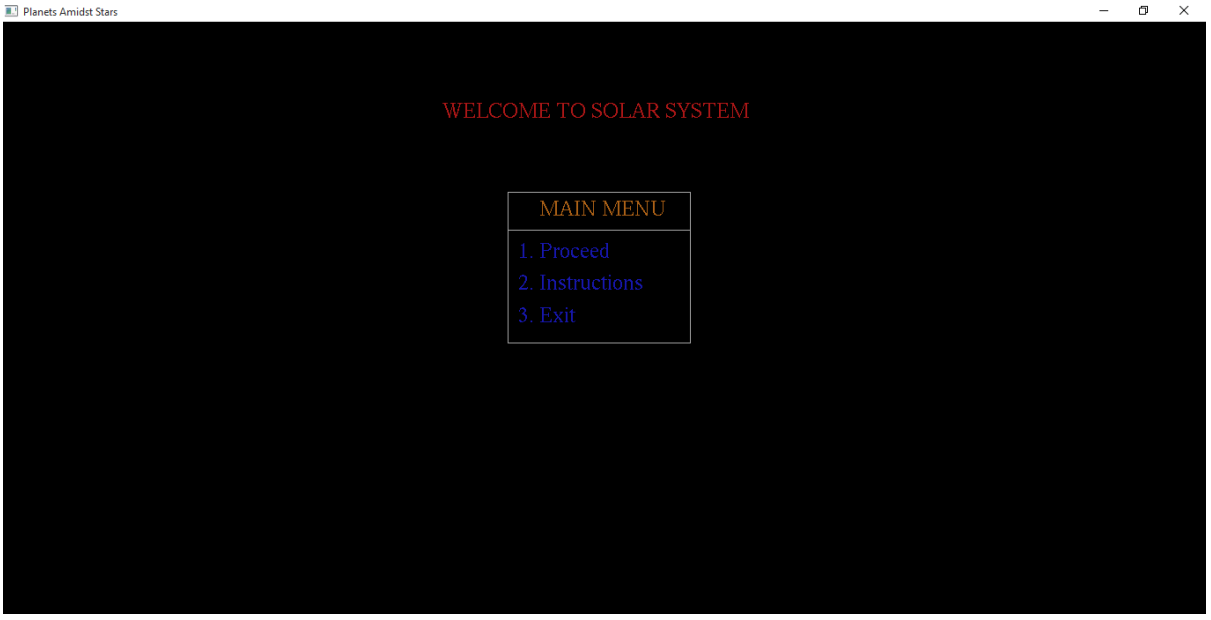


Fig 4.2 The program output starts with a menu on the screen providing these options Proceed Instructions and Exit

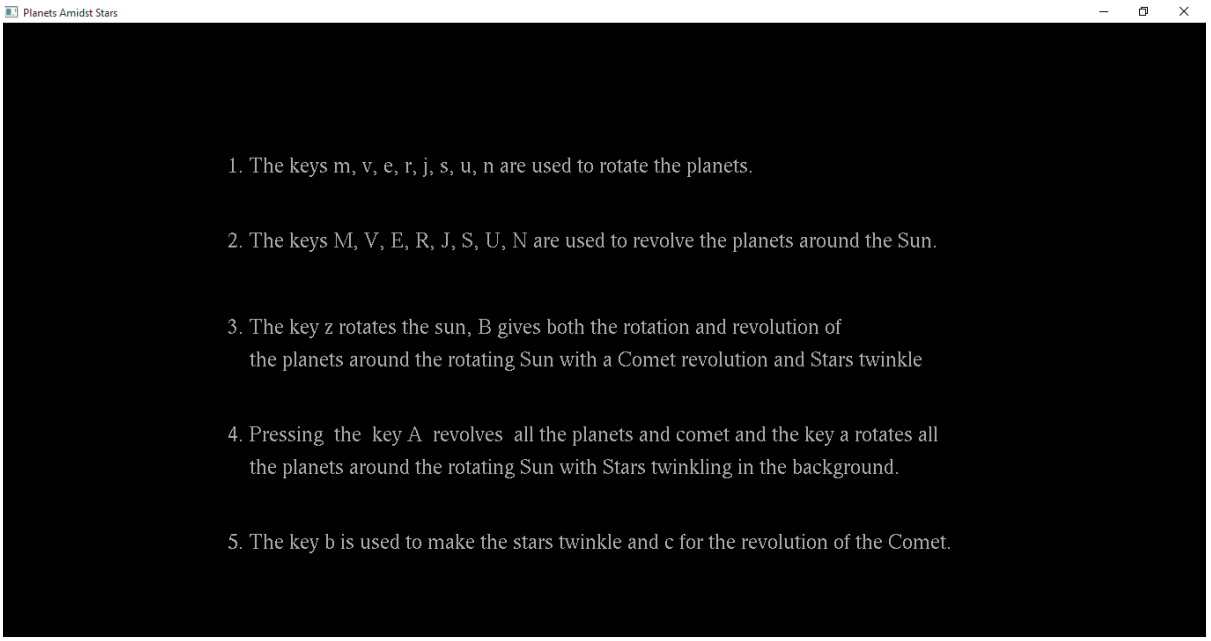


Fig 4.3 This is the instruction page that appears when pressing 2 in main menu

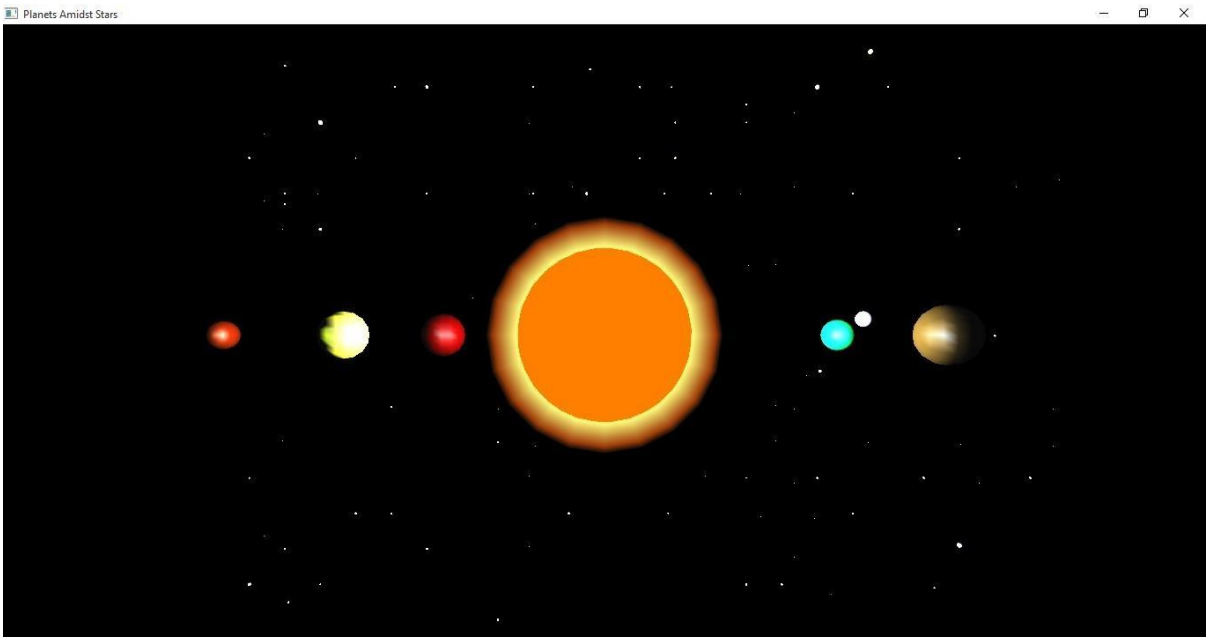


Fig 4.4 The frontpage of the project looks like this

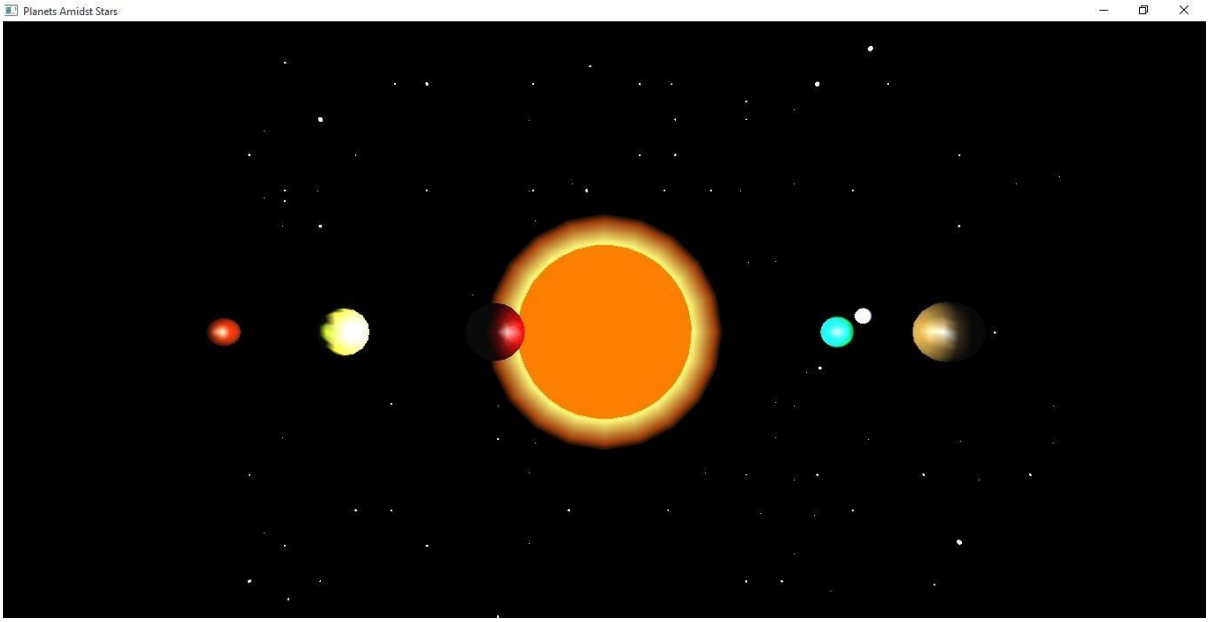


Fig 4.5 The revolution of mars(red planet) is shown below.

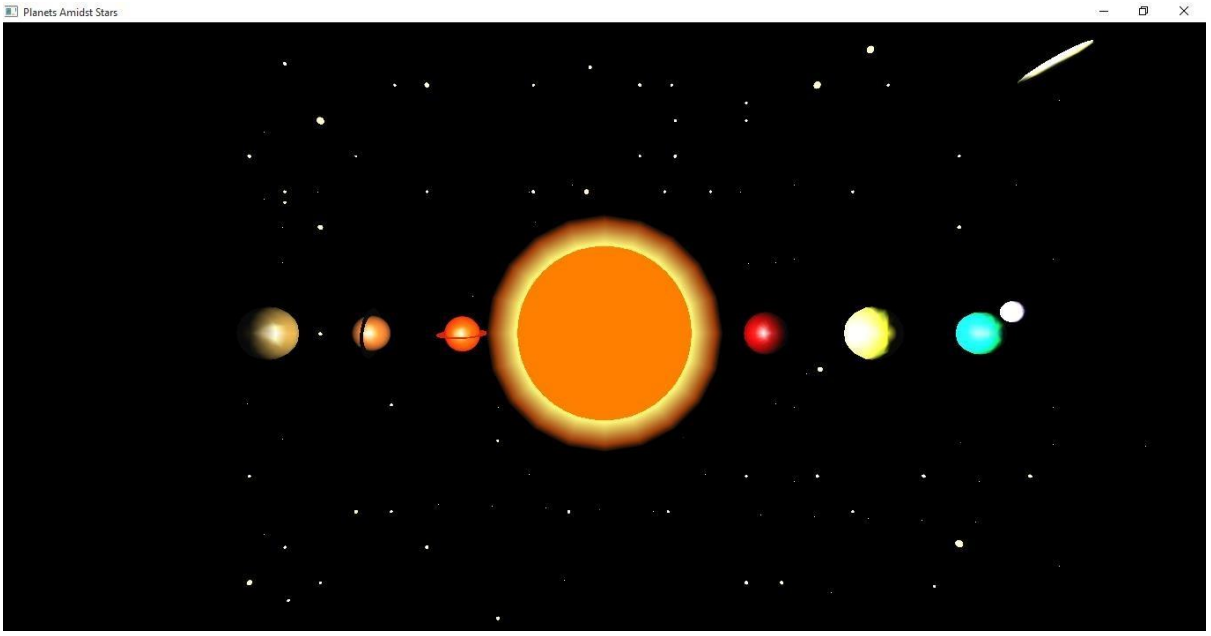


Fig 4.6 The mars has come to right side of the sun as shown below.

CHAPTER 5
CONCLUSION & ENHANCEMENT

We have tried our level best to build the project “Solar System” efficiently, but may not be the best project. An attempt has been made to develop a CG application package in OpenGL which meets the necessary requirements of the user successfully. The development of the mini-project has given us a good exposure to OpenGL by which we have learned some of the techniques which help in the development of animated pictures, gaming. Finally, we conclude that we have completed the Solar System model with possible features that we have learned through computer graphics.

Enhancement

In the future the following enhancement could be done

- Improving nature in the background.
- Giving view from different angles.

REFERENCES

Books

- Donald Hearn & Pauline Baker Computer Graphics with OpenGL Version, 3rd/4th Edition, Pearson Education, 2011.
- E. S. Angel, Interactive Computer Graphics, A top-down approach with OpenGL, (5e), Pearson Education, 2009.
- James D Foley, Andries Van Dam, Steven K Feiner, John F Huges "Computer graphics with OpenGL", Pearson education
- Kelvin Sung, Peter Shirley, Steven Baer Interactive Computer Graphics, Concepts and Applications, Cengage Learning.
- Xiang, Plastock Computer Graphics , sham's outline series, 2nd edition, TMG.
- M M Raiker, Computer Graphics using OpenGL, Filip learning/Elsevier.

e-books/online resources

- https://nptel.ac.in/courses/106106090/
- https://nptel.ac.in/courses/106102065/8
- http://www.cse.iitm.ac.in/~vplab/courses/CG/PDF/OPENGL_BASICS.pdf
- https://learnopengl.com/
- https://www.mooc-list.com/tags/computer-graphics
- https://nptel.ac.in/courses/112102101/47
- https://www.mooc-list.com/tags/computer-graphics
- https://www.nptelvideos.com/computer_graphics/

Websites

- http://www.opengl-tutorial.org/beginners-tutorials/
- https://www.glprogramming.com/blue/ch01.html
- https://www.javatpoint.com/computer-graphics-tutorial