

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgavi-590018



A MINI PROJECT REPORT

ON

“SOLAR SYSTEM SIMULATION”

A dissertation submitted in the partial fulfilment of the requirement for the Mini-Project

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by

PRAJWAL P KASHYAP

1KI20CS070

SAPTAMI S DANAPPANAVAR

1KI20CS088

Under the Guidance of

Prof. Niranjana S J

B.E, M Tech,

Assistant Professor

Dept. of CSE, KIT, Tiptur



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
KALPATARU INSTITUTE OF TECHNOLOGY**

NH - 206, TIPTUR - 572201

2022 - 2023

KALPATARU INSTITUTE OF TECHNOLOGY

(Affiliated to Visvesvaraya Technological University, Belagavi & Recognized by AICTE, New Delhi)

Department of Computer Science & Engineering

NBA ACCREDITED 2022-25

NH - 206, TIPTUR - 572201



CERTIFICATE

Certified that the mini project work entitled **"SOLAR SYSTEM SIMULATION"** is a bonafide work carried out by

PRAJWAL P KASHYAP

1KI20CS070

SAPTAMI S DANAPPANAVAR

1KI20CS088

in partial fulfilment for the Mini-Project of **Bachelor of Engineering** in **Computer Science & Engineering** of the **Visvesvaraya Technological University, Belagavi** during the year 2022-2023. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements.

Name of the guide

Prof. Niranjan S J

B.E, M Tech,

Assistant Professor

Dept. of CSE, KIT, Tiptur

Head of Department

Prof. Shashidara M S

Associate Professor & HOD

Dept. of CSE,

KIT, Tiptur

Name of Examiners

Signature with date

1. _____

2. _____

KALPATARU INSTITUTE OF TECHNOLOGY

(Affiliated to Visvesvaraya Technological University, Belagavi & Recognized by AICTE, New Delhi)

Department of Computer Science & Engineering

NBA ACCREDITED 2022-25

NH - 206, TIPTUR - 572201



DECLARATION

We, the students of Sixth Semester of Computer Science & Engineering, Kalpataru Institute of Technology NH – 206, Tiptur -572201, declare that the work entitled **SOLAR SYSTEM SIMULATION** has been successfully completed under the guidance of **Prof. Niranjan S J**, Department of Computer Science & Engineering. This dissertation work is submitted to Visvesvaraya Technological University in partial fulfilment for the mini-project of Engineering in *Computer Science & Engineering* during the academic year **2022 - 2023**.

Place: Tiptur

Date: 27/06/2023

Project Associates:

<i>Sl. No.</i>	<i>Student Name</i>	<i>USN</i>
1	PRAJWAL P KASHYAP	1KI20CS070 -
2	SAPTAMI S DANAPPANAVAR	1KI20CS088 -

ACKNOWLEDGEMENT

The satisfaction and great happiness that accompany the successful completion of any task would be incomplete without mentioning about the people who made it possible. Here we make an honest attempt to express our deepest gratitude to all those who have been helpful and responsible for the successful completion of our project.

We would like to thank **Dr. G. D. GURUMURTHY, Principal**, Kalpataru Institute of Technology, Tiptur for his continuous support and encouragement throughout the course of this project.

We would like to thank **Prof. SHASHIDHARA M S, Head of Department**, Department of Computer Science and Engineering, Kalpataru Institute of Technology, Tiptur for his continuous support and encouragement throughout the course of this project.

We are immensely indebted to our internal guide, **Prof. NIRANJAN S J**, Department of Computer Science and Engineering, Kalpataru Institute of Technology, Tiptur for his support, technical assistance and constructive suggestions and guidance for successful completion of our project. We are very much thankful to him for the encouragement that has infused at most real into us to complete the project work.

We would like to thank all faculty members of Department of Computer Science and Engineering, KIT, Tiptur, our family members, and to our friends who are directly or indirectly responsible for our success.

Thanking you,

PRAJWAL P KASHYAP

SAPTAMI S DANAPPANAVAR

ABSTRACT

Computer graphics can do many things, including modeling simulation and visualization of an object. Problem modeling is a representation of how people describe or explain an object, system, or a concept, which is usually manifested by simplification or idealization. This can be represented by physical models, the model image or mathematical formulas. 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. OpenGL support this modeling capability as OpenGL has additional features to better produce something more realistic OpenGL allows us to create a graph that can be run on any operating system only minor adjustments.

The aim of this project is to create an interactive solar system simulator that allows the creation of arbitrary dynamic systems. A survey of other work in the field is carried out and a requirements list is formulated. Novel aspects of the program's design and implementation are described. A tour of the finished program is presented, along with the results of testing and an analysis of the same. In the conclusion, proposals for improving the program are outlined.

TABLE OF CONTENTS

• ACKNOWLEDGEMENT	I
• ABSTRACT	II
• LIST OF FIGURES	III
1. INTRODUCTION -----	1
1.1 Problem Statement	
1.2 Objective of the Project	
2. HISTORICAL REVIEW -----	2-6
2.1 History of Solar System	
2.2 History of Solar System Planets	
2.3 History of Computer Graphics	
2.4 History of OpenGL	
3. REQUIREMENT SPECIFICATION -----	7-9
3.1 System Requirement	
3.1.1 Hardware Constraints	
3.2.2 Software Constraints	
3.2 Development Environment	
4. SYSTEM DESIGN -----	10-11
4.1 Flow Chart	
4.2 Keyboard Functions	
4.3 Mouse Functions	
5. SYSTEM IMPLEMENTATION -----	12-30
5.1 OpenGL Libraries	
5.2 OpenGL Primitives	
5.3 Header Files	
5.4 Functions	
5.5 Source Code	
6. RESULTS -----	31-34
7. CONCLUSION -----	35
• BIBLIOGRAPHY -----	36

LIST OF FIGURES

FIGURE NO	DESCRIPTION	PAGE NO
3.a	OpenGL	8
3.b	Microsoft Visual Studio	9
4	Flow Chart	10
5.a	OpenGL Libraries	12
5.b	OpenGL Primitives	14
6.a	Solar system with revolution of mercury	31
6.b	Solar system with revolution of planets and comet	32
6.c	Sun, twinkling stars and comet	33
6.d	Final view of solar system	34

CHAPTER 1

INTRODUCTION

The project is on SOLAR SYSTEM SIMULATION using OpenGL functions. It is a user interactive program where the user can view the required display on the screen by making use of the input devices such as keyboard and mouse. In this project a source of recreation where one can sit in front of the computer and have the vision of a planets 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!”.

1.1 Problem Statement

The aim of this project is to design and simulate the “Solar System”. The key challenges here are:

- 1) how to model the solar system?
- 2) how to rotate around the solar system?
- 3) how to make solar system visible in rotation and revolution modes?

In this project, this model is simulated by using OpenGL functions. This implementation is achieved by using a standard user input interface.

1.2 Objective of the Project

- 1) The main objective of the project is to display the rotation and revolution of planets in the solar system using controls.
- 2) To implement the concepts of Computer Graphics we have learnt.
- 3) To implement movements of the planets.
- 4) To incorporate colouring effects.

CHAPTER 2

HISTORICAL REVIEW

2.1 History of Solar System

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.

2.2 History of Solar System Planets

Sun: The Sun is the Solar System's star, and by far its chief component. Its large mass (332,900 Earth masses) produces temperatures and densities in its core great enough to sustain nuclear fusion, which releases enormous amounts of energy, mostly radiated into space as electromagnetic radiation, peaking in the 400–700 nm band we call visible light. The Sun is classified as a type G2 yellow dwarf, but this name is misleading as, compared to the majority of stars in our galaxy, the Sun is rather large and bright.

Stars are classified by the Hertzsprung–Russell diagram, a graph that plots the brightness of stars with their surface temperatures. Generally, hotter stars are brighter. Stars following this pattern are said to be on the main sequence, and the Sun lies right in the middle of it. However, stars brighter and hotter than the Sun are rare, while substantially dimmer and cooler stars, known as red dwarfs, are common, making up 85 percent of the stars in the galaxy.

Mercury: Mercury (0.4 AU from the Sun) is the closest planet to the Sun and the smallest planet in the Solar System (0.055 Earth masses). Mercury has no natural satellites, and it's only known geological features besides impact craters are lobed ridges or rupes, probably produced by a period of contraction early in its history. Mercury's almost negligible atmosphere consists of atoms blasted off its surface by the solar wind. Its relatively large iron core and thin mantle have not yet been adequately explained. Hypotheses include that its outer layers were stripped off by a giant impact, and that it was prevented from fully accreting by the young Sun's energy.

Venus: Venus (0.7 AU from the Sun) is close in size to Earth, (0.815 Earth masses) and like Earth, has a thick silicate mantle around an iron core, a substantial atmosphere and evidence of internal geological activity. However, it is much drier than Earth and its atmosphere is ninety times as dense. Venus has no natural satellites. It is the hottest planet, with surface temperatures over 400 °C, most likely due to the amount of greenhouse gases in the atmosphere. No definitive evidence of current geological activity has been detected on Venus, but it has no magnetic field that would prevent depletion of its substantial atmosphere, which suggests that its atmosphere is regularly replenished by volcanic eruptions.

Earth: Earth (1 AU from the Sun) is the largest and densest of the inner planets, the only one known to have current geological activity, and is the only place in the universe where life is known to exist. Its liquid hydrosphere is unique among the terrestrial planets, and it is also the only planet where plate tectonics has been observed. Earth's atmosphere is radically different from those of the other planets, having been altered by the presence of life to contain 21% free oxygen. It has one natural satellite, the Moon, the only large satellite of a terrestrial planet in the Solar System.

Mars: Mars (1.5 AU from the Sun) is smaller than Earth and Venus (0.107 Earth masses). It possesses an atmosphere of mostly carbon dioxide with a surface pressure of 6.1 millibars (roughly 0.6 percent that of the Earth's). Its surface, peppered with vast volcanoes such as Olympus Mons and rift valleys such as Valles Marineris, shows geological activity that may have persisted until as recently as 2 million years ago. Its red colour comes from iron oxide (rust) in its soil. Mars has two tiny natural satellites (Deimos and Phobos) thought to be captured asteroids.

Jupiter: Jupiter (5.2 AU), at 318 Earth masses, is 2.5 times the mass of all the other planets put together. It is composed largely of hydrogen and helium. Jupiter's strong internal heat creates a number of semi-permanent features in its atmosphere, such as cloud bands and the Great Red Spot. Jupiter has 63 known satellites. The four largest, Ganymede, Callisto, Io, and Europa, show similarities to the terrestrial planets, such as volcanism and internal heating. Ganymede is the largest satellite in the Solar System, is larger than Mercury.

Saturn: Saturn (9.5 AU), distinguished by its extensive ring system, has several similarities to Jupiter, such as its atmospheric composition and magnetosphere. Although Saturn has 60% of Jupiter's volume, it is less than a third as massive, at 95 Earth masses, making it the least dense planet in the Solar System. The rings of Saturn are made up of small ice and rock particles.

Saturn has 62 confirmed satellites; two of which, Titan and Enceladus, show signs of geological activity, though they are largely made of ice. Titan, the second largest moon in the Solar System, is larger than Mercury and the only satellite in the Solar System with a substantial atmosphere.

Uranus: Uranus (19.6 AU), at 14 Earth masses, is the lightest of the outer planets. Uniquely among the planets, it orbits the Sun on its side; its axial tilt is over ninety degrees to the ecliptic. It has a much colder core than the other gas giants, and radiates very little heat into space. Uranus has 27 known satellites, the largest ones being Titania, Oberon, Umbriel, Ariel and Miranda.

Neptune: Neptune (30 AU), though slightly smaller than Uranus, is more massive (equivalent to 17 Earths) and therefore more dense. It radiates more internal heat, but not as much as Jupiter or Saturn. Neptune has 13 known satellites. The largest, Triton, is geologically active, with geysers of nitrogen. Triton is the only large satellite with a retrograde orbit. Neptune is accompanied in its orbit by a number of minor planets, termed Neptune Trojans, that are in 1:1 resonance with it.

2.3 History of Computer Graphics

Computer graphics deals with generating images with the aid of computers. Today, computer graphics is a core technology in digital photography, film, video games, cell phone and computer display, and many specialized applications.

The first cathode ray tube, the Braun tube, was invented in 1897 – it in turn would permit the oscilloscope and the military control panel – the more direct precursors of the field, as they provided the first two-dimensional electronic displays that responded to programmatic or user input. New kinds of displays were needed to process the wealth of information resulting from such projects, leading to the development of computer graphics as a discipline.

In 1996, Krishnamurty and Levoy invented normal mapping – an improvement on Jim Blinn's bump mapping. By the end of the decade, computers adopted common frameworks for graphics processing such as DirectX and OpenGL. Since then, computer graphics have only become more detailed and realistic, due to more powerful graphics hardware and 3D modeling software.

2.4 History of OpenGL

By the early 1990s, Silicon Graphics (SGI) was a leader in 3D graphics for workstations. Their IRIS GL API became the industry standard, used more widely than the open standards-based PHIGS. This was because IRIS GL was considered easier to use, and because it supported immediate mode rendering. By contrast, PHIGS was considered difficult to use and outdated in functionality.

SGI's competitors (including Sun Microsystems, Hewlett-Packard and IBM) were also able to bring to market 3D hardware supported by extensions made to the PHIGS standard, which pressured SGI to open source a version of IrisGL as a public standard called OpenGL.

However, SGI had many customers for whom the change from IrisGL to OpenGL would demand significant investment. Moreover, IrisGL had API functions that were irrelevant to 3D graphics. For example, it included a windowing, keyboard and mouse API, in part because it was developed before the X Window System and Sun's NeWS. And, IrisGL libraries were unsuitable for opening due to licensing and patent issues. These factors required SGI to continue to support the advanced and proprietary Iris Inventor and Iris Performer programming APIs while market support for OpenGL matured.

CHAPTER 3

REQUIREMENT SPECIFICATION

3.1 System Requirement

The basic requirements for the development of this mini project are as follows:

3.1.1 Hardware Constraints

- Processor : Pentium PC
- RAM : 512MB
- Hard Disk : 20GB(approx)
- Display : VGA Color Monitor

3.1.2 Software Constraints

- Software: OpenGL 4.3 or above
- Operating System : Windows 98SE/2000/XP/Vista/UBUNTU
- Compiler : Eclipse/Microsoft Visual studio 2005
- Programming languages: C/C++

3.2 Development Environment

1. Software – OpenGL:

OpenGL (Open Graphics Library) is a cross-language, cross-platform API for rendering 2D and 3D vector graphics. The API is typically used to interact with a graphics processing unit (GPU), to achieve hardware-accelerated rendering.

Silicon Graphics, Inc. (SGI) began developing OpenGL in 1991 and released it on June 30, 1992; applications use it extensively in the fields of computer-aided design (CAD), virtual reality, scientific visualization, information visualization, flight simulation, and video games. Since 2006, OpenGL has been managed by the non-profit technology consortium Khronos Group.

The OpenGL specification describes an abstract API for drawing 2D and 3D graphics. Although it is possible for the API to be implemented entirely in software, it is designed to be implemented mostly or entirely in hardware. The API is defined as a set of functions which may be called by the client program, alongside a set of named integer constants (for example, the constant `GL_TEXTURE_2D`, which corresponds to the decimal number 3553).

<div>OpenGL</div> <div></div>	
Original author(s)	Silicon Graphics
Developer(s)	Khronos Group
Initial release	June 30, 1992
Stable release	4.6 / July 31, 2017
Written in	C
Type	3D graphics API
License	<p>Open source license for use of the S.I. This is a Free Software License B closely modeled on BSD, X, and Mozilla licenses.</p> <p>Trademark license for new licensees who want to use the OpenGL trademark and logo and claim conformance.</p>
Website	opengl.org

Fig 3.a OpenGL

2. Development tool – Microsoft Visual Studio:

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services, mobile apps and GUI applications. 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.

Microsoft Visual Studio	
	
Developer(s)	Microsoft
Stable release	2019 version 16.9.4 (April 13, 2021)
Preview release	2019 version 16.10.0 (April 22, 2021)
Operating system	Windows 7 SP1 and later, Windows Server 2012 R2 and later
Available in	13 languages
Type	Integrated development environment
License	Freemium
Website	visualstudio.microsoft.com

Fig 3.b Microsoft Visual Studio

CHAPTER 4

SYSTEM DESIGN

4.1 Flow Chart

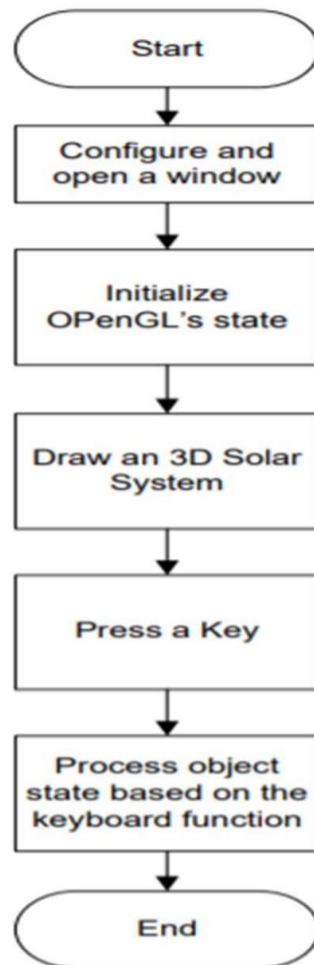


Fig 4 Flow Chart

Initially the “Main Menu” screen is displayed. Two options for further progress in the execution is displayed. In Mouse function Press left button to rotates and revolves the planets and Comet in anticlockwise direction. Press right button to rotates and revolves the planets and Comet in clockwise direction. In Keyboard function to rotates the sun press key “z”. Press key “b” is used to make the stars twinkle and “c” for the revolution of the Comet.

4.2 Keyboard Functions

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.

4.3 Mouse Functions

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.

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 OpenGL Libraries

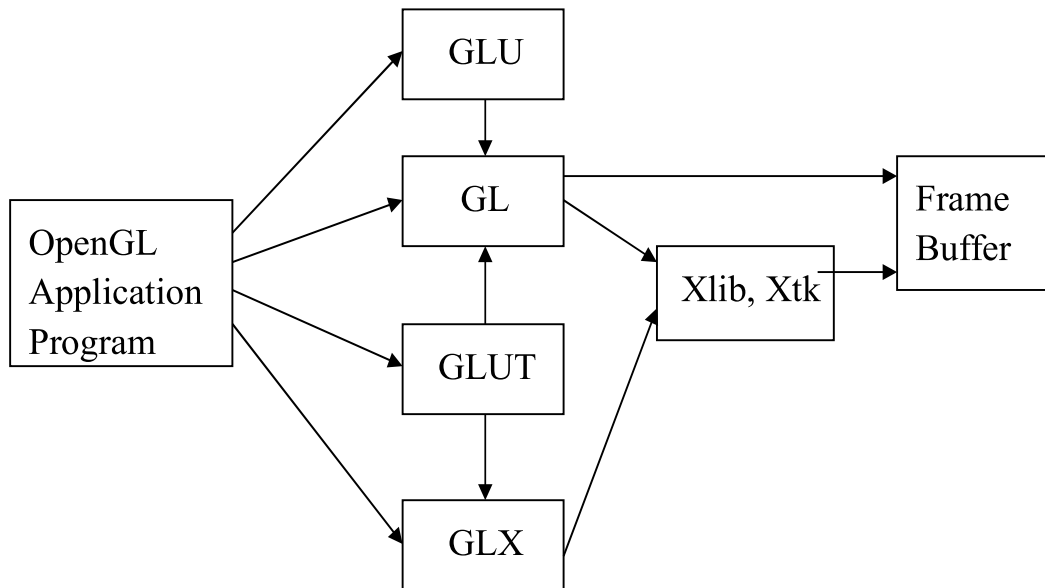


Fig 5.a OpenGL Libraries

Fig 5.1 shows the organization of the libraries for an X Window System environment. For this window system, GLUT will use GLX and the X libraries. The application program, however, can use only GLUT functions and thus can be recompiled with the GLUT library for other window systems.

1. GLU Library:

The OpenGL Utility Library (GLU) is a computer graphics library for OpenGL. It consists of a number of functions that use the base OpenGL library to provide higher-level drawing routines from the more primitive routines that OpenGL provides. It is usually distributed with the base OpenGL package.

Among these features are mapping between screen- and world-coordinates, generation of texture mipmaps, drawing of quadric surfaces, NURBS, tessellation of polygonal primitives, interpretation of OpenGL error codes, an extended range of transformation routines for setting up viewing volumes and simple positioning of the camera, generally in more human-friendly terms than the routines presented by OpenGL. All GLU functions start with the 'glu' prefix.

2. GL Library:

A graphics library is a program library designed to aid in rendering computer graphics to a monitor. This typically involves providing optimized versions of functions that handle common rendering tasks. This can be done purely in software and running on the CPU, common in embedded systems, or being hardware accelerated by a GPU, more common in PCs. By employing these functions, a program can assemble an image to be output to a monitor. Graphics libraries are mainly used in video games and simulations.

3. GLUT Library:

The OpenGL Utility Toolkit (GLUT) is a library of utilities for OpenGL programs, which primarily perform system-level I/O with the host operating system. Functions performed include window definition, window control, and monitoring of keyboard and mouse input. Routines for drawing a number of geometric primitives (both in solid and wireframe mode) are also provided, including cubes, spheres and the Utah teapot. All GLUT functions start with the 'glut' prefix. (for example, "glutPostRedisplay" marks the current window as needing to be redrawn).

FreeGLUT is an open-source alternative to the OpenGL Utility Toolkit (GLUT) library. GLUT (and hence FreeGLUT) allows the user to create and manage windows containing FreeGLUT is intended to be a full replacement for GLUT, and has only a few differences.

4. GLX Library:

GLX (initialism for "OpenGL Extension to the X Window System") is an extension to the X Window System core protocol providing an interface between OpenGL and the X Window System as well as extensions to OpenGL itself. It enables programs wishing to use OpenGL to do so within a window provided by the X Window System. GLX distinguishes two "states": indirect state and direct state.

5. Frame Buffer:

A Framebuffer is a collection of buffers that can be used as the destination for rendering. OpenGL has two kinds of framebuffers: the Default Framebuffer, which is provided by the OpenGL Context; and user-created framebuffers called Framebuffer Objects (FBOs). The buffers for default framebuffers are part of the context and usually represent a window or display device. The buffers for FBOs reference images from either Textures or Render buffers; they are never directly visible.

Default framebuffers cannot change their buffer attachments, but a particular default framebuffer may not have images associated with certain buffers. For example, the GL_BACK_RIGHT buffer will only have an image if the default framebuffer is double-buffered and uses stereoscopic 3D.

5.2 OpenGL Primitives

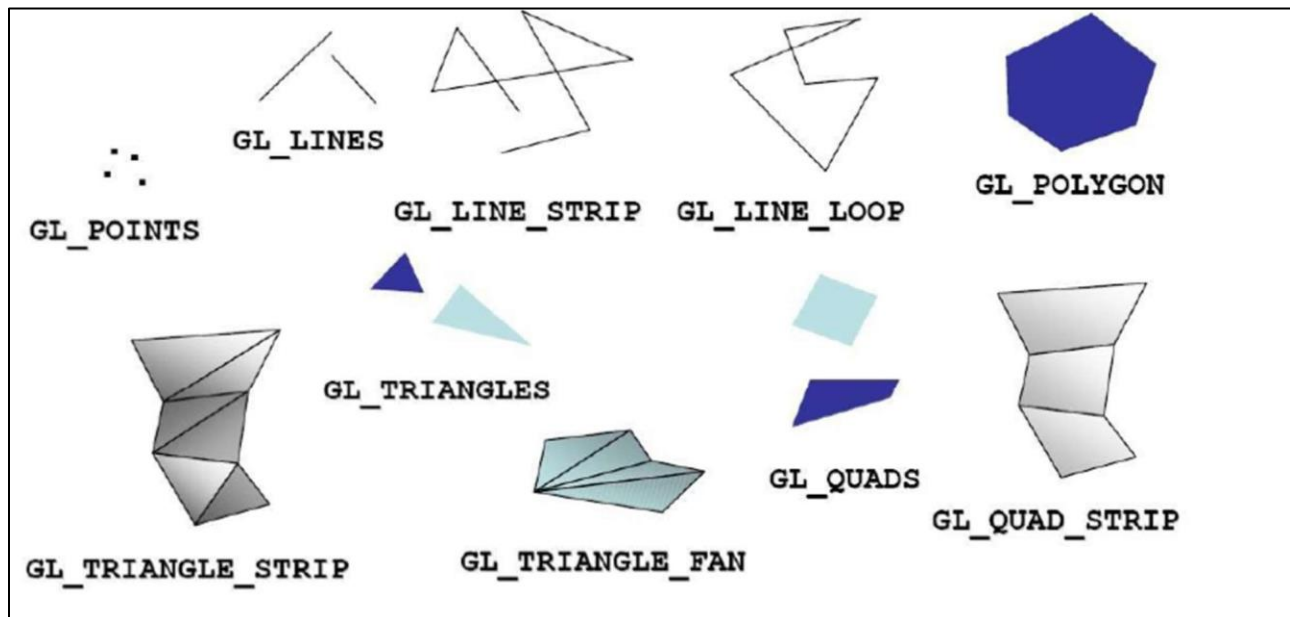


Fig 5.b OpenGL Primitives

The term Primitive in OpenGL is used to refer to two similar but separate concepts. The first meaning of "Primitive" refers to the interpretation scheme used by OpenGL to determine what a stream of vertices represents when being rendered ex: "GL_POINTS". Such sequences of vertices can be arbitrarily long.

1. Point Primitive:

There is only one kind of point primitive:

- `GL_POINTS`: This will cause OpenGL to interpret each individual vertex in the stream as a point. Points that have a Texture mapped onto them are often called "point sprites".

2. Line Primitive:

There are 3 kinds of line primitives, based on different interpretations of a vertex stream.

- `GL_LINES`: Vertices 0 and 1 are considered a line. Vertices 2 and 3 are considered a line.

And so on. If the user specifies a non-even number of vertices, then the extra vertex is ignored.

- `GL_LINE_STRIP`: The adjacent vertices are considered lines. Thus, if you pass n vertices, you will get $n-1$ lines. If the user only specifies 1 vertex, the drawing command is ignored.
- `GL_LINE_LOOP`: As line strips, except that the first and last vertices are also used as a line. Thus, you get n lines for n input vertices. If the user only specifies 1 vertex, the drawing command is ignored. The line between the first and last vertices happens after all of the previous lines in the sequence.

3. Triangle Primitives:

A triangle is a primitive formed by 3 vertices. It is the 2D shape with the smallest number of vertices, so renderers are typically designed to render them. Since it is created from only 3 vertices, it is also guaranteed to be planar. There are 3 kinds of triangle primitives, based again on different interpretations of the vertex stream:

- `GL_TRIANGLES`: Vertices 0, 1, and 2 form a triangle, Vertices 3, 4, and 5 form a triangle, and so on.
- `GL_TRIANGLE_STRIP`: Every group of 3 adjacent vertices forms a triangle. The face direction of the strip is determined by the winding of the first triangle. Each successive triangle will have its effective face order reversed, so the system compensates for that by testing it in the opposite way. A vertex stream of n length will generate $n-2$ triangles.

4. Quads:

A quad is a 4-vertex quadrilateral primitive. The four vertices are expected to be coplanar; failure to do so can lead to undefined results. A quad is typically rasterized as a pair of triangles. This is not defined by the GL specification, but it is allowed by it. This can lead to some artifacts due to how vertex/geometry shader outputs are interpolated over the 2 generated triangles.

- `GL_QUADS`: Vertices 0-3 form a quad, vertices 4-7 form another, and so on. The vertex stream must be a number of vertices divisible by 4 to work
- `GL_QUAD_STRIP`: Similar to triangle strips, a quad strip uses adjacent edges to form the next quad. In the case of quads, the third and fourth vertices of one quad are used as the edge of the next quad.

5.3 Header Files

The headers that are used are as follows:

- `#include<GL/glut.h>`: To include glut library files.
- `#include<stdio.h>`: To include standard input and output files.
- `#include<math.h>`: To include various mathematical functions.

5.4: Functions

- `void glScalef (TYPE sx, TYPE sy, TYPE sz)` alters the current matrix by a scaling of (sx, sy, sz). TYPE here is GLfloat. Here in the above considered example we use scaling to minimize the length of the curve at each iteration. For this curve we use the scale factor to be 3 units because we substitute a line by 4 lines in each iteration.
- `void glRotatef(TYPE angle, TYPE dx, TYPE dy, TYPE dz)` alters the current matrix by a rotation of angle degrees about the axis(dx, dy, dz). TYPE heris GLfloat. For a Koch curve we rotate by 60° about the z-axis.
- `void glTranslatef(TYPE x, TYPE y, TYPE z)` alters the current matrix by a displacement of (x, y, z). TYPE here is GLfloat. We need to translate to display the new position of the line from the old position and also to go out to the beginning of the next side while drawing.
- `void glLoadIdentity()` sets the current transformation matrix to an identity matrix.
- `void glPushMatrix()` pushes to the matrix stack corresponding to the current matrix mode.
- `void glPopMatrix()` pops from the matrix stack corresponding to the current matrix mode.
- `void gluOrtho2D(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top)` defines a two-dimensional viewing rectangle in the plane $z=0$.
- `void glutMouseFunc(myMouse)` refers to the mouse callback function. Here mouse interface is given to increase a level of recursion by clicking mouse button and also to decrease a level of recursion by doing the same holding the shift on the keyboard.
- `void glutKeyboardFunc(myKey)` refers to the keyboard callback function. Here keyboard interface is given to quit, the user can quit by pressing 'q' and to see next example of the implementation, the user should press 'n'.

- ☐ `void glutInit(int *argc, char**argv)` Initializes GLUT< the arguments from main are passed in and can by the application.
- ☐ `void glutCreateWindow(char *title)` Creates a window on the display. The string title can be used to label the window. The return value provides a reference to the window that can be used when there are multiple windows.
- ☐ `void glutInitDisplaymode(unsigned int mode)` Requests a display with the properties in mode. The value of mode is determined by the logical OR of options including the color model(`GLUT_RGB<GLUT_INDEX`) and buffering (`GLUT_SINGLE<GLUT_DOUBLE`).
- ☐ `void glutInitWindowSize(int width,int heights)` Specifies the initial height and width of the window in pixels.
- ☐ `void glutInitWindowPosition(int x,int y)` Specifies the initial position of the top-left corner of the window in pixels.
- ☐ `void glutMainLoop()` Cause the program to enter an event –processing loop.it should be the statement in main.
- ☐ `void glutPostRedisplay()` Requests that the display callback be executed after the current callback returns.
- ☐ `void gluLookAt(GLdouble eyex, GLdouble eyey, GLdouble eyez, GLdouble atx, GLdouble aty, GLdouble atz, GLdouble upx, GLdouble upy, GLdouble upz)` o Postmultiplies the current matrix determined by the viewer at the eye point looking at the point with specified up direction.
- ☐ `void gluPerspective(GLdouble fovy, GLdouble aspect, GLdouble near, GLdouble far)` o Defines a perspective viewing volume using the y direction field of view fovy measured in degree,the aspect ratio of the front clipping plane, and the near and far distance.

5.5 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(0.04, 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(-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();
    }
}
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

RESULTS

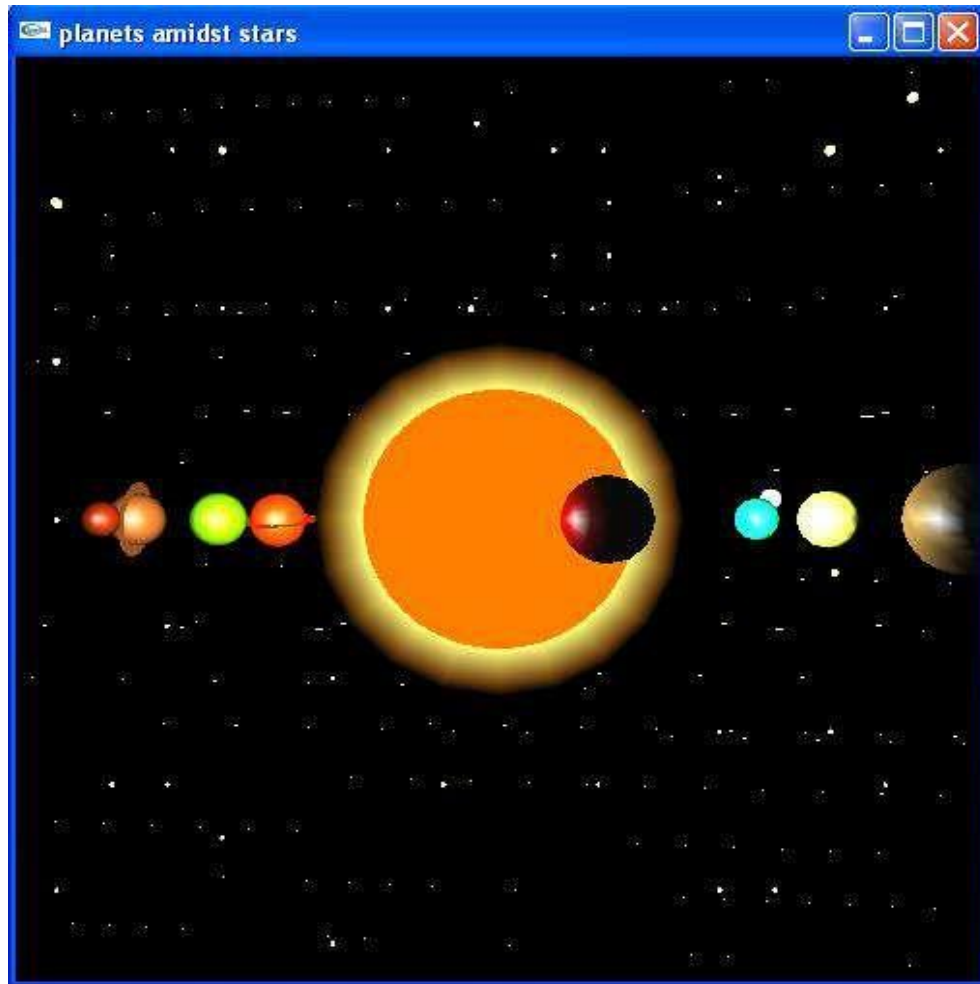


Fig 6.a Solar system with revolution of mercury

Here we can see the sun at the centre 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.

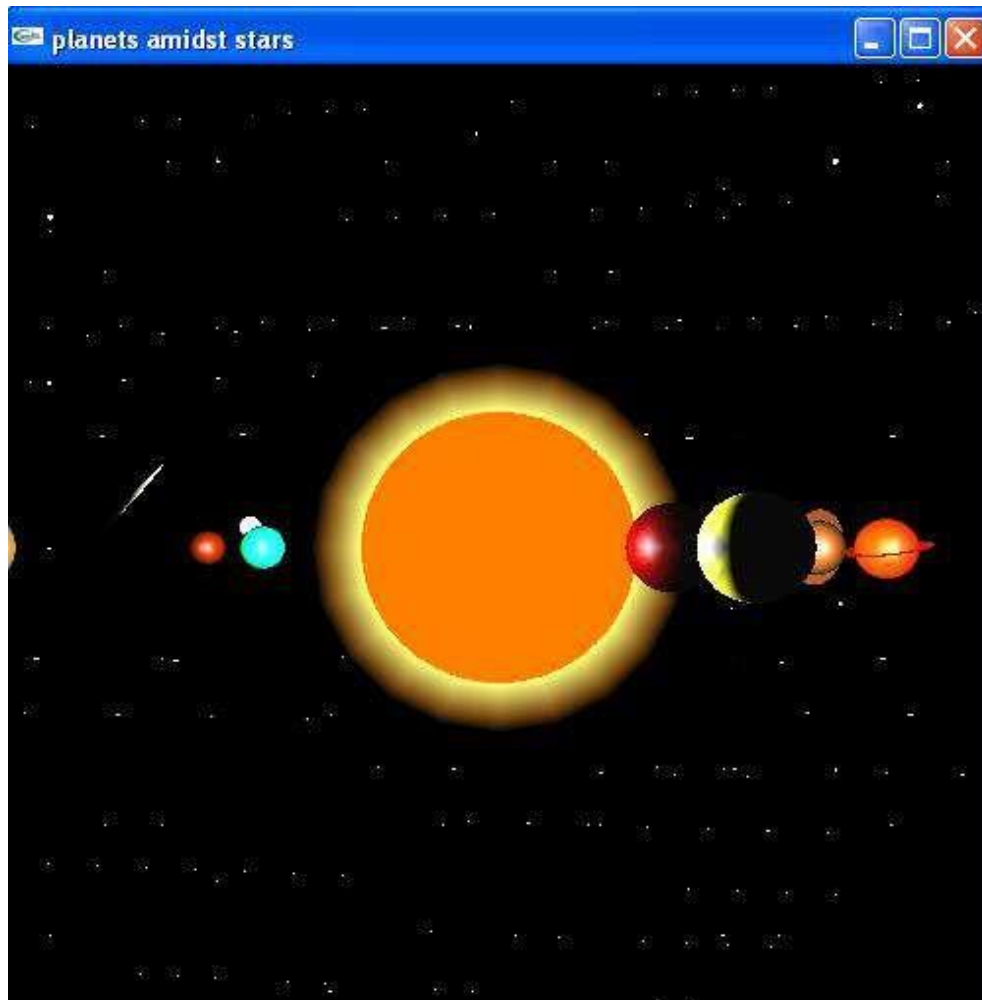


Fig 6.b Solar system with revolution of planets and comet

In this snapshot, we can see sun at the centre 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.

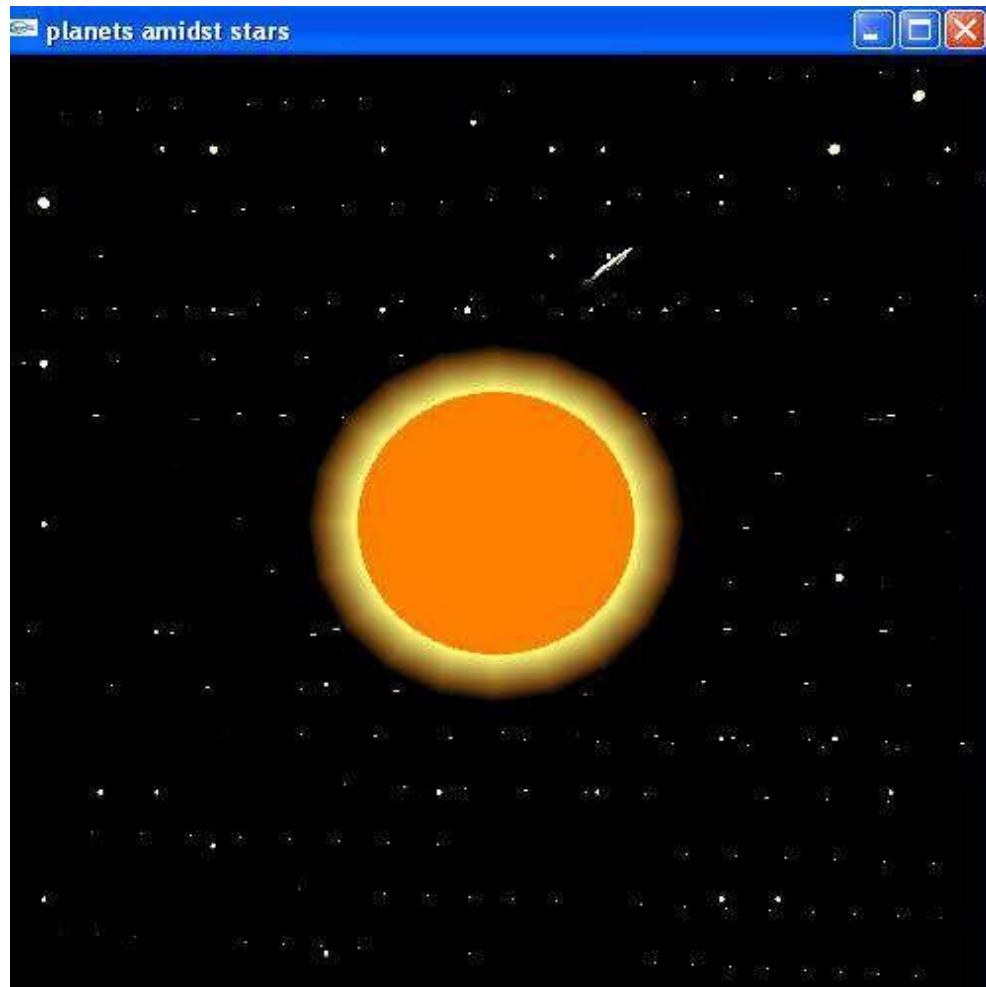


Fig 6.c Sun, twinkling stars and comet

In this snapshot we can the sun that is imagined to be placed at the centre and is also placed 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

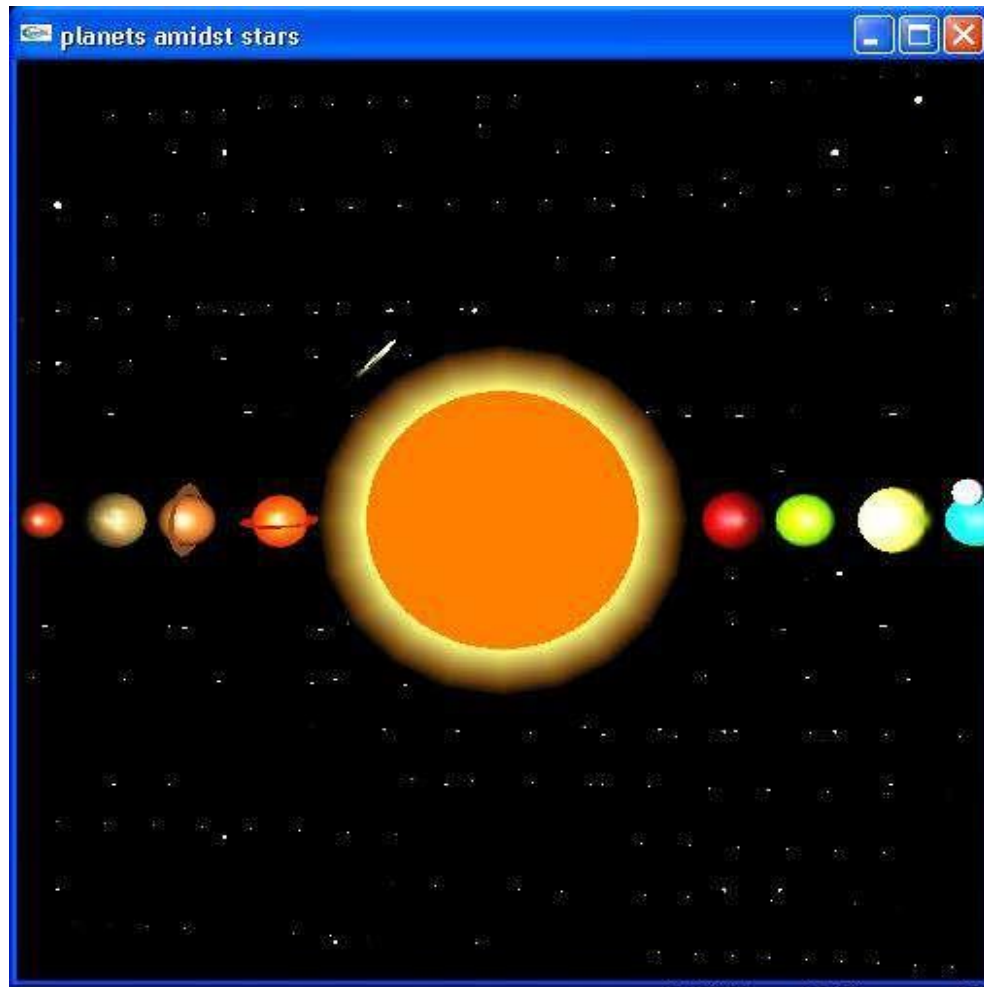


Fig 6.d Final view of solar system

In this snapshot, sun is placed at the centre 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

The code we have implemented for our project is working well to the best of our knowledge. In this project the planets, sun, comet and stars act as per the user's command. This project will serve as a delight to the eyes of the night sky watchers.

This project is both informative and entertaining. This project provided an opportunity to learn the various concepts of the subject in detail and provided us a platform to express our creativity and imagination come true.

BIBLIOGRAPHY


- **Books:**

1. Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version, 3rd / 4th Edition, Pearson Education, 2011
2. Edward Angel: Interactive Computer Graphics: A Top-Down approach with OpenGL, 5th edition. Pearson Education, 2008

- **Links:**

1. https://en.wikipedia.org/wiki/Computer_graphics - History of Computer Graphics
2. <https://en.wikipedia.org/wiki/OpenGL> - History of OpenGL
3. <https://en.wikipedia.org/wiki/OpenGL> - About OpenGL
4. https://en.wikipedia.org/wiki/OpenGL_Utility_Library - About GLU
5. https://en.wikipedia.org/wiki/OpenGL_Utility_Toolkit - About GLUT
6. <https://www.khronos.org/opengl/wiki/Primitive> - About OpenGL Primitives

Photos and Address of Mini-Project Batch Students :-

NAME	USN	ADDRESS	PHOTO
Prajwal P Kashyap	1KI20CS070	<p>Prajwal P Kashyap S/O B Prashanth, Praphulla Nilaya, Near Shankaranagara Park, Halepalya Post, Shankaranagara, Tumkur District, Tiptur Taluk, Tiptur – 572201</p> <p>E-MAIL : prajwalpkashyap26@gmail.com</p> <p>PHONE : +91 9353132694</p>	
Saptami S Danappanavar	1KI20CS088	<p>Saptami S Danappanavar D/O Somashekhara Danappanavar Akkialur Post Hangal Taluk Haveri district Akkialur - 581102</p> <p>E-MAIL : saptamisd@gmail.com</p> <p>PHONE : +91 8431801494</p>	