CHAPTER 1

INTRODUCTION

Open Graphics Library (OpenGL) is a cross-language, cross-platform application programming interface (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. he 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.

The planetary motions and the motion of satellites are governed by Kepler's laws of planetary motion and Newtons law of gravitation. While Copernicus rightly observed that the planets revolve around the Sun, it was Kepler who correctly defined their orbits and formulated these laws. There have been several misconceptions about these motions starting from people who believe in geocentric model to people who believe that the earth is flat. Some people who believe in heliocentric model don't have a clear picture about it. This project aims to enlighten the user by simulating and allowing them to visualize the motion of a satellite around a planet and helps them to get a better comprehension about satellite motion

1.1 Problem Statement

This project aims to simulate the different motions of a satellite around a planet. It also provides top view and bottom view for better understanding of retrograde motion and angled motion along with normal motions of a satellite. This will help the common user to understand the complex science behind satellite motions

1.2 Objectives

- The model aims to explain people about the planetary motions and satellite motions around a planet.
- Gives a better understanding of the different types of orbits of satellites around a planet.
- Allows the user to view the planets in different angles for better understanding of the revolution of satellites around it.
- Allows the user to understand retrograde and angular motion of a satellite by providing top view and front view.

- Demonstrates basic geometric transformations like rotation, translation and scaling in OpenGL.

1.3 Scope

Applications include:

- Space simulation programs.
- Simulation of solar system and related motions of planets.
- Space related games.
- Astronomy related demonstrations.
- Demo programs in planetariums.

CHAPTER 2

LITERATURE SURVEY

People use the term “computer graphics” to mean different things in different context. Computer graphics are pictures that are generated by a computer. Everywhere you look today, you can find examples, especially in magazines and on television. Some images look so natural you can’t distinguish them from photographs of a real scene. Others have an artificial look, intended to achieve some visual effects.

There are several ways in which the graphics generated by the program can be delivered.

- Frame- by- frame: A single frame can be drawn while the user waits.
- Frame-by-frame under control of the user: A sequence of frames can be drawn, as in a corporate power point presentation; the user presses to move on to the next slide, but otherwise has no way of interacting with the slides
- Animation: A sequence of frames proceeds at a particular rate while the user watches with delight.
- Interactive program: An interactive graphics presentation is watched, where the user controls the flow from one frame to another using an input device such as a mouse or keyboard, in a manner that was unpredictable at the time the program was written. This can delight the eye.

2.1 History

OpenGL was developed by **‘Silicon Graphics Inc‘**(SGI) on 1992 and is popular in the gaming industry where it competes with the Direct3D in the Microsoft Windows platform. OpenGL is broadly used in CAD (Computer Aided Design), virtual reality, scientific visualization, information visualization, flight simulation and video games development.

OpenGL is a standard specification that defines an API that is multi-language and multi-platform and that enables the codification of applications that output computerized graphics in 2D and 3D.

The interface consists in more than 250 different functions, which can be used to draw complex tridimensional scenes with simple primitives. It consists of many functions that help to create a real world object and a particular existence for an object can be given.

2.2 Characteristics

- OpenGL is a better documented API.
- OpenGL is also a cleaner API and much easier to learn and program.
- OpenGL has the best demonstrated 3D performance for any API.
- Microsoft's Direct3D group is already planning a major API change called Direct Primitive that will leave any existing investment in learning Direct3D immediate mode largely obsolete.

2.3 Computer Graphics Library Organisation

OpenGL stands for Open Source Graphics Library. Graphics Library is a collection of APIs (Application Programming Interfaces).

Graphics Library functions are divided in three libraries. They are as follows-

i. GL Library (OpenGL in Windows)
ii. GLU (OpenGL Utility Library)
iii. GLUT (OpenGL Utility Toolkit)

Functions in main GL library name function names that begin with the letter ‘gl’.

- GLU library uses only GL functions but contains code for creating objects and simplify viewing.
- To interface with the window system and to get input from external devices GLUT library is used, which is a combination of three libraries GLX for X windows, ‘wgl’ for Windows and ‘agl’ for Macintosh.

- These libraries are included in the application program using preprocessor directives. E.g.: #include<GL/glut.h>
- The following figure shows the library organization in OpenGL.

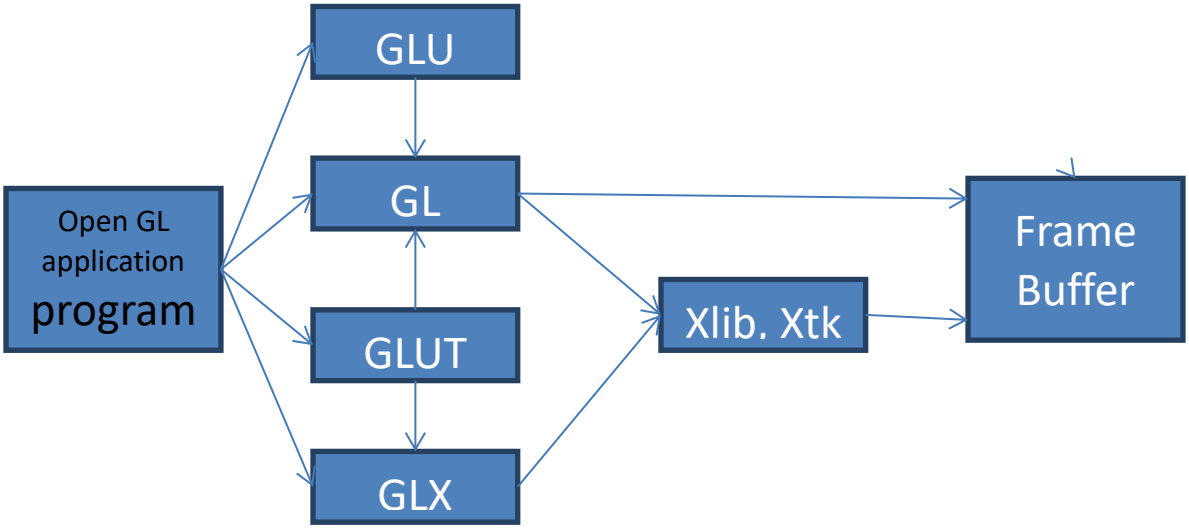

Fig 2.1 Library Organization

1.3 Graphics System and Functions

- Graphics system and functions can be considered as a black box, a term used to denote a system whose properties are only described by its inputs and output without knowing the internal working.
- Inputs to graphics system are functions calls from application program, measurements from input devices such as mouse and keyboard.
- Outputs are primarily the graphics sent to output devices.

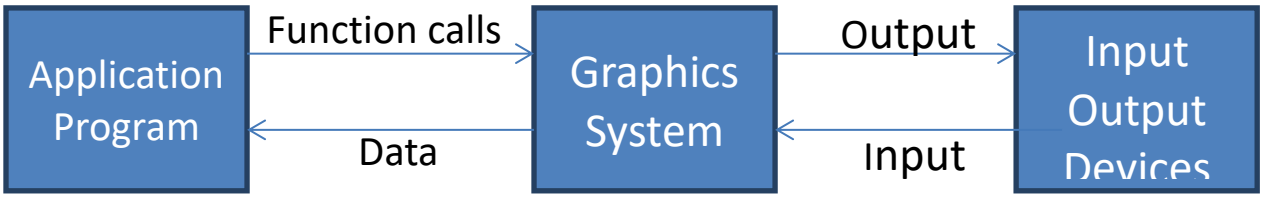

Fig 2.2 Graphics System as a Black Box

API's are described through functions in its library. These functions are divided into seven major groups.

1) Primitive Functions:

   Primitive functions define the low level objects or atomic entities that a system can display, the primitives include line segments, polygons, pixels, points, text and various types of curves and surfaces.

2) Attribute Functions:

   Attribute Functions allow us to perform operations ranging from choosing the color to display a line segment, to packing a pattern to fill inside any solid figure.

3) Viewing Functions:

   Viewing functions allow us to specify various views.

4) Transformation Functions:

   Transformation functions allow us to carry out transformation of objects such as rotation, translation and scaling.

5) Input Functions:

   Input functions allow us to deal with the diverse forms of input that characterize modern graphics system. It deals with devices such as keyboard, mouse and data tablets.

6) Control Functions:

   Control Functions enable us to communicate with the window system, to initialize the programs, and to deal with any errors that occur during the execution of the program.

7) Query Functions:

   Query Functions provides information about the API.

CHAPTER 3

SYSTEM REQUIREMENTS

Requirement's analysis is critical for project development. Requirements must be documented, actionable, measurable, testable and defined to a level of detail sufficient for system design. Requirements can be architectural, structural, behavioural, functional, and non-functional.A software requirements specification (SRS) is a comprehensive description of the intended purpose and the environment for software under development.

3.1 Hardware Requirement

- Minimum of 2GB of main memory
- Minimum of 3GB of storage
- Keyboard
- Mouse
- Display Unit
- Dual-Core or AMD with minimum of 1.5GHz speed

3.2 Software Requirement

- Windows – XP/7/8/10
- Microsoft Visual Studio C/C++ 7.0 and above versions
- OpenGL Files
- DirectX 8.0 and above versions

Header Files

- glut.h

Object File Libraries

- glut32.lib

DLL files

- glut32.dll

CHAPTER 4

DESIGN AND IMPEMENTATION

4.1 Header Files Used

- **#include<GL/glut.h>**

GLUT (pronounced like the glut in gluttony) is the OpenGL Utility Toolkit, a window system independent toolkit for writing OpenGL programs. It implements a simple windowing application programming interface (API) for OpenGL. GLUT makes it considerably easier to learn about and explore OpenGL programming. GLUT provides a portable API so you can write a single OpenGL program that works across all PC and workstation OS platforms

- **#include<window.h>**

windows.h is a Windows-specific header file for the C and C++ programming languages which contains declarations for all of the functions in the Windows API, all the common macros used by Windows programmers, and all the data types used by the various functions and subsystems. It defines a very large number of Windows specific functions that can be used in C. The Win32 API can be added to a C programming project by including the <windows.h> header file and linking to the appropriate libraries. To use functions in *xxxx*.dll, the program must be linked to *xxxx*.lib (or lib*xxxx*.dll.a in MinGW). Some headers are not associated with a .dll but with a static library (e.g. scrnsave.h needs scrnsave.lib).

4.2 OpenGLAPI's Used

- **void glColor3f(float red, float green, float blue);**

This function is used to mention the color in which the pixel should appear. The number 3 specifies the number of arguments that the function would take. 'f ' gives the type that is float. The arguments are in the order RGB (Red, Green, Blue)**.** The color of the pixel can be specified as the combination of these 3 primary colors.

- **Void glClearColor(int red, int green, int blue, int alpha);**

This function is used to clear the color of the screen. The 4 values that are passed as arguments for this function are (RED, GREEN, BLUE, ALPHA) where the red green and blue components are taken to set the background color and alpha is a value that specifies depth of the window. It is used for 3D images.

- **void glutKeyboardFunc();**

void glutKeyboardFunc(void (*func)(unsigned char key, int x, int y)); where func is the new keyboard callback function. glutKeyboardFunc sets the keyboard callback for the current window. When a user types into the window, each key press generating an ASCII character will generate a keyboard callback. The key callback parameter is the generated ASCII character. The x and y callback parameters indicate the mouse location in window relative coordinates when the key was pressed. When a new window is created, no keyboard callback is initially registered, and ASCII keystrokes in the window are ignored. Passing NULL to glutKeyboardFunc disables the generation of keyboard callbacks.

- **void glFlush();**

Different GL implementations buffer commands in several different locations, including network buffers and the graphics accelerator itself. glFlush empties all of these buffers, causing all issued commands to be executed as quickly as they are accepted by the actual rendering engine. Though this execution may not be completed in any particular time period, it does complete in finite time.

- **void glMatrixMode(GLenum mode);**

where mode Specifies which matrix stack is the target for subsequent matrix operations. Three values are accepted: GL_MODELVIEW, GL_PROJECTION, and GL_TEXTURE. The initial value is GL_MODELVIEW. glMatrixMode sets the current matrix mode. mode can assume one of three values:

GL_MODELVIEW Applies subsequent matrix operations to the modelview matrix stack.
GL_PROJECTION Applies subsequent matrix operations to the projection matrix stack

- **void glViewport(GLint x, GLint y, GLsizei width, GLsizei height)**

where x, y Specify the lower left corner of the viewport rectangle, in pixels. The initial value is (0, 0). The width, height Specify the width and height of the viewport. When a GL context is first attached to a surface (e.g. window), width and height are set to the dimensions of that surface. GlViewport specifies the affine transformation of x and y from normalized device coordinates to window coordinates. Let (xnd, ynd) be normalized device coordinates. Then the window coordinates (xw, yw) are computed as follows: xw = (xnd + 1) width/2 + x yw = (ynd + 1) height/2 + y Viewport width and height are silently clamped to a range that depends on the implementation. To query this range, call glGetInteger with argument GL_MAX_VIEWPORT_DIMS.

- **void glutInit(int *argcp, char **argv);**

glutInit will initialize the GLUT library and negotiate a session with the window system. During this process, glutInit may cause the termination of the GLUT program with an error message to the user if GLUT cannot be properly initialized. Examples of this situation include the failure to connect to the window system, the lack of window system support for OpenGL, and invalid command line options.glutInit also processes command line options, but the specific options parse are window system dependent.

- **void glutReshapeFunc(void (*func)(int width, int height));**

glutReshapeFunc sets the reshape callback for the current window. The reshape callback is triggered when a window is reshaped. A reshape callback is also triggered immediately before a window's first display callback after a window is created or whenever an overlay for the window is established. The width andheight parameters of the callback specify the new window size in pixels. Before the callback, the current window is set to the window that has been reshaped. If a reshape callback is not registered for a window or NULL is passed to glutReshapeFunc (to deregister a previously registered callback), the default reshape callback is used. This default callback will simply glOrtho ( ) Syntax: void glOrtho (GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far); The function defines an orthographic viewing volume with all parameters measured from the centre of the projection plane.

- **void glutMainLoop(void);**

glutMainLoop enters the GLUT event processing loop. This routine should be called at most once in a GLUT program. Once called, this routine will never stop.

- **glutPostRedisplay()**

glutPostRedisplay, glutPostWindowRedisplay — marks the current or specified window as needing to be redisplayed

4.3 User Defined Functions

- **void background()**

This functions act as basis for creating the stars. The stars are specified at the random coordinates using glVertex function and glPointSize(2). These functions is later invoked by calling it thus creating the stars at the random coordinates

- **void drawArtSat()**

This function is used to draw the satellite at a specific position. The different background materials like Satellite body, Satellite panels, Satellite dish and antennae are designed here using glutSolidCube and glutSolidSphere. These functions are later invoked by calling it thus creating the satellite at a specific position using glTranslate.

- **void frontview()**

This function is used to display the model from tthe front. This model contains a moon which is revolving around a planet in a non retrograde motion and a satelilite at its orbit.
The moon can revolve at an angle around the planet as per input given by user. Similarly, the Satellite can also move at an angle as per the user's input.

- **void topview()**

This function is used to display the model from top view. This model contains a moon which is revolving around a planet in a non retrograde motion and a satelilite at its orbit.

The moon can revolve at an angle around the planet as per input given by user. Similarly, the Satellite can also move at an angle as per the user's input.

- **void sideview()**

This function is used to display the model from side view. This model contains a moon which is revolving around a planet in a non retrograde motion and a satelilite at its orbit.
The moon can revolve at an angle around the planet as per input given by user. Similarly, the Satellite can also move at an angle as per the user's input.

CHAPTER 5

SNAPSHOTS

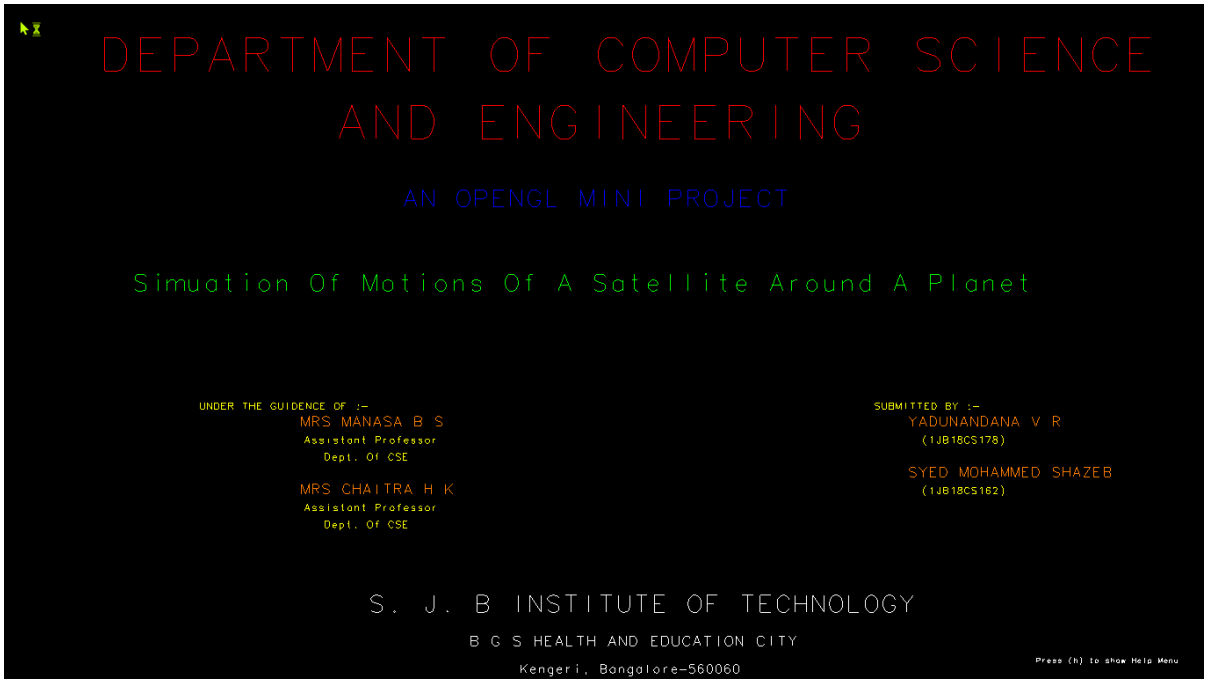

Fig 5.1 Introduction Page

Fig 5.2 Help page

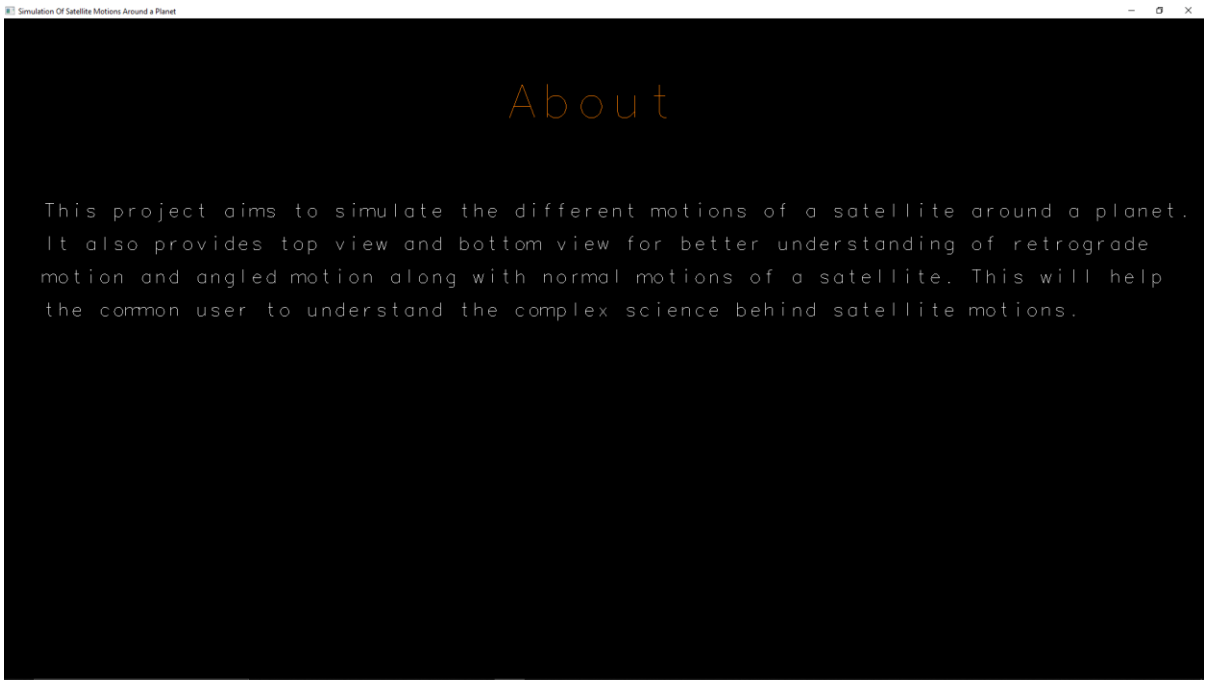

Fig 5.3 About page

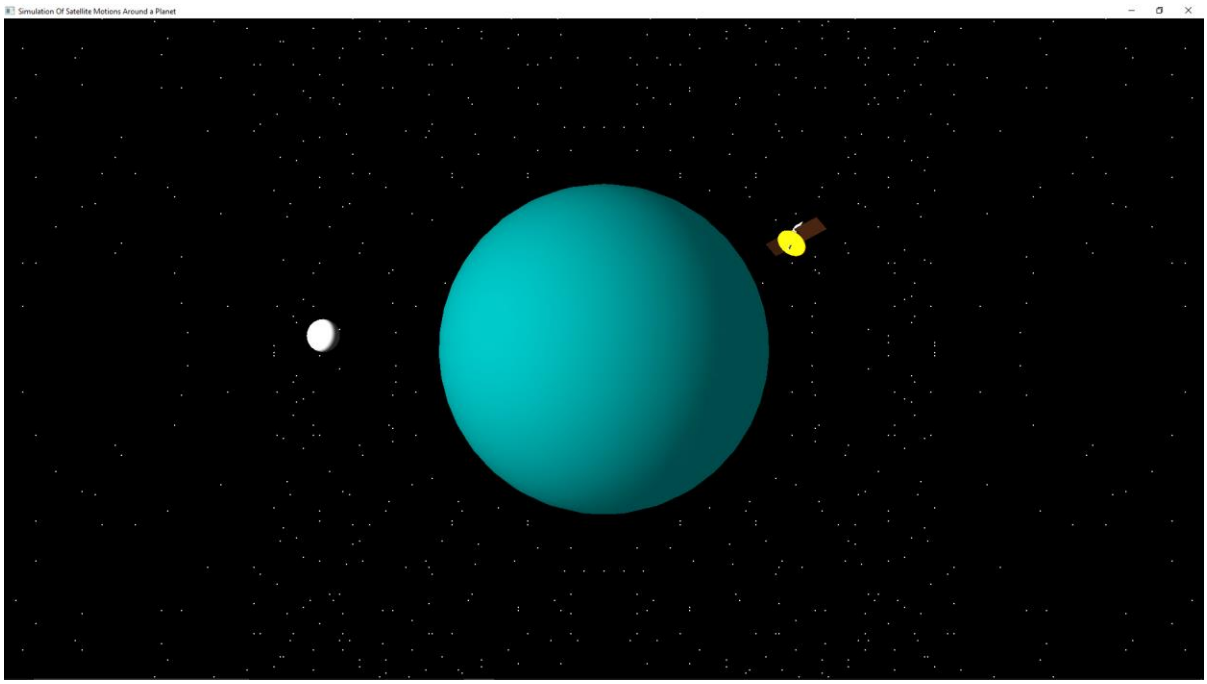

Fig 5.4 Front View

(Depicts the Front View of the planet and motion of the natural and artificial satellites)

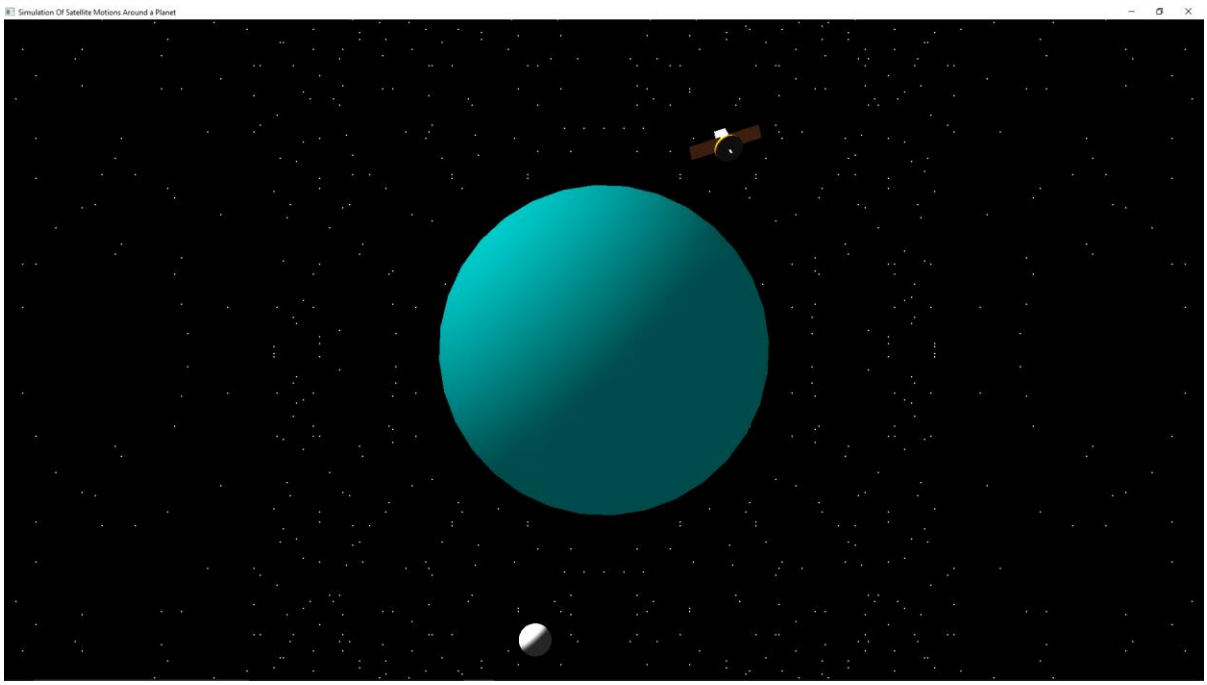

Fig 5.5 Top View

(Depicts the Top View of the planet and motion of the natural and artificial satellites)

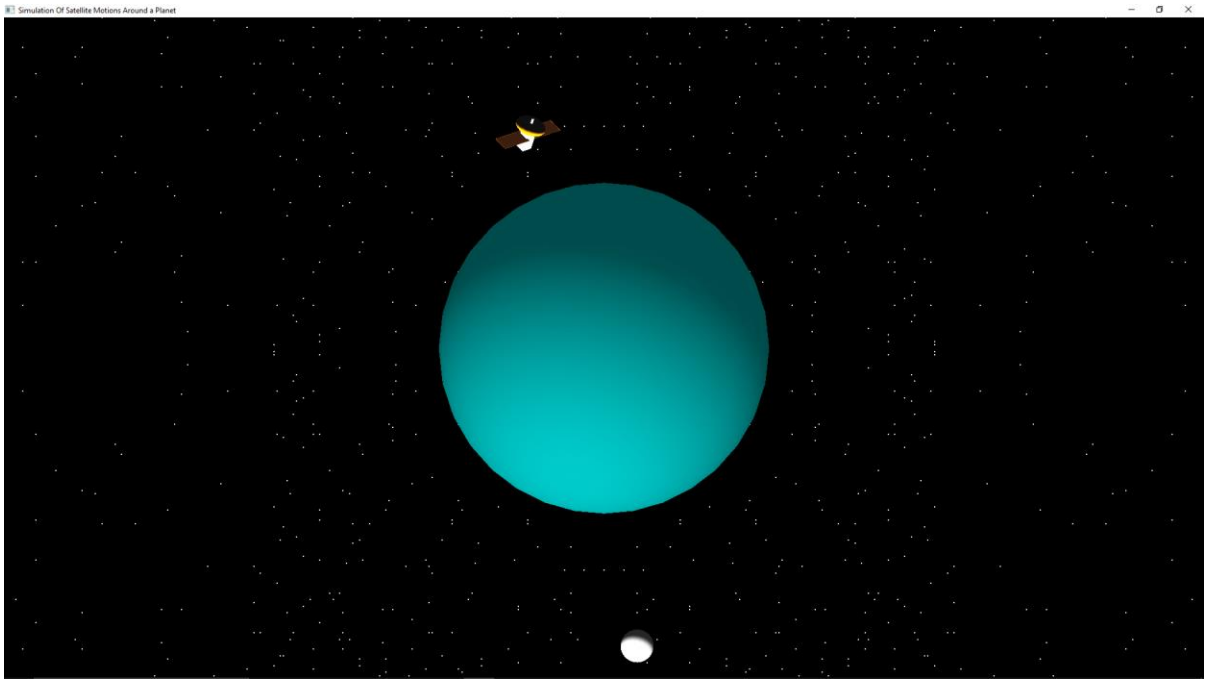

Fig 5.6 Side View

(Depicts the Side View of the planet and motion of the natural and artificial satellites)

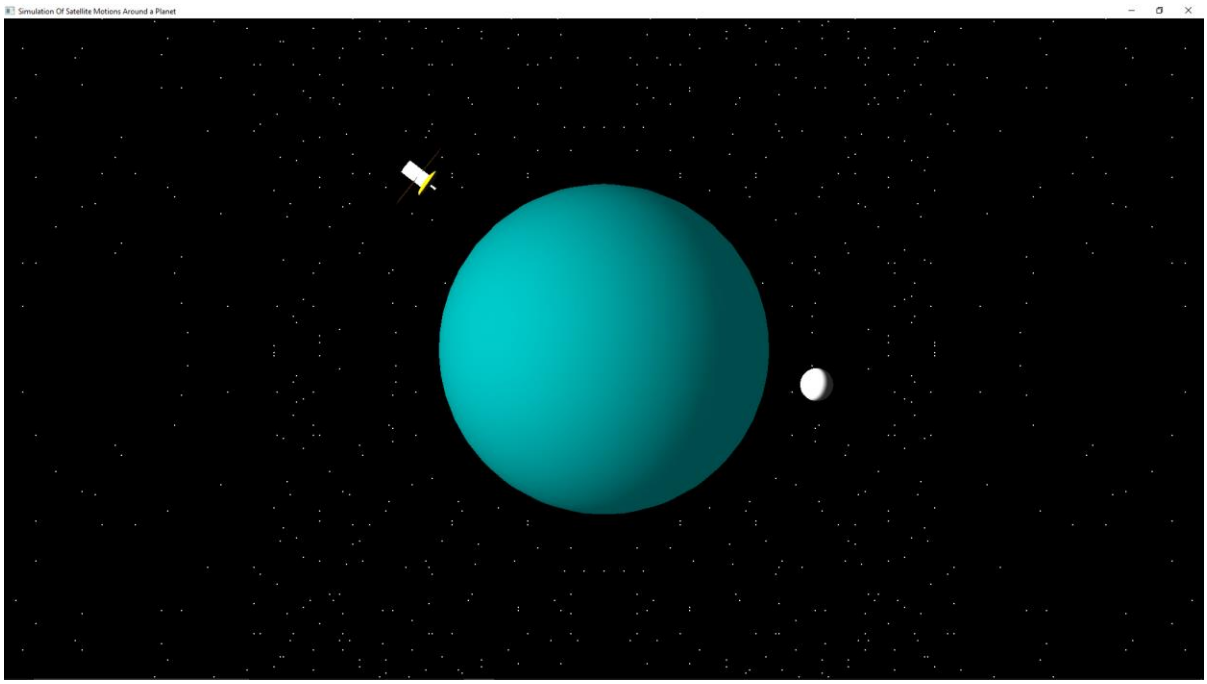

Fig 5.7 Angular Motion

(Depicts the Angular Motion of natural satellite)

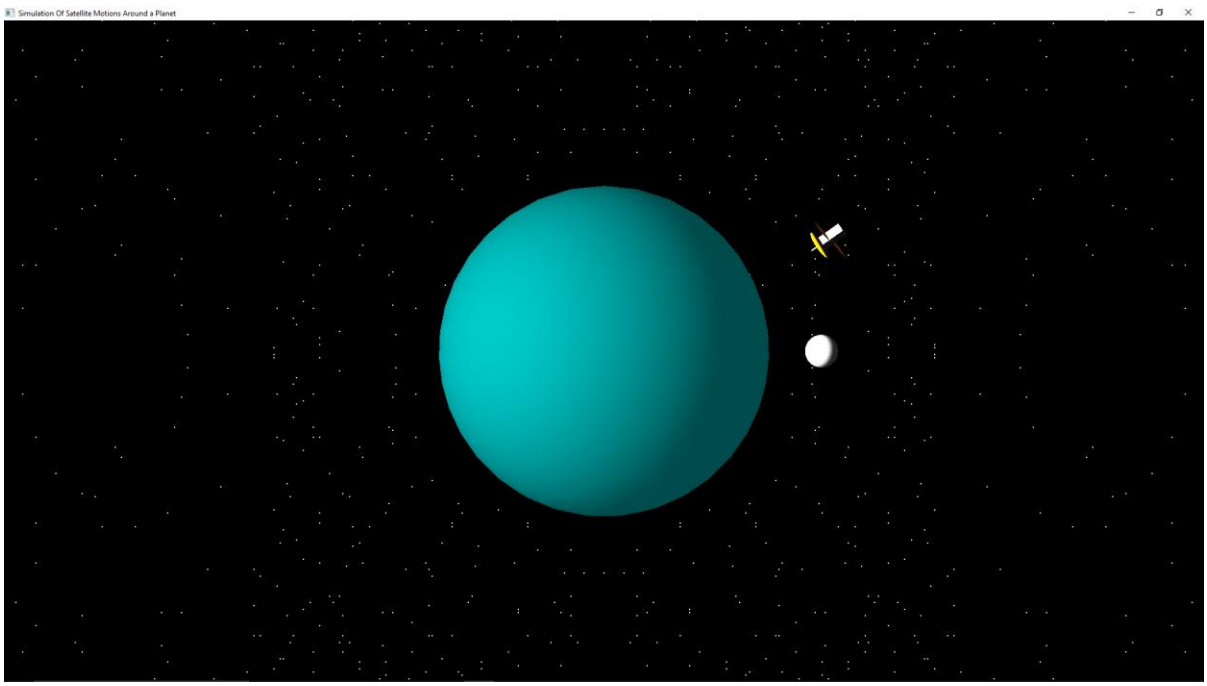

Fig 5.8 Retrograde Motion

(Depicts the Retrograde Motion of natural satellite)

CONCLUSION

This project helps to understand the concepts behind OpenGL and its programming. This project gives a brief insight as to how programs, involving graphics, are written using OpenGL.

This project demonstrates how the OpenGL can be used to implement graphics which can be applied in various fields such as advertising industries to endorse company's products animation studios to make animated films, cartoons, gaming industries by displaying the use of high-end graphics in designing games, in engineering, architecture, medical fields by creating real-world models for better understanding, clarity and bringing out fresh, new ideas to enhance them.

The conclusion from this project is that it describes the motions of a satellite around a planet clearly so that the users can understand them without any doubts.

FUTURE ENHANCEMENT

- This project attempts to show the motion of just one natural satellite and one artificial satellite. This can be extended to more than one.
- The whole model can be extended to also include the entire solar system.
- This project uses limited mouse and keyboard functions for user interaction and as a future enhancement this project can make use of more keyboard functions to select the number of satellites and select different planets of the solar system.
- The facility to view in a perspective mode can be added in future.
- Textures can be added to make the object look like real.

BIBLIOGRAPHY

[1] Interactive Computer Graphics Pearson Education 5th Edition: Edward Angel

[2] OpenGL Documentation

[3] https://learnopengl.com

[4] https://docs.microsoft.com/en-us/windows/win32/opengl/opengl

[5] Computer Graphics with OpenGL Version, 4thEdition: Donald Hearn, Pauline Baker