**CHAPTER 1**

**INTRODUCTION**

**Computer graphics** are graphics created using computers and more generally, the representation and manipulation of image data by a computer. The term computer graphics has been used in a broad sense to describe "almost everything on computers that is not text or sound".

The development of computer graphics has made computers easier to interact with, better for understanding and interpreting many types of data. Developments in computer graphics have had a profound impact on many types of media and have revolutionized animation, movies and the video game industry.

**The various applications of computer graphics are:**

* Graphs and charts
* Computer-Aided design
* Virtual-Reality environment
* Data Visualization
* Education and Training
* Computer Art
* Entertainment
* Image Processing
* Graphical User interfaces

**Graphs and Charts:**

An early application for computer graphics is the display of simple data graphs, usually plotted on a character printer. Data plotting is still one of the most common graphics applications, but today one can easily generate graphs showing highly complex data relationships for printed reports or for presentations using 35 mm slides, transparencies, or animated videos. Graphs and charts are commonly used to summarize financial, statistical, mathematical, scientific, engineering, and economic data for research reports, managerial summaries, consumer information bulletins, and other types of publications.

**Computer Aided Design**:

A major use of computer graphics is in design processes—particularly for engineering and architectural systems, although most products are now computer designed. Generally referred to as CAD, computer-aided design, or CADD, computer-aided drafting and design, these methods are now routinely used in the design of buildings, automobiles, aircraft, watercraft, spacecraft, computers, textiles, home appliances, and a multitude of other products. The manufacturing process is also tied in to the computer description of designed objects so that the fabrication of a product can be automated, using methods that are referred to as CAM, computer-aided manufacturing.

**Virtual Reality Environment**:

It is a recent application of computer graphics which is used to create virtual-reality environments in which a user can interact with the objects in a three-dimensional scene. Specialized hardware devices provide three-dimensional viewing effects and allow the user to “pick up” objects in a scene. Animations in virtual-reality environments are often used to train heavy equipment operators or to analyze the effectiveness of various cabin configurations and control placements. This allows the designer to explore various positions of the bucket or backhoe that might obstruct the operator’s view, which can then be taken into account in the overall tractor design.

**Data Visualization**:

Producing graphical representations for scientific, engineering, and medical data sets and processes is another fairly new application of computer graphics, which is generally referred to as scientific visualization. The term business visualization is used in connection with data sets related to commerce, industry, and other nonscientific areas. Numerical computer simulations, for example, frequently produce data files containing thousands and even millions of values. Similarly, satellite cameras and other recording sources are amassing large data files faster than they can be interpreted. Other visualization techniques include contour plots, renderings for constant-value surfaces or other spatial regions, and specially designed shapes that are used to represent different data types.

**Education and Training**:

Computer-generated models of physical, financial, political, social, economic, and other systems are often used as educational aids. Models of physical processes, physiological functions, population trends, or equipment, such as the color-coded diagram in for some training applications, special hardware systems are designed. Examples of such specialized systems are the simulators for practice sessions or training of ship captains, aircraft pilots, heavy-equipment operators, and air traffic-control personnel. Some simulators have no video screens; a flight simulator with only a control panel for instrument flying. But most simulators provide screens for visual displays of the external environment with multiple panels is mounted in front of the simulator.

**Entertainment:**

Television productions, motion pictures, and music videos routinely use computer-graphics methods. Sometimes graphics images are combined with live actors and scenes, and sometimes the films are completely generated using computer-rendering and animation techniques. Many TV series regularly employ computer-graphics methods to produce special effects, such as the scene in Figure from the television series Deep Space Nine. Some television programs also use animation techniques to combine computer-generated figures of people, animals, or cartoon characters with the live actors in a scene or to transform an actor’s face into another shape. And many programs employ computer graphics to generate buildings, terrain features, or other backgrounds for a scene.

**Computer Art**:

Both fine art and commercial art make use of computer-graphics methods. Artists now have available a variety of computer methods and tools, including specialized hardware, commercial software packages (such as Lumena), symbolic mathematics programs (such as Mathematica), CAD packages, desktop publishing software, and animation systems that provide facilities for designing object shapes and specifying object motions. Example: use of a paintbrush program that allows an artist to “paint” pictures on the screen of a video monitor .A paintbrush system, with a Wacom cordless, pressure-sensitive stylus, was used to produce the electronic painting. The stylus translates changing hand pressure into variable line widths, brush sizes, and color gradations.

**Image Processing**:

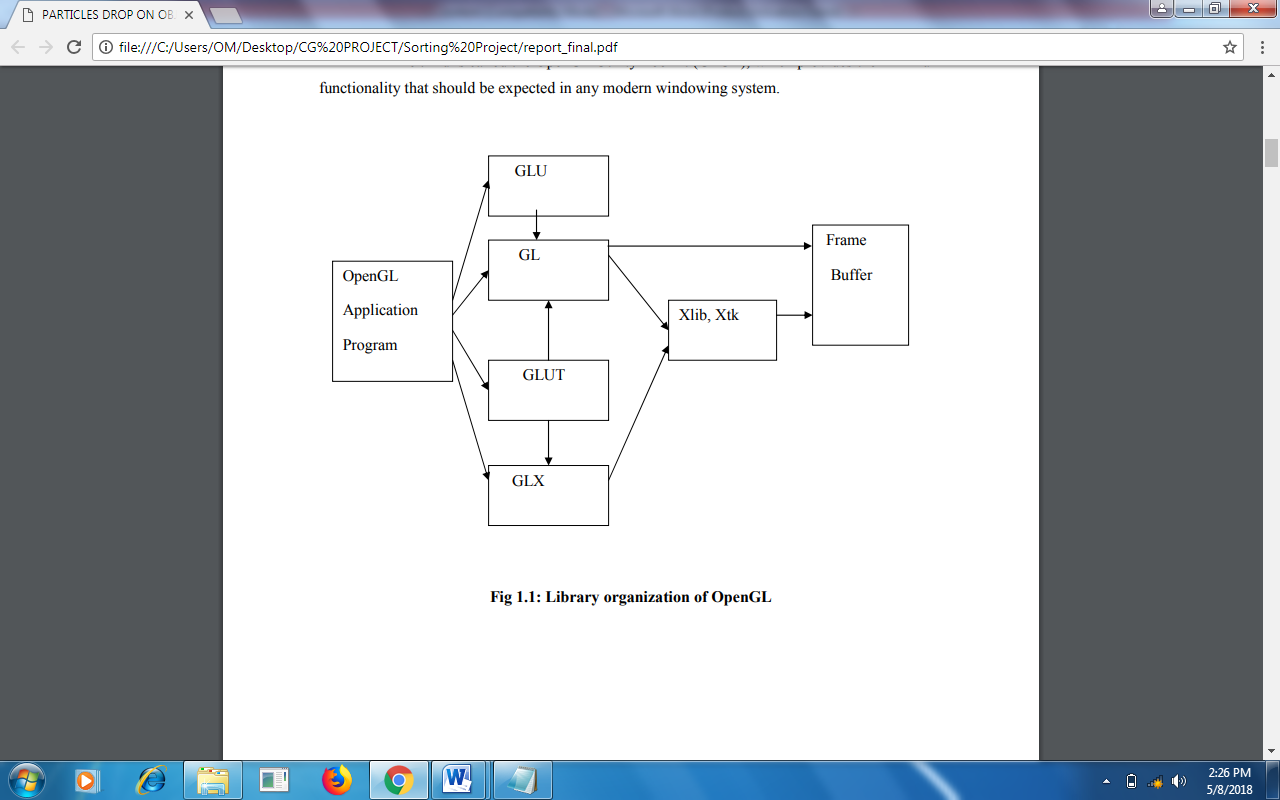
The modification or interpretation of existing pictures, such as photographs and TV scans, is called image processing. In computer graphics, a computer is used to create a picture. Image-processing techniques, on the other hand, are used to improve picture quality, analyze images, or recognize visual patterns for robotics applications. However, image-processing methods are often used in computer graphics, and computer-graphics methods are frequently applied in image processing. Typically, a photograph or other picture is digitized into an image file before image-processing methods are employed. Then digital methods can be used to rearrange picture parts, to enhance color separations, or to improve the quality of shading OpenGL (Open Graphics Library) is a standard specification defining a cross-language, cross-platform API for writing applications that produce [2D](http://en.wikipedia.org/wiki/2D_computer_graphics) 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.](http://en.wikipedia.org/wiki/Silicon_Graphics) (SGI) in 1992 and is widely used in CAD, virtual reality, scientific visualization, information visualization, and flight simulation [1].

**OpenGL**

OpenGL has become a widely accepted standard for developing graphics application. Most of our applications will be designed to access OpenGL directly through functions in three libraries. Functions in main GL library have names that begin with the letters gl and are stored in a library usually referred to as GL.

The second is the OpenGL Utility Library (GLU). This library uses only GL functions but contains code for creating common objects and simplifying viewing. All functions in GLU can be created from the core GL library. The GLU library is available in all OpenGL implementations; functions in the GLU library begin with the letters glu.

The third is called the OpenGL Utility Toolkit (GLUT), which provides the minimum functionality that should be expected in any modern windowing system.



**Fig 1.1: Library organization of OpenGL**

* 1. **OVERVIEW OF THE PROJECT**
     1. **PROBLEM STATEMENT**

To develop and demonstrate a simple Satellite Communication.

**The various concepts used in this application are as follows:**

**Transformations:**

A transformation is a function that takes a point or vector and maps that point or vector into another point or vector. In this application the transformations used are 3D translation, 3D rotation and 3D scaling.

**Light models and Perspective viewing:**

The application uses lighting effects, shade models, textures and perspective viewing.

**The main features of the project are as follows**

* The application is user friendly, thereby it can be used by children and who do not know computer in depth too.
* As provision for future development.
* It has many options of keyboard functions.
* It has a good visual effect.
  1. **AIM OF THE PROJECT**

Main aim of this Mini Project is to illustrate the concepts of working of a Satellite in OpenGL. Satellites are used for a large number of purposes. Common types include military and civilian Earth observation satellites, communications satellites, navigation satellites, weather satellites, and research satellites. SolidCube is used for forming a complete network setup which helps to understand concept of Congestion Control very well. Small SolidCube is used to represent a data, which travels as data transfer from source to destination.

**CHAPTER 2**

**REQUIREMENT SPECIFICATION**

A software requirement definition is an abstract description of the services which the system should provide, and the constraints under which the system must operate. It should only specify the external behavior of the system.

**2.1 FUNCTIONAL REQUIREMENTS**

In software engineering, a **functional requirement** defines a function of a software system or its component. A function is described as a set of inputs, the behavior, and outputs (see also software). Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define *what* a system is supposed to accomplish. Behavioural requirements describing all the cases where the system uses the functional requirements are captured in use cases.

**The various methods used in this project are as follows:-**

* **Display**

The module draws the output on the screen and the functions in it.

* **Mouse**

This module specifies the action corresponding to the mouse

* **Keyboard**

The module specifies the action corresponding to the key board.

**2.2 NON-FUNCTIONAL REQUIREMENTS**

These are constraints on the services or functions offered by the system. They include timing constraints, constraints on the development process and standards. Non-functional requirements often apply to the system as a whole.

Non-functional requirements are as follows:

* Dependability
* Availability
* Reliability
* Safety
* Security

**Dependability:**

The dependability of a computer system is a property of the system that equates to its trustworthiness. Trustworthiness essentially means the degree of user confidence that the system will operate as they expect and that the system will not ‘fail’ in normal use.

**Availability:**

The ability of the system to deliver services when requested. There is no error in the program while executing the program.

**Reliability:**

The ability of the system to deliver services as specified. The program is compatible with all types of operating system without any failure.

**Safety:**

The ability of the system to operate without catastrophic failure. This program is user friendly and it will never affect the system.

**Security:**

The ability of the system to protect itself against accidental or deliberate intrusion.

**2.3 DETAILS OF THE SOFTWARE**

Here, the coding of our project is done in Microsoft Visual C++ which is a commercial integrated development environment (IDE) with OpenGL (Open Graphics Library) which is a standard specification to produce 2D and 3D computer graphics. We use the OpenGL Utility Toolkit called GLUT which is a library of utilities for OpenGL programs.

**2.3.1 Microsoft Visual C++**

Microsoft Visual C++ is a commercial [integrated development environment](http://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) product engineered by [Microsoft](http://en.wikipedia.org/wiki/Microsoft) for the [C](http://en.wikipedia.org/wiki/C_(programming_language)), [C++](http://en.wikipedia.org/wiki/C%2B%2B) and [C++/CLI](http://en.wikipedia.org/wiki/C%2B%2B/CLI) [programming languages](http://en.wikipedia.org/wiki/Programming_language). It has tools for [developing](http://en.wikipedia.org/wiki/Software_development_process) and [debugging](http://en.wikipedia.org/wiki/Debugging) C++ code, especially code written for the [Microsoft Windows](http://en.wikipedia.org/wiki/Microsoft_Windows) [API](http://en.wikipedia.org/wiki/Application_programming_interface), OpenGL API, the [DirectX](http://en.wikipedia.org/wiki/DirectX) [API](http://en.wikipedia.org/wiki/API) and the [Microsoft .NET](http://en.wikipedia.org/wiki/Microsoft_.NET) Framework.

**2.3.2 OpenGL and GLUT**

OpenGL (Open Graphics Library) is a standard specification defining a cross-language, [cross-platform](http://en.wikipedia.org/wiki/Cross-platform) [API](http://en.wikipedia.org/wiki/Application_programming_interface) for writing applications that produce [2D](http://en.wikipedia.org/wiki/2D_computer_graphics) and [3D computer graphics](http://en.wikipedia.org/wiki/3D_computer_graphics), describing a set of functions and the precise behaviors that they must perform. From this specification, hardware vendors create implementations - libraries of functions created to match the functions stated in the OpenGL specification, making use of hardware acceleration where possible. Hardware vendors have to meet specific tests to be able to qualify their implementation as an OpenGL implementation.

GLUT 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.

**2.4 SOFTWARE REQUIREMENTS**

* OPERATING SYSTEM **:** Windows 7, Windows 8.1, Windows 10
* FRONT END **:** Microsoft Visual Studio 2010
* CODING LANGUAGE **:** C++

**2.5 HARDWARE REQUIREMENTS**

* SYSTEM : Pentium IV 2.4 GHz or above
* HARD DISK : 40 GB, 80 GB, 160 GB or above
* MONITOR : 15 VGA colour
* RAM : 256 MB, 512 MB, 1 GB or above

**Chapter 3**

**Design**

Data flow design is as shown below - covering the flow of the data in the system. It describes the relation between user input and the system behavior.

**Main**

**Initialize openGL callback function**

**Hold the appropriate key or mouse button to see the corresponding action being implemented on the screen**

Key or mouse Key or mouse

mouse button pressed button released

**No action takes place**

**The appropriate**

**action occurs and is shown on screen**

**End**

**Figure 3.1: Flowchart for representing flow of execution of Solar System**

**CHAPTER 4**

**Implementation**

To implement the Current system we have used different functions of our project which are as follows:

**4.1 BUILT IN FUNCTIONS**

* **void glClear (GLbitfield*mask*);**
* ***mask*** *–* Bitwise OR of masks that indicate the buffers to be cleared. The four masks are GL\_COLOR\_BUFFER\_BIT, GL\_DEPTH\_BUFFER\_BIT, GL\_ACCUM\_BUFFER\_BIT and GL\_STENCIL\_BUFFER\_BIT.
* It clears buffers to preset values.
* **void glClearColor (GLclampf *red*, GLclampf *green*, GLclampf *blue,* GLclampf *alpha*);**
* *red, green, blue, alpha* – specify the red, green, blue and alpha values used when the colour buffers are cleared. The initial values are all 0.
* It specifies clear values for the colour buffers.
* **void glColor3f (GLfloat *red*, GLfloat *green,* GLfloat *blue*);**
* *red, green, blue –* specify new red, green, and blue values for the current colour.
* It sets the current color.
* **glutCreateWindow (char \**name*);**
* *name –* ASCII character string for use as window name.
* It creates a top-level window.
* **void glutDisplayFunc (void (*\*func*) (void));**
* *func* – the new display callback function.
* It sets the display callback for the current window.
* **void glutInitWindowSize (int *width,* int *height*);**
* width– width in pixels.
* height – height in pixels.
* It is used to set the initial window size
* **void glutMainLoop (void);**
* It enters the GLUT event processing group. This routine should be called at most once in a GLUT program. Once called, this routine will never return. It will call as necessary any callbacks that have been registered.
* **void glutPostRedisplay (void);**
* It marks the current window as needing to be redisplayed.
* **void display (void);**
* It contains the function definition for display callback.
* **glshadeModel(GL\_SMOOTH);**
* To enable the smooth shading we set the shade as the above function
* **glEnable(GL\_DEPTH\_TEST);**
* 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.
* **glMaterialfv(GLenum face,GLenum type,GLfloat \*pointer\_to\_array); glMaterialf(GLenum face,GLenum type,GLfloat value);**
* We can specify different material properties for the front and back faces of a surface through the above functions
* **glLightfv(GLenum source,GLenum parameter,GLfloat \*pointer\_to\_array); glLightf(GLenum source,GLenum parameter,GLfloat value);**
* This function is used to enable a light source. The function specifies the required vector and scalar parameters to enable a light source.
* **void glRotatef(GLfloat angle, GLfloat x, GLfloat y, GLfloat z);**
* *angle* – Specifies the rotation in degrees
* *x, y, z* – Specifies the x, y and z coordinates of a vector respectively
* **void glTranslatef(GLfloat x, GLfloat y, GLfloat z);**
* *x, y, z* – Specify the x, y and z coordinates of a translation vector
* **void glutSolidSphere(GLdouble radius,GLint slices, GLint stacks);**
* radius - The radius of the sphere.
* slices - The number of subdivisions around the Z axis (similar to lines of longitude).
* stacks - The number of subdivisions along the Z axis (similar to lines of latitude).
* Renders a sphere centered at the modelling coordinates origin of the specified radius. The sphere is subdivided around the Z axis into slices and along the Z axis into stacks.
* **void gluLookAt(GLdouble eyeX, GLdouble eyeY, GLdouble eyeZ, GLdouble centerX, GLdouble centerY, GLdouble centerZ, GLdouble upX, GLdouble upY, GLdouble upZ);**
* Parameters eyeX, eyeY, eyeZ - Specifies the position of the eye point.
* centerX, centerY, centerZ - Specifies the position of the reference point.
* upX, upY, upZ - Specifies the direction of the up vector.
* gluLookAt creates a viewing matrix derived from an eye point, a reference point indicating the center of the scene, and an UP vector.
* **void glPushMatrix(void);**
* glPushMatrix pushes the current matrix stack down by one, duplicating the current matrix
* **void glPushMatrix(void);**
* glPushMatrix pushes the current matrix stack down by one, duplicating the current matrix

**CHAPTER 5**

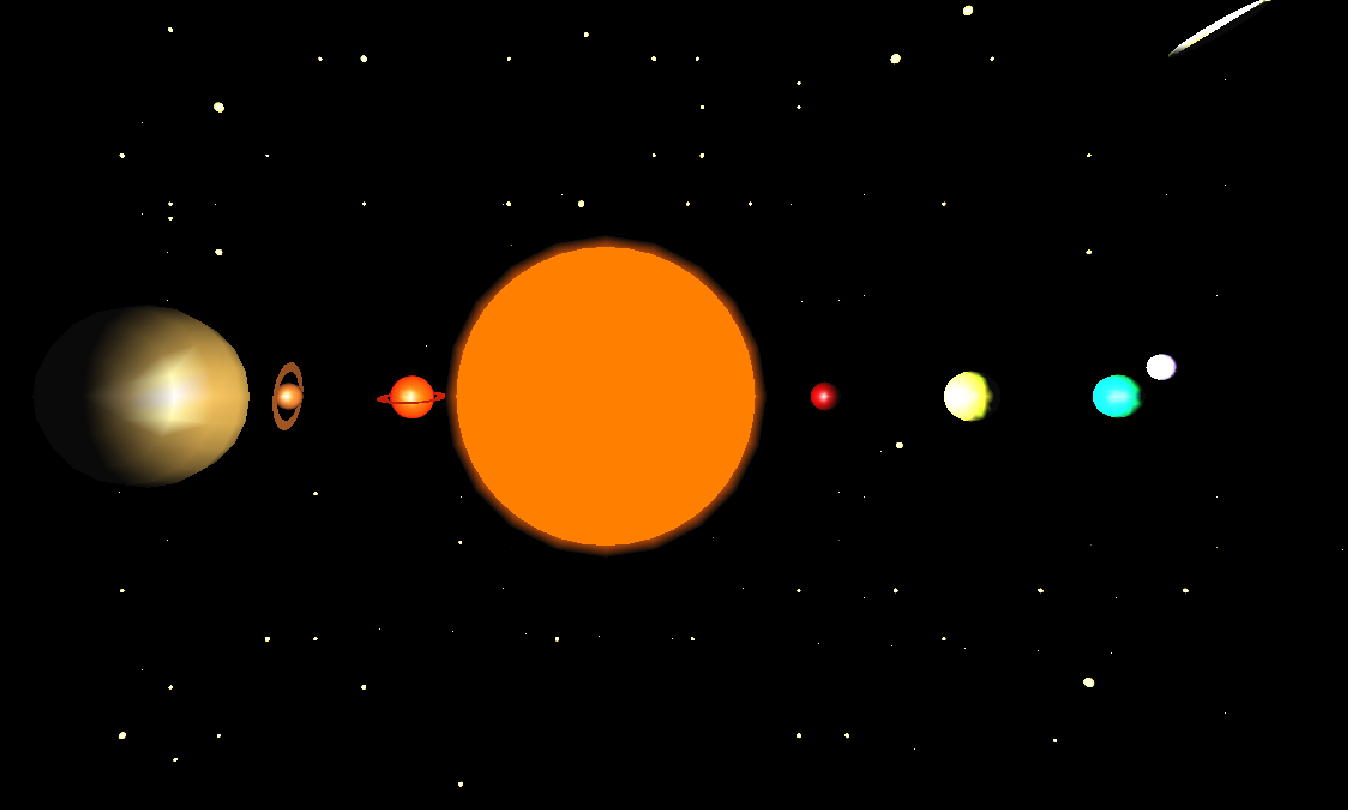
**TESTING**

Testing has been conducted as tabulated below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Functions** | **Expected result** | **Actual result** | **Comment** |
| 1 | Keyboard function | The keys m, v, e, r, j, u, s, n are used to rotate the planets. | The planets rotate about their initial position. | PASS |
| 2 | Keyboard function | The keys M, V, E, R, J, U, S, N are used to revolve the planets around the Sun. | The planets revolve around the Sun. | PASS |
| 4 | Keyboard function | The key z rotates the Sun, B gives both the rotation and revolution of the planets, rotation of the Sun, comet revolution, stars twinkling, A revolves all the planets, a rotates all the planets and the Sun, with the stars twinkling, b makes the stars twinkle and c performs revolution of the comet. | All corresponding holding of the keys should show their corresponding actions. | PASS |
| 5 | Mouse function | When left mouse button is clicked and held, the planets and the comet rotate and revolve in anticlockwise direction.  When right or middle mouse button is clicked and held, the planets and the comet rotate and revolve in clockwise direction. | The planets and the comet should move accordingly. | PASS |

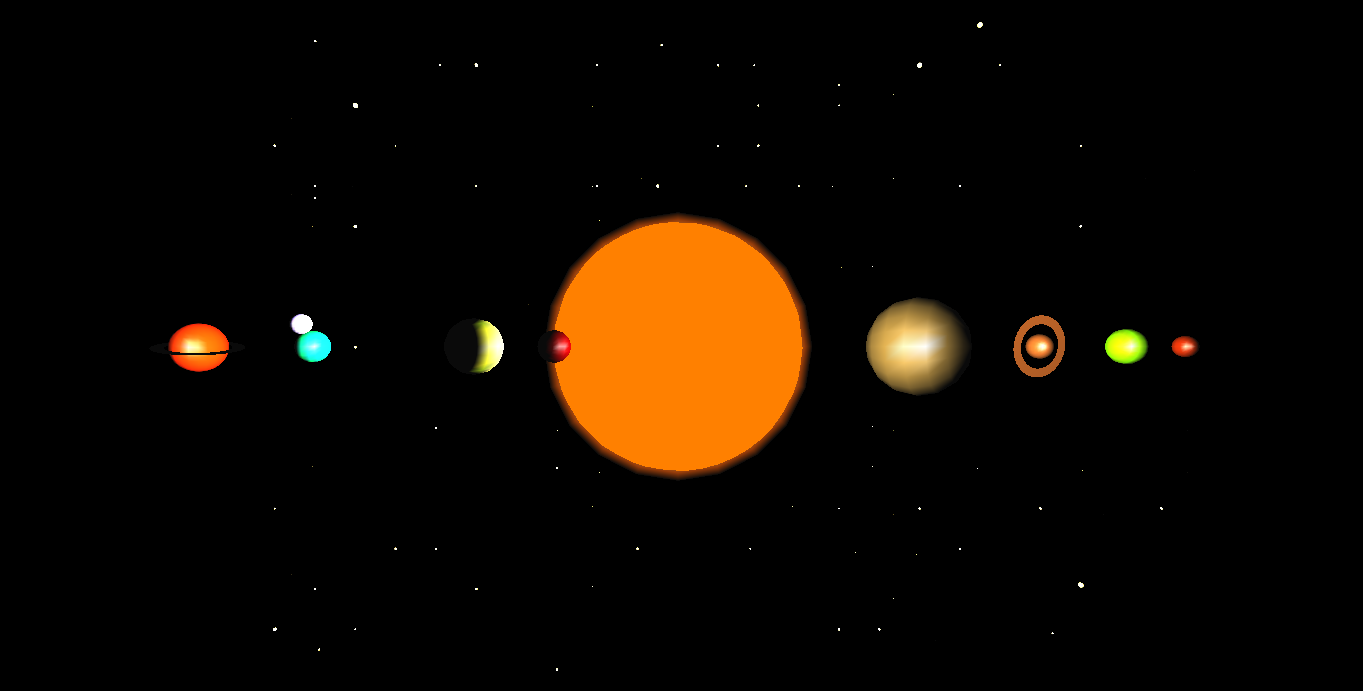
**CHAPTER 6**

**RESULTS AND SCREENSHOTS**

****

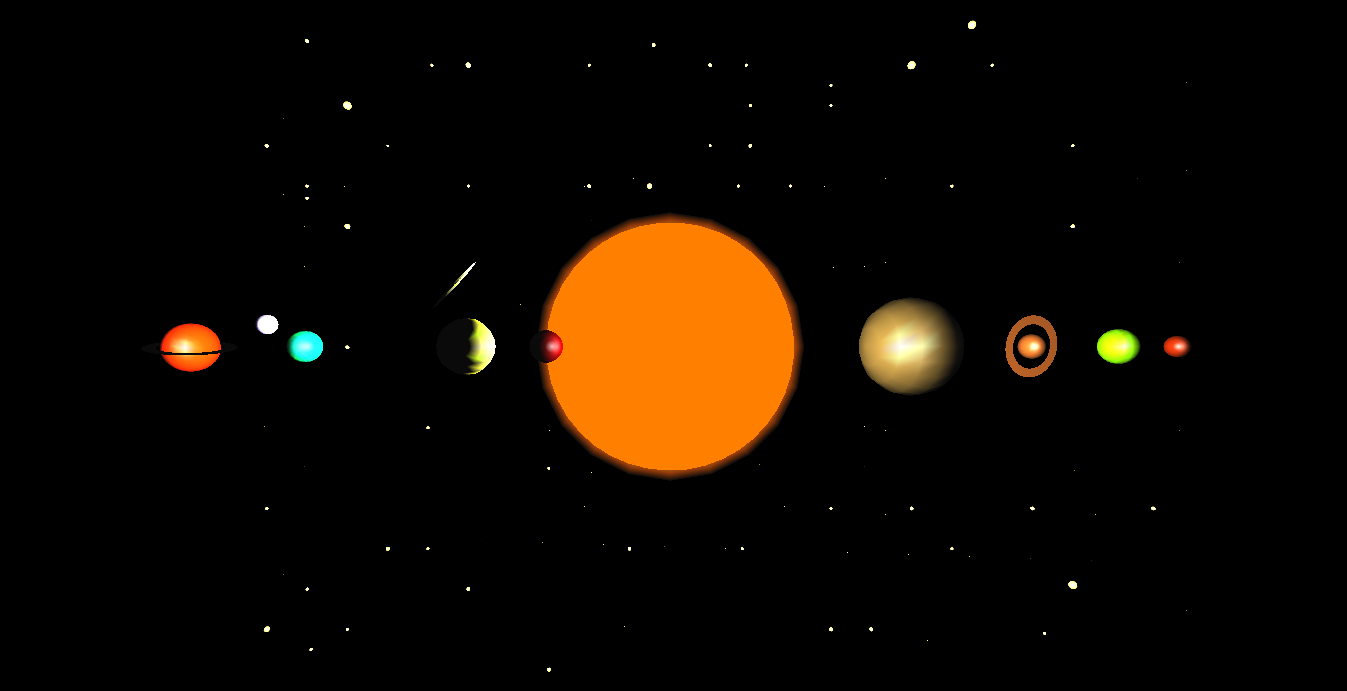
**Figure 6.1: Implementation of Revolution**

Figure 6.1 shows revolution of the planets being implemented. Due to the planets being at varying positions in their revolution, not all planets are visible.

****

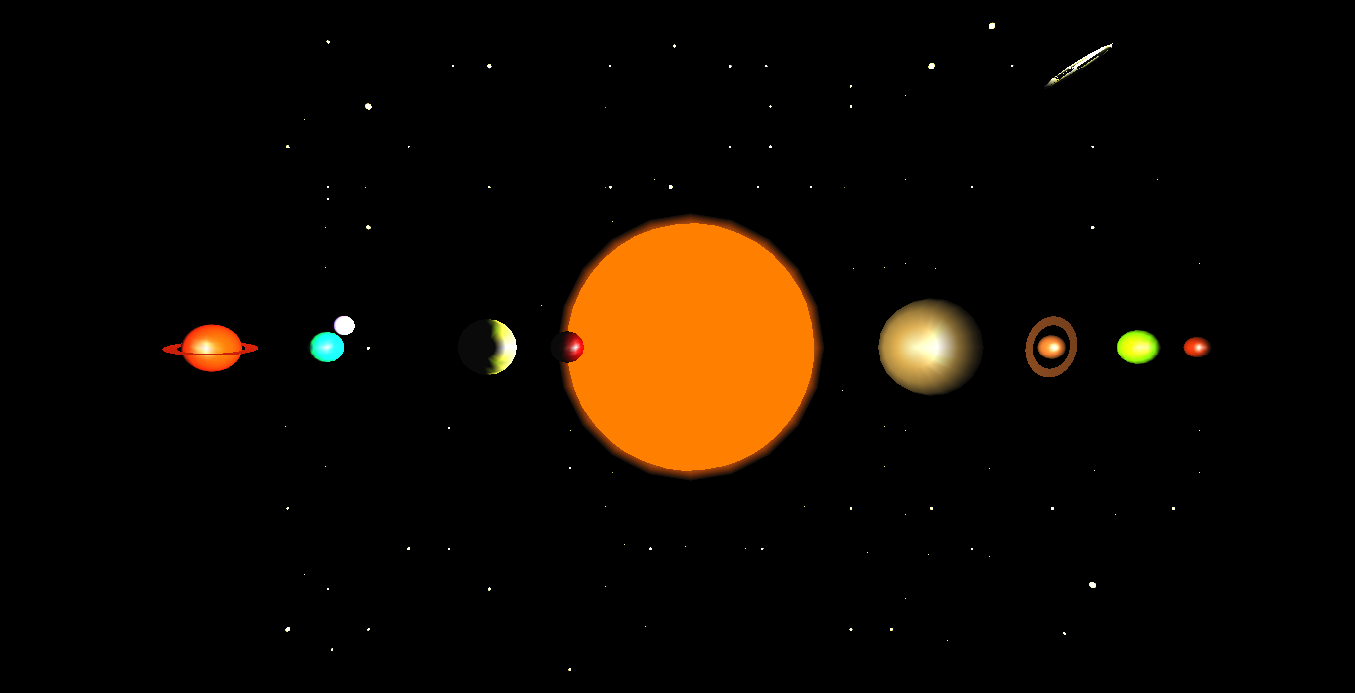
**Figure 6.2: Positioning of planets by revolving them to a position**

Figure 6.2 shows the planets being positioned along a line, so that all the planets are visible. This has been achieved by controlled revolution of the planets.

****

**Figure 6.3: Movement of the comet**

Figure 6.3 shows the comet moving from the lower left of the screen across the screen tracing a diagonal path.

****

**Figure 6.4: Rotation of the planets**

Figure 6.4 shows the rotation of the planets. Since the planets are rotating about their axis at a fixed position, it is difficult to identify rotation. However, we can make out the Earth’s rotation since the Moon also revolves around the Earth as it rotates. Comparing Figure 6.3 and 6.4, we can see that the Earth remains static, but the Moon’s position has changed, which clearly indicates that the Earth is rotating.

**CHAPTER 7**

**CONCLUSION**

This project has been developed using the OpenGL API, through the use of Microsoft Visual Studio. This project is an attempt to represent a real world entity using OpenGL functions. By developing the project using OpenGL, we are trying to understand the application of some basic in-built functions and their working. Although not as complex and thorough as large-scale graphics projects, this project is aimed at the dual-purpose of learning how the world around us works and how graphics are implemented

**Bibliography**

**Reference 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

**Websites:**

* [www.google.com](http://www.google.com)
* [www.youtube.com](http://www.youtube.com)
* [www.wikipedia.com](http://www.wikipedia.com)
* [www.opengl.org](http://www.opengl.org)