**CHAPTER 1**

**INTRODUCTION**

The Satellite Communication system using OpenGL is used to represent the working of a satellite i.e., the revolution of satellite around the earth, transmission and receiving of data packets between base stations on earth.

Computer graphic**s** 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**:

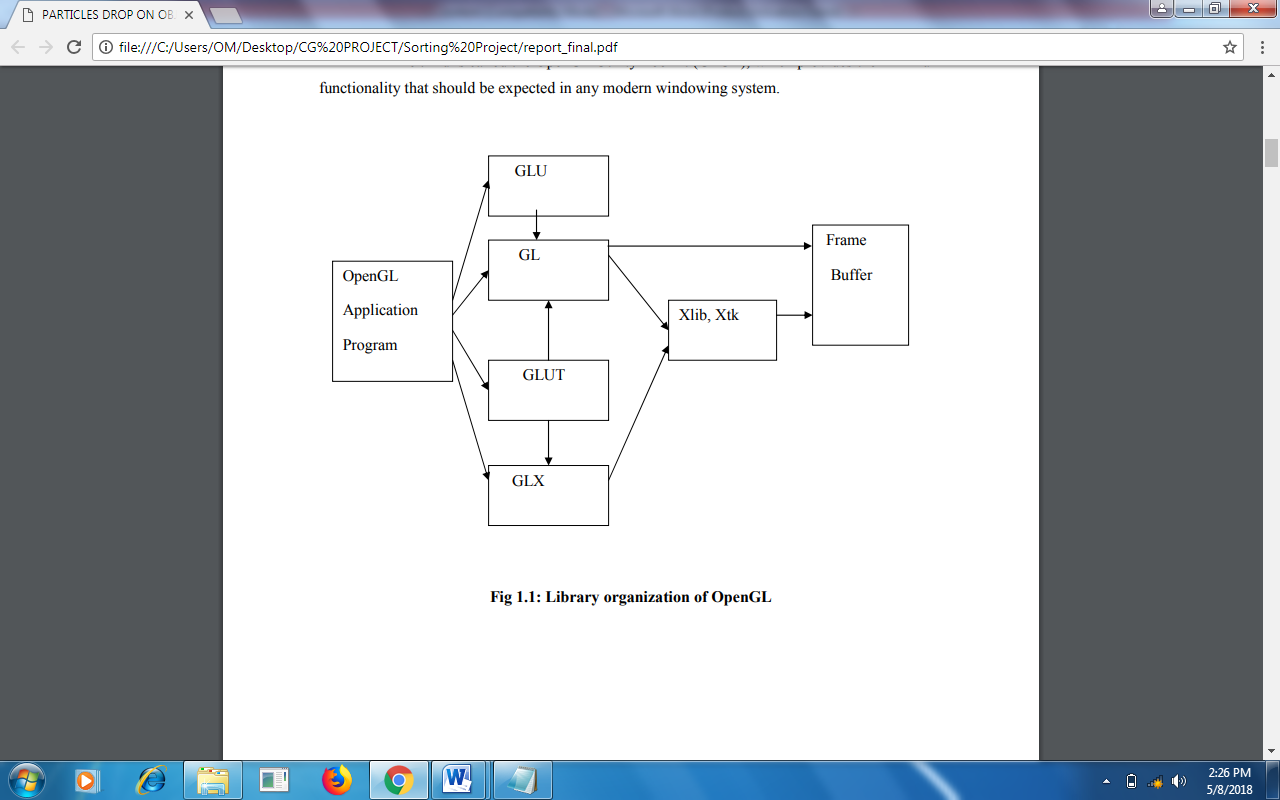
The modification or interpretation of existing pictures, such as photographs and TV scans, is called image processing. In computer graphics, a computer is used to create a picture. Image-processing techniques, on the other hand, are used to improve picture quality, analyze images, or recognize visual 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 design develop and demonstrate a simple Satellite Communication using OpenGL functions.

**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**

The main aim of this system is to demonstrate the working of simple satellite communication system controlling the congestion and transmits the data packets to the various base stations.

**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. Behavioral 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

**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 this system to deliver services when requested. There is no error 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 during or after the runtime.

**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.

START

INITIALIZE OPENGL callback function

KEYBOARD FUNC

MOUSE FUNC

On pressing

S button on keyboard

Right click mouse button

Start Revolution

Display Menu

On pressing q

On keyboard

On pressing keyboard

Quit

Functions

Quit

Revolve – s/S

Transmit – t

Quit - q

Start Revolution

Transmit

Stop

END

**Figure 3.1: DataFlow Diagram for representing flow of execution of Satellite Communication**

**CHAPTER 4**

**Implementation**

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

* 1. **BUILT IN FUNCTIONS**

**Various functions used in this program.**

* **glutInit()** : interaction between the windowing system and OPENGL is initiated
* **glutInitDisplayMode()** : used when double buffering is required and depth information is required
* **glutCreateWindow()** : this opens the OPENGL window and displays the title at top of the window
* **glutInitWindowSize()** : specifies the size of the window
* **glutInitWindowPosition()** : specifies the position of the window in screen co-ordinates
* **glutKeyboardFunc()** : handles normal ascii symbols
* **glutSpecialFunc()** : handles special keyboard keys
* **glutReshapeFunc()** : sets up the callback function for reshaping the window
* **glutIdleFunc()** : this handles the processing of the background
* **glutDisplayFunc()** : this handles redrawing of the window
* **glutMainLoop()** : this starts the main loop, it never returns
* **glViewport()** : used to set up the viewport
* **glVertex3fv()** : used to set up the points or vertices in three dimensions
* **glColor3fv()** : used to render color to faces
* **glFlush()** : used to flush the pipeline
* **glutPostRedisplay()** : used to trigger an automatic redrawal of the object
* **glMatrixMode()** : used to set up the required mode of the matrix
* **glLoadIdentity()** : used to load or initialize to the identity matrix
* **glTranslatef()** : used to translate or move the rotation centre from one point to another in three dimensions
* **glRotatef()** : used to rotate an object through a specified rotation angle
  1. **USER DEFINED FUNCTIONS**
* **satellite() :**
* It is used to construct solar panels for transmitting and receiving signals
* **building() :**
* It is used to construct main structure of building having windows in form of quadrilateral with dish on top acting as a receiver.
* **waves() :**
* It is used to construct signal waves generated from dish to satellite and vice versa

**CHAPTER 5**

**TESTING**

Testing has been conducted as tabulated below:

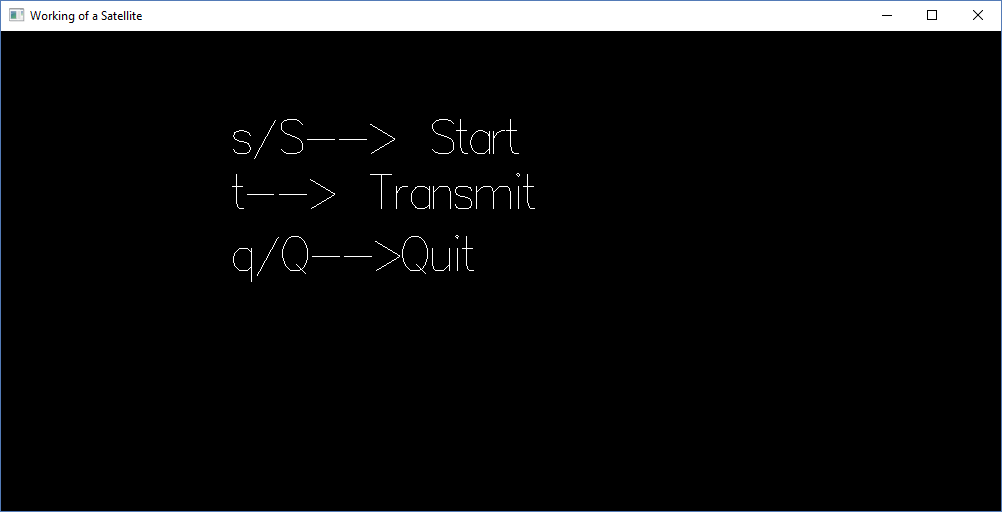
* 1. **TESTCASE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Test Key** | **Expected Output** | **Actual Output** | **Result** |
| **1.1** | **S/s** | **Revolution of Satellite.** | **Revolution of Satellite.** | **PASS** |
| **1.2** | **S/s** | **Revolution of Satellite.** | **Code Crash.** | **FAIL** |
| **2.1** | **t** | **Transmission of Data Packets.** | **Transmission of Data Packets.** | **PASS** |
| **2.1** | **t** | **Transmission of Data Packets.** | **Revolution of Satellite.** | **FAIL** |
| **3** | **q** | **Termination of Program** | **Termination of Program** | **PASS** |

**Table 1: Test case results**

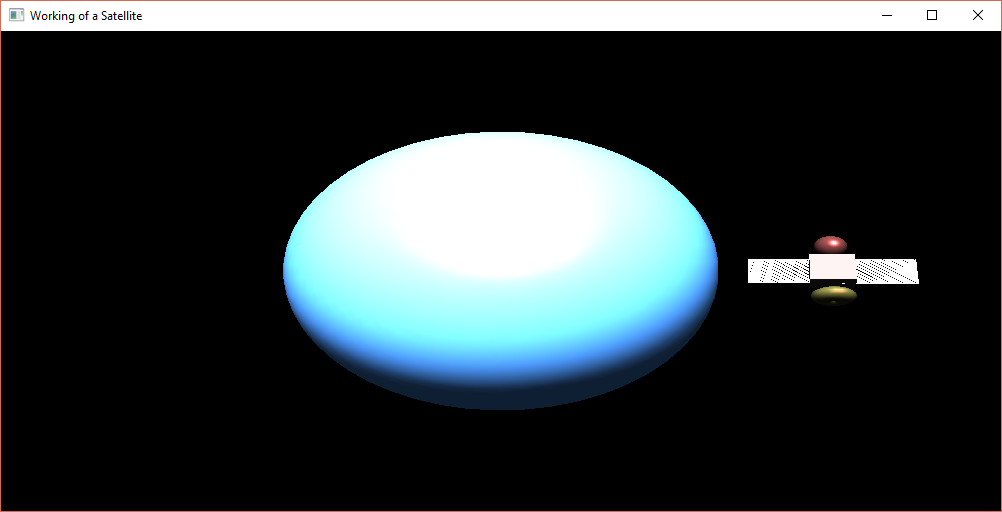
**CHAPTER 6**

**RESULTS AND SCREENSHOTS**

****

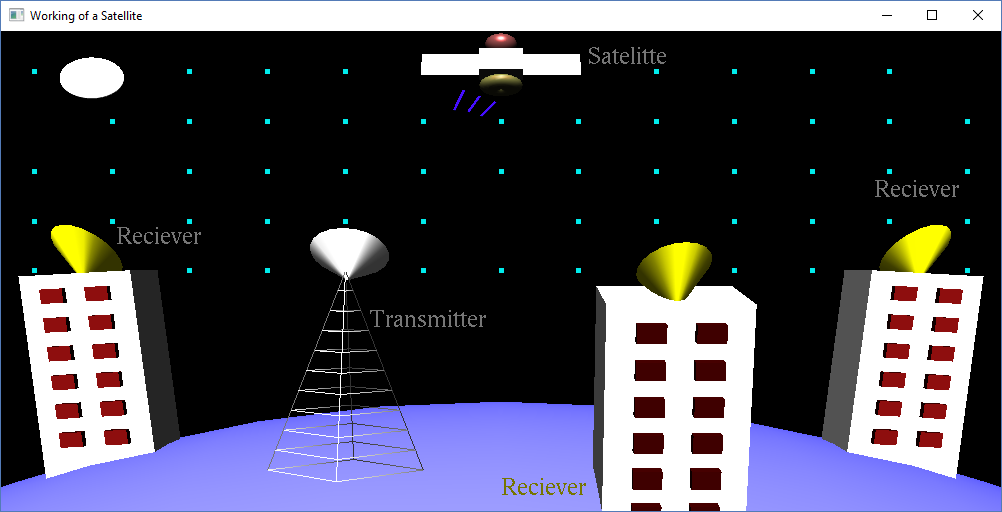
**Figure 6.1: Initial display screen**

Figure 6.1 shows initial input screen indicating various keyboard options to navigate through the system.



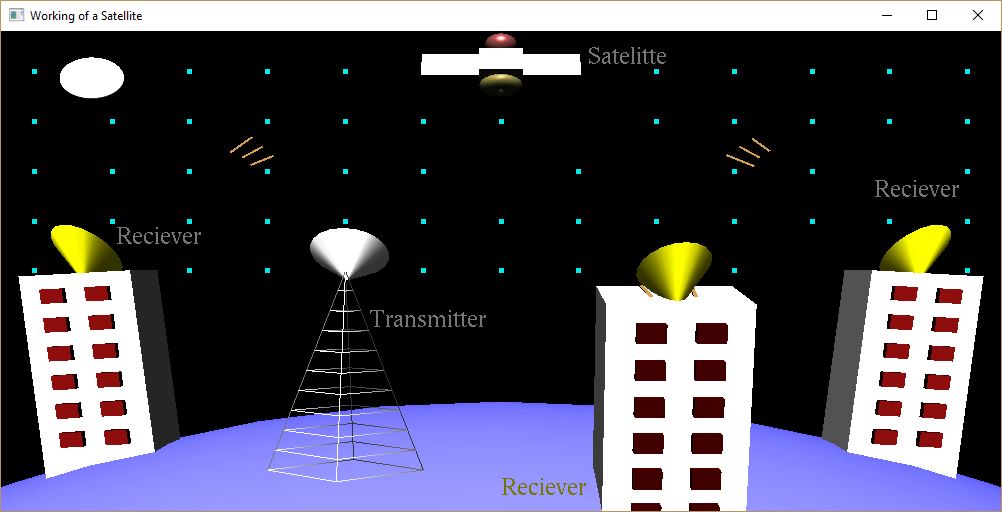
**Figure 6.2: Planet with an artificial satellite revolving around it.**

Figure 6.2 shows the artificial satellite revolving around the planet in clockwise direction.

****

**Figure 6.3: Transmission of signals from transmitter to satellite**

Figure 6.3 shows the positioning of various base stations including transmitters and receivers. The satellite receives the signals from the transmitter.



**Figure 6.4: Transmission of signals from satellite to receiver**

Figure 6.4 shows the transmission of amplified signals from the satellite to various receivers at the base station. Thus communication is achieved.

**CHAPTER 7**

**CONCLUSION**

Satellite Communication System has been developed using the OpenGL API, through the use of Microsoft Visual Studio. It is an attempt to represent a real world entity using OpenGL functions. By developing this system using OpenGL, a try to understand the application of some basic in-built functions and their working. Although not as complex and thorough as large-scale graphics projects, the system is aimed at the dual-purpose of learning how the satellite communication works and how graphics are implemented. The complex procedure of satellite communication is demonstrated in a simple understandable way implementing graphics.

**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)