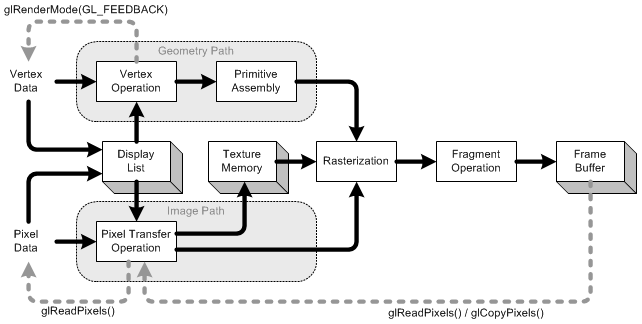
**CHAPTER 1**

**INTRODUCTION**

This report contains implementation of ‘**3D SATELLITE COMMUNICATION ‘**using a set of OpenGL functions. The project consists of different views for 3D bicycle. We are mainly using keyboard and mouse as interface to view the bicycle, to accelerate, control it’s movements. The objects are drawn by using GLUT functions. This project has been developed using **Windows 7** with OpenGL package.



**Fig** **1.1:OpenGL rendering pipeline.**

**1.1 Computer Graphics**

Graphics provides one of the most natural means of communicating within a computer, since our highly developed 2D and 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly and effectively. Interactive computer graphics is the most important means of producing pictures since the invention of photography and television. It has the added advantage that, with the computer, we can make pictures not only of concrete real world objects but also of abstract, synthetic objects, such as mathematical surfaces and of data that have no inherent geometry, such as survey results.

Computer graphics started with the display of data on hardcopy plotters and cathode ray tube screens soon after the introduction of computers themselves. It has grown to include the creation, storage, and manipulation of models and images of objects. These models come from a diverse and expanding set of fields, and include physical, mathematical, engineering, architectural, and even conceptual structures, natural phenomena, and so on. Computer graphics today is largely interactive. The user controls the contents, structure, and appearance of the objects and of their displayed images by using input devices, such as keyboard, mouse, or touch-screen. Due to close relationships between the input devices and the display, the handling of such devices is included in the study of computer graphics. The advantages of the interactive graphics are many in number. Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D patter-recognition abilities allow us to perceive and process data rapidly and efficiently. In many design, implementation, and construction processes today, the information pictures can give is virtually indispensable. Scientific visualization became an important field in the 1980s when

scientists and engineers realized that they could not interpret the prodigious quantities of data produced in supercomputer runs without summarizing the data and highlighting trends and phenomena in various kinds of graphical representations.

**1.2 OpenGL Interface**

OpenGL is an application program interface (API) offering various functions to implement primitives, models and images. This offers functions to create and manipulate render lighting, colouring, viewing the models. OpenGL offers different coordinate system and frames. OpenGL offers translation, rotation and scaling of objects.

Most of our applications will be designed to access OpenGL directly through functions in three libraries. They are:

1. Main GL: Library has names that begin with the letter *gl* and are stored in a library usually referred to as GL.

2. OpenGL Utility Library (GLU): This library uses only GL functions but contains code for creating common objects and simplifying viewing.

3. OpenGL Utility Tool kit (GLUT): This provides the minimum functionality that should be accepted in any modern windowing system.

**1.3 Overview**

* OpenGL (Open Graphics Library) is the interface between a graphic program and graphics hardware. *It is streamlined*. In other words, it provides low-level functionality. For example, all objects are built from points, lines and convex polygons. Higher level objects like cubes are implemented as six four-sided polygons.
* OpenGL supports features like 3-dimensions, lighting, anti-aliasing, shadows, textures, depth effects, etc. *It is system-independent*. It does not assume anything about hardware or operating system and is only concerned with efficiently rendering mathematically described scenes. As a result, it does not provide any windowing capabilities.
* *It is a state machine*. At any moment during the execution of a program there is a current model transformation.
* *It is a rendering pipeline*. The rendering pipeline consists of the following steps:
* Defines objects mathematically.
* Arranges objects in space relative to a viewpoint.
* Calculates the colour of the objects.
* Rasterizes the objects.

Graphics provides one of the most natural means of communicating with a computer,

Since our highly developed 2D and 3D pattern-recognition abilities allow us to perceive and

Process pictorial data rapidly and efficiently. Interactive computer graphics is the most

important means of producing pictures since the invention of photography and television. It

has the added advantage that, with the computer, we can make pictures not only of concrete

real world objects but also of abstract, synthetic objects, such as mathematical surfaces and

of data that have no inherent geometry, such as survey results.

OpenGL (open graphics library) is a standard specification defining a cross language

cross platform API for writing applications that produce 2D and 3D computer graphics.

OpenGL was developed by silicon graphics Inc. (SGI) in 1992 and is widely used in CAD,

virtual reality, scientific visualization, information visualization and flight simulation. It is

also used in video games.

OpenGL serves two main purposes:

* To hide the complexities of interfacing with different 3D accelerators, by

presenting programmer with a single, uniform API

* To hide the differing capabilities of hardware platforms, by requiring that all

Implementations support the full openGL, feature set.

OpenGL has historically been influential on the development of 3D accelerator, promoting a base level of functionality that is now common in consumer level hardware:

* Rasterized points, lines and polygons are basic primitives.
* A transform and lighting pipeline.
* Z buffering.
* Texture Mapping.
* Alpha
* Blending.

**CHAPTER 2**

**LITERATURE SURVEY**

CG (Computer graphics) started with the display of data on hardcopy plotters and cathode ray tube screens soon after the introduction of computer themselves. It includes the creation, storage, and manipulation of models and images of objects. These models include physical, mathematical, engineering, architectural, and even conceptual or abstract structures, natural phenomena, and so on. Computer Graphics today is largely interactive- the user controls the contents, structure, and appearance of objects and their displayed images by using input devices, such as keyboard, mouse or touch sensitive panel on the screen. Bitmap graphics is used for user-computer interaction. A Bitmap is an ones and zeros representation of points (pixels, short for ‘picture elements’) on the screen. Bitmap graphics provide easy-to-use and inexpensive graphics based applications.

The concept of ‘desktop’ is a popular metaphor for organizing screen space. By means of a window manager, the user can create, position, and resize rectangular screen areas, called windows, that acted as virtual graphics terminals, each running an application. This allowed users to switch among multiple activities just by pointing at the desired window, typically with the mouse. Graphics provides one of the most natural means of communicating with the computer, since our highly developed 2D and 3D pattern – recognition abilities allow us to perceive and process pictorial data rapidly and efficiently. In many design, implementation, and construction processes, the information pictures can give is virtually indispensable.

Computer graphics is the creation and manipulation of pictures with the aid of computers. It is divided into two broad classes:

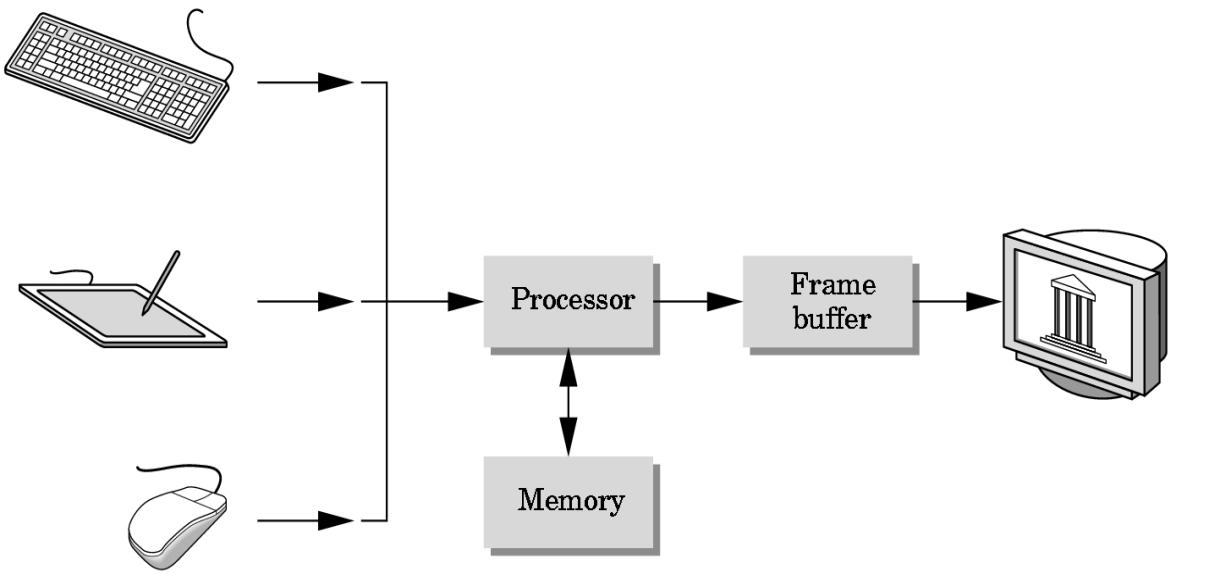
* Non-Interactive Graphics.
* Interactive Graphics.

**2.1 Non-Interactive Graphics**

This is a type of graphics where observer has no control over the pictures produced on the screen. It is also called as Passive graphics.

**2.2 Interactive Graphics**

This is the type of computer graphics in which the user can control the pictures produced. It involves two-way communication between user and computer. The computer upon receiving signal from the input device can modify the displayed picture appropriately. To the user it appears that the picture changes instantaneously in response to his commands. The following fig. shows the basic graphics system:



Input devices Image in FB output devices

**fig 2.1: basic graphics system**

**2.3 About OpenGL**

OpenGL is an open specification for an applications program interface for defining 2D and 3D objects. The specification is cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. It renders 3D objects to the screen, providing the same set of instructions on different computers and graphics adapters. Thus it allows us to write an application that can create the same effects in any operating system using any OpenGL-adhering graphics adapter.

In Computer graphics, a 3-dimensional primitive can be anything from a single point to an ‘n’ sided polygon. From the software standpoint, primitives utilize the basic 3-dimensional rasterization algorithms such as Bresenham's line drawing algorithm, polygon scan line fill, texture mapping and so forth.

OpenGL is a low-level, procedural API, requiring the programmer to dictate the exact steps required to render a scene. OpenGL's low-level design requires programmers to have a good knowledge of the graphics pipeline, but also gives a certain amount of freedom to implement novel rendering algorithms

**CHAPTER 3**

**SYSTEM REQUIREMENTS SPECIFICATION**

**3.1 Hardware Requirements**

* Microprocessor: **1.0 GHz** and above CPU based on either AMD or INTEL Microprocessor Architecture
* Main memory : **2 GB RAM**
* Hard Disk : **40 GB**
* Hard disk speed in RPM:**5400 RPM**
* Keyboard: **QWERTY** Keyboard
* Mouse :**2 or 3** Button mouse
* Monitor : **1024 x 768** display resolution

**3.2 Software Requirements**

* Programming language – C/C++ using OpenGL
* Operating system – Windows 7 and above version operating system
* Compiler – C Compiler
* Graphics library – GL/glut.h
* IDE - Codeblocks / Visual studio

**3.3 Functional Requirements**

**OpenGL API**

If we want to have a control on the flow of program and if we want to interact with

the window system then we use OpenGL API’S. Vertices are represented in the same manner

internally, whether they are specified as two-dimensional or three-dimensional entities,

everything that we do are here will be equally valid in three dimensions. Although OpenGL

is easy to learn, compared with other APIs, it is nevertheless powerful. It supports the simple

three dimensional programs and also supports the advanced rendering techniques.

**GL/glut.h**

We use a readily available library called the OpenGL Utility Toolkit (GLUT), which

provides the minimum functionality that should be expected in any modern windowing system. The application program uses only GLUT functions and can be recompiled with the GLUT library for other window system. OpenGL makes a heavy use of macros to increase code readability and avoid the use of magic numbers. In most implementation, one of the include lines.

**CHAPTER 4**

**DESIGN**

**4.1 Overview**

The aim of this project is to create a **3-D/ VIRTUAL SATELLITE COMMUNICATION**.A satellite is an object that orbits around a larger celestial body, such as a planet or a star. In the context of space exploration and communication, a satellite typically refers to an artificial satellite, which is a human-made object designed to be launched into space and placed in orbit around the Earth or other celestial bodies.So to understand this, our project demonstrate the working so that one can understand the working of the satellite.

**4.2 System Architecture**

The following figure depicts the system architecture of the project:

**Fig 4.2 System Architecture**

**4.3 User Interface**

A set of keys are used to change the following:

* S -> Start the Project
* t/T -> to transmit and receive signals.
* Q-> Quit
* To view the satellite view with help of mouse press left click and choose satellite option.
* To view the city with the help of mouse press right click and choose satellite transmitter and receiver option.
* To quit the program with the help of mouse press right click and choose Quit option

**4.4 Objective**

* The main objective of the SATELLITE COMMUNICATION is to Mini Project is to illustrate the concepts of working of a Satellite in OpenGL.
* A Satellite is an object which has been placed into orbit by human endeavor. Such objects are sometimes called artificial satellites to distinguish them from natural satellites such as the Moon .
* The purpose of the project is to build an application program which gives an animated vision of working of the satellite and to understand how the satellite communicating.

**CHAPTER 5**

**IMPLEMENTATION AND TESTING**

To achieve three dimensional effects, open GL software is proposed. It is software

which provides a graphical interface. It is an interface between application program and

graphics hardware. The advantages are:

1. Open GL is designed as a streamlined.

2. It’s a hardware independent interface i.e it can be implemented on many different

hardware platforms.

3. With Open GL we can draw a small set of geometric primitives such as points,

lines and polygons etc.

4. It provides double buffering which is vital in providing transformations.

5. It is event driven software.

6. It provides call back functions.

**5.1 User Defined Functions**

* **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

**5.2 OpenGL Functions**

* **glColor3f (float, float, float):-**This function will set the current drawing color
* **glClear( ):-**Takes a single argument that is the bitwise OR of several values indicating which buffer is to be cleared.
* **glClearColor ():-**Specifies the red, green, blue, and alpha values used by **glClear** to clear the color buffers.
* **GlLoadIdentity( ):-**the current matrix with the identity matrix.
* **glMatrixMode(mode):-**Sets the current matrix mode, *mode* can be **GL\_MODELVIEW,GL\_PROJECTION or GL\_TEXTURE.**
* **void glutInit (int \*argc, char\*\*argv):-**Initializes GLUT, the arguments from main are passed in and can be used by the application.
* **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 and buffering.
* **void glutInitWindowSize (int width, int height):-** Specifies the initial position of the topleft corner of the window in pixels
* **glutInitCreateWindow (char \*title):-**A window on the display.The string title can be used to label the window. The return value provides references to the window that can be used when there are multiple windows.
* **void glutMouseFunc(void \*f(int button, int state, int x, int y):-**Register the mouse

callback function f. The callback function returns the button,the state of button after the event and the position of the mouse relative to the top-left corner of the window.

* **void glutKeyboardFunc(void(\*func) (void)):-**This function is called every time when you press enter key to resume the game or when you press ‘b’ or ‘B’ key to go back to the initial screen or when you press esc key to exit from the application.
* **void glutDisplayFunc (void (\*func) (void)):-**Register the display function func that is executed when the window needs to be redrawn.
* **void glutSpecialFunc(void(\*func)( void)):-**This function is called when you press the special keys in the keyboard like arrow keys, function keys etc. In our program, the func is invoked when the up arrow or down arrow key is pressed for selecting the options in the main menu and when the left or right arrow key is pressed for moving the object(bicycle) accordingly.
* **glutPostReDisplay ( )** :-which requests that the display callback be executed after the current callback returns.

**5.3 Testing**

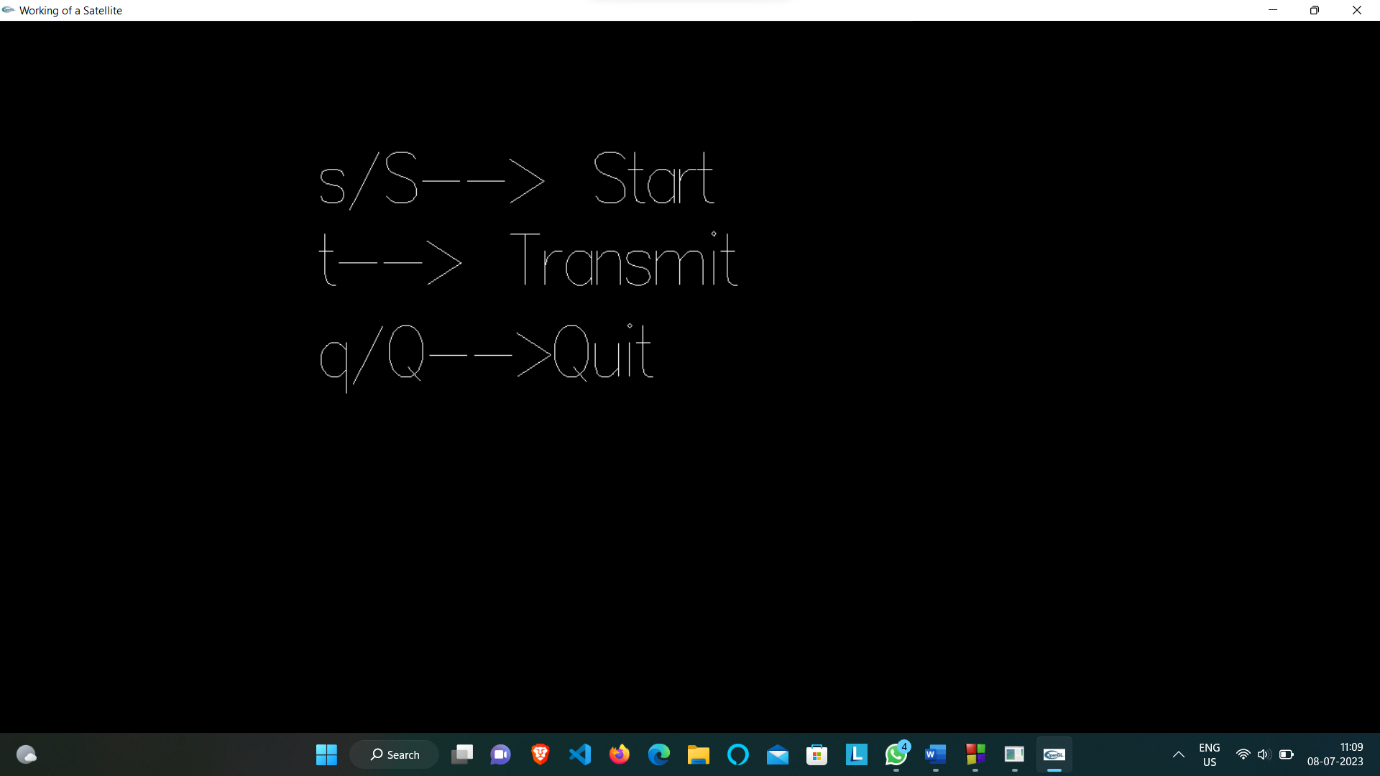
|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Input** | **Output Observed** | **Test Result** |
| 1 | Key Press S or s | Simulation stops | Pass |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |

**Table 5.1: Test Cases**

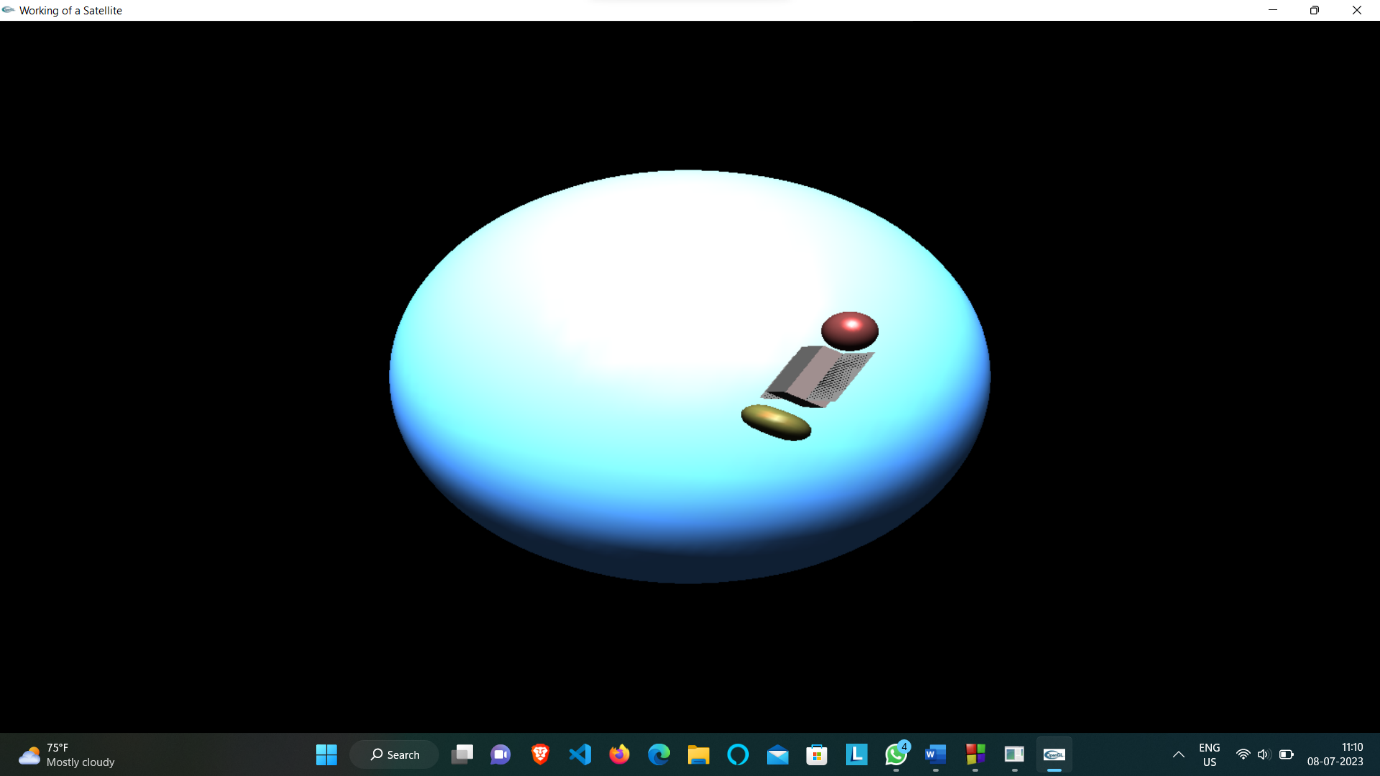
**CHAPTER 6**

**RESULTS**

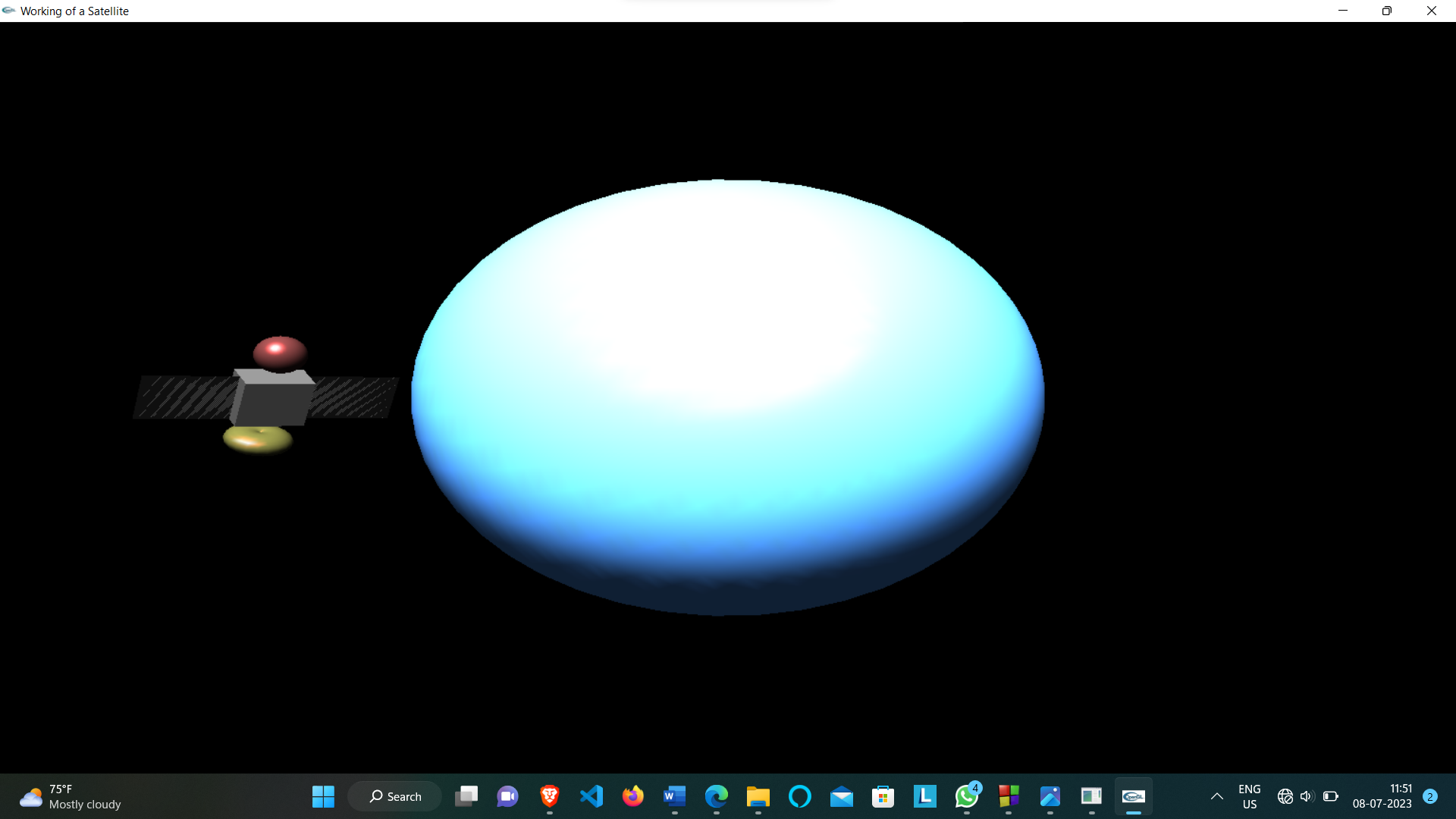
**SNAPSHOTS**



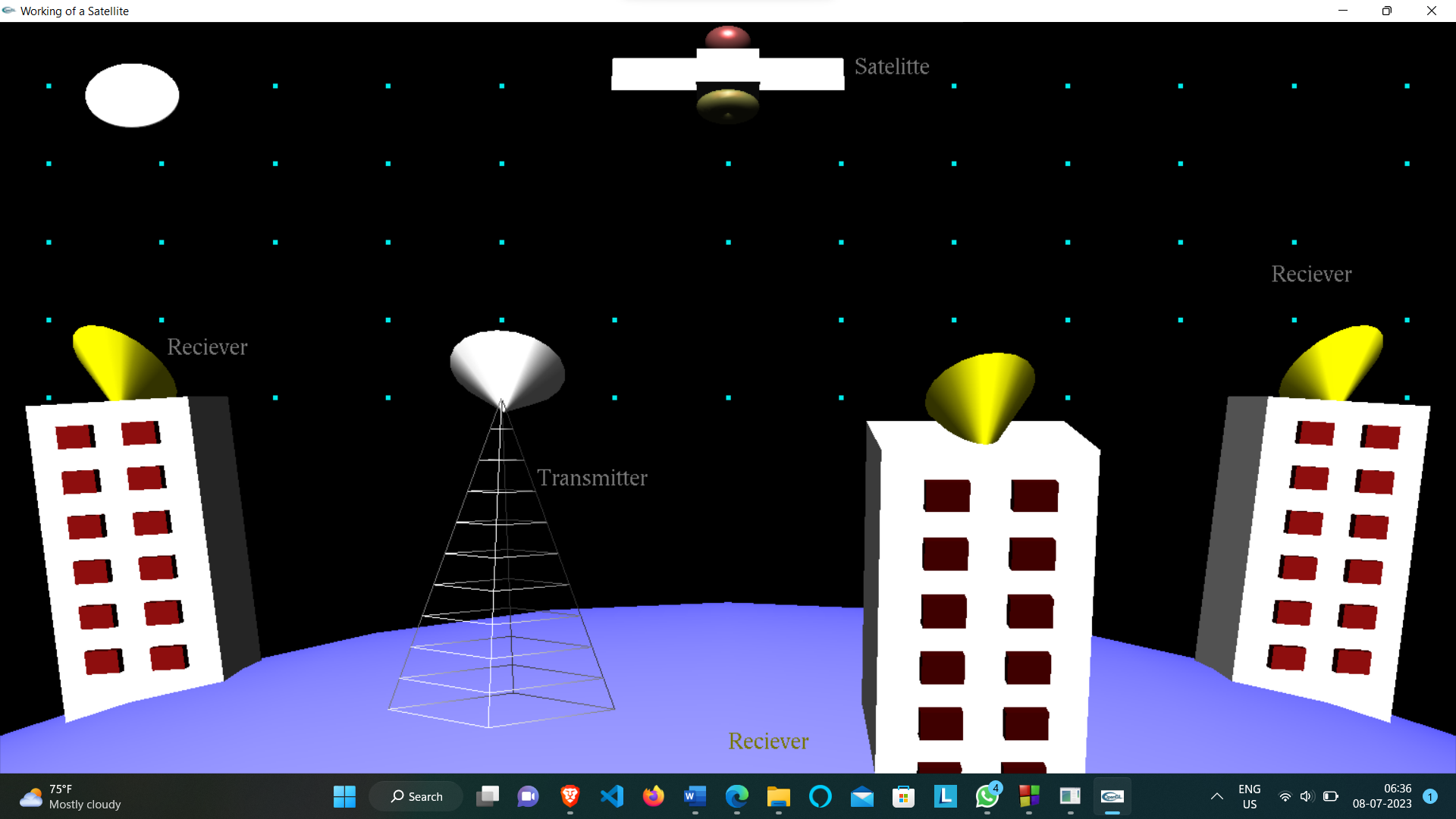
**Fig 6.1: default view when the code is executed**



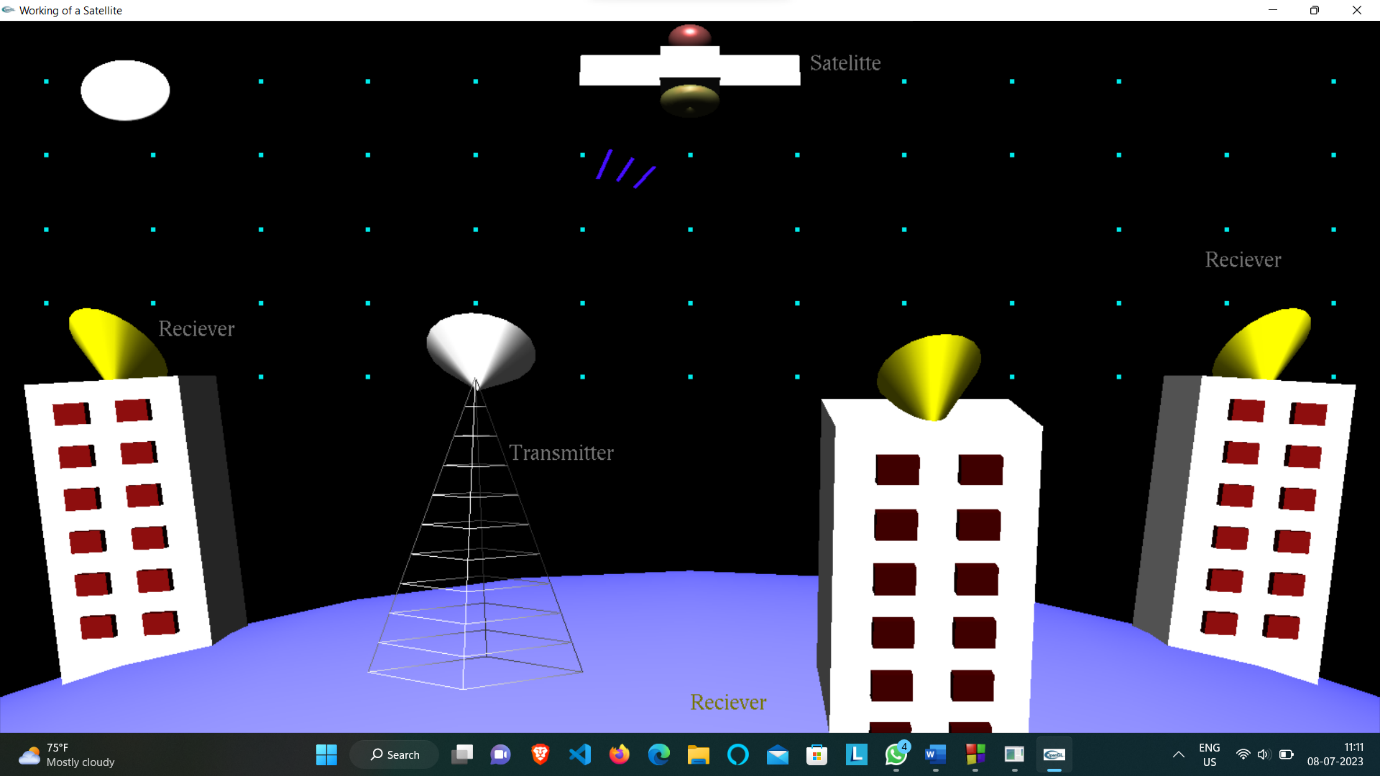
**Fig 6.2: when s or S is pressed**

****

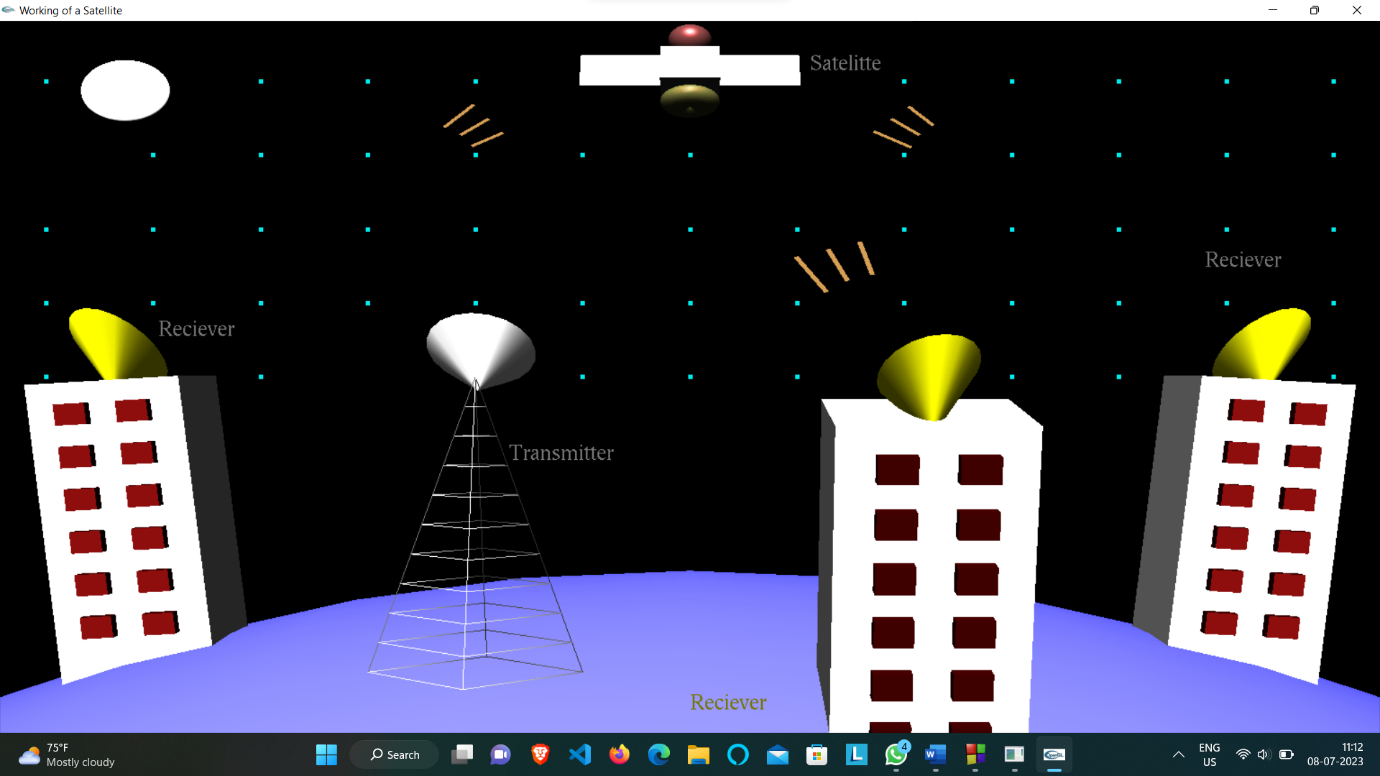
**Fig 6.3: satellite moving around earth**

****

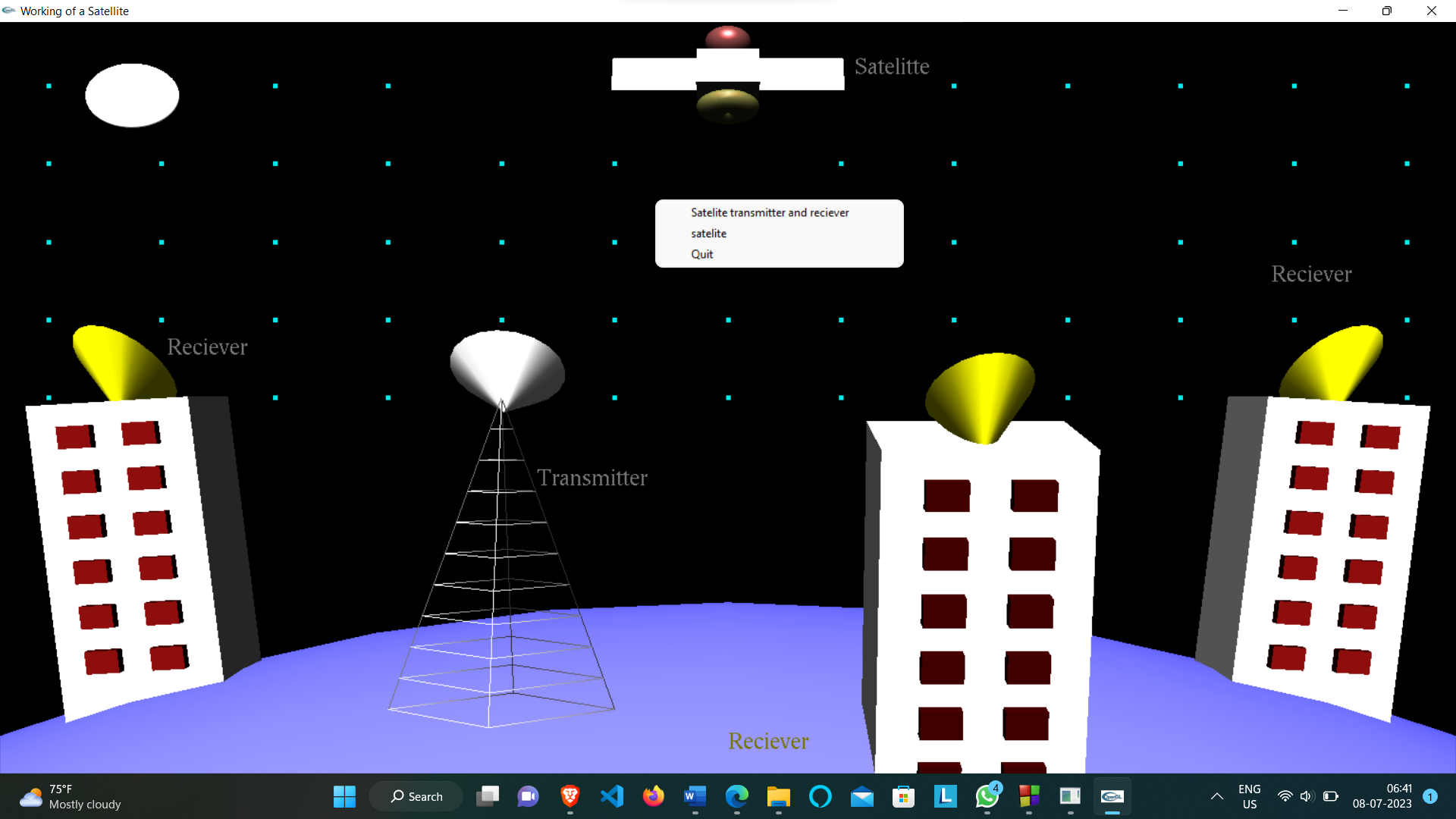
**Fig 6.4: city view**



**Fig 6.5: transmittion of signals from transmitter to satellite**



**Fig 6.6: transmittion of signals from satellite to receiver**

****

**Fig 6.7: options**

**CONCLUSION**

The project “Working of a Satellite” demonstrates how signals are transmitted and received to and from a satellite. Finally we conclude that this program clearly illustrate the working of a satellite using OpenGL and has been completed successfully and is ready to be demonstrated.

**FUTURE SCOPE**

These are the features that are planned to be supported in the future

Satellite communication refers to the transmission of signals, such as voice, data, or video, between ground-based stations and satellites in space. It involves the use of specialized equipment and technology to establish communication links over long distances.

On the other hand, OpenGL (Open Graphics Library) is a programming interface for rendering 2D and 3D graphics. It provides a set of functions that allow developers to create and manipulate graphical elements, such as polygons, textures, and shaders, for use in computer graphics applications.

While both satellite communication and OpenGL are important technologies in their respective fields, they do not have direct connections or dependencies on each other. Satellite communication focuses on enabling long-range communication, while OpenGL is primarily used for graphics rendering and visualization.

**REFERENCES**

**Text Books And Websites:**

1. Edward Angel: Interactive Computer Graphics a Top down Approach with Open GL 5th edition. Addison- Wesley, 2008.
2. James D Foley, Andries Van Dam, Steven K Feiner, John F Hughes, Computer Graphics.
3. [www.openGL.org](http://www.opengl.org)
4. [www.google.com](http://www.google.com)
5. <http://jerome.jouvie.free.fr/OpenGl/Lessons/Lesson3.php>
6. <http://opengl.org>