**CHAPTER 1**

**INTRODUCTION**

**1.1 About Computer Graphics:**

The Computer Graphics is one of the most effective and commonly used methods to communicate the processed information to the user. It displays the information in the form of graphics objects such as pictures, charts, graphs and diagram instead of simple text.

In computer graphics, pictures or graphics objects are presented as a collection of discrete picture elements called pixels. The pixel is the smallest addressable screen element.

Computer graphics today is largely interactive. The user controls the content structure, and appearance of objects and their displayed images by using input devices, such as a keyboard, mouse, or touch-sensitive panel on the screen.

Computer Graphics relies on an internal model of the scene, that is, mathematical representation suitable for graphical computations.

The model describes the 3D shapes, layout and materials of the scene. This 3D representation then has to be projected to compute a 2D image from a given viewpoint, this is rendering step.

Rendering involves projecting the objects, handling visibility (which parts of objects are hidden) and computing their appearance and lighting interactions. Finally, for animated sequence, the motion of objects has to be specified.

Computer graphics can be separated into two different categories as raster graphics and vector graphics. While both in essence set out to achieve the same goal, they use different techniques and therefore have different strengths and weaknesses.

**1.2 About OpenGL:**

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. 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. OpenGL's basic operation is to accept primitives such as points, lines and polygons, and convert them into pixels. This is done by a graphics pipeline known as the OpenGL state machine.

**Features of OpenGL:**

* Geometric Primitives allow you to construct mathematical descriptions of objects.
* Viewing and Modeling permits arranging objects in a 3D scene, move our camera around space and select desired vantage point for viewing the scene to be rendered.
* Materials lighting OpenGL provides commands to compute the color of any point given the properties of the material and the sources of light in the room.
* Transformations: rotation, scaling, translations, perspectives in 3D, etc.

**1.3 About the Project:**

We have created a 3d model of a building YONGSAN INTERNATIONAL BUSINESS, SEOUL SOUTH KOREA. We analyzed the structure of the building and tried to build it using opengl functions. Our project has a walkthrough of the building where you can move the camera around the building

**CHAPTER 2**

**REQUIREMENT SPECIFICATION**

The requirement specification is a comprehensive description of the software and the hardware requirements required to run the project successfully.

**2.1 Software Requirements:**

* Operating System: Windows 11
* Language Tools: OpenGL
* Compiler: GNU GCC compiler / C++ compiler
* Documentation Tool: Code Blocks with OpenGL functions
  1. **Hardware Requirements:**

* Processor: AMD Ryzen 5 4000HS / Intel Core 9th Gen i5 processor or higher variant
* RAM: 8GB or more
* ROM: 256GB SSD or more
* Input: Keyboard and Mouse
* Display: 1920 X 1080 resolution display or more
* Memory: 4GB

**Chapter 3**

**DESIGN**

The design specifies the design of various aspects and different stages of the project. Design is the ultimate frame work of the project. In essence the design is a plan or a mindful blueprint of the project to be developed.

The flow of the operation of the output window is shown below:

The Twin Tower

User Interaction

Keyboard

Press Arrow keys

To change camera angle

Press ‘T’ button

To look from Top view

Figure 3.1: Design of the project

**CHAPTER 4**

**IMPLEMENTATION**

Implementation is the stage where all planned activities are put into action. Before the implementation of a project, the implementors (spearheaded by the project committee or executive) should identify their strength and weaknesses (internal forces), opportunities and threats (external forces).

Implementation also includes a pseudo code.

**4.1 Pseudo code:**

void display()

{

Clears the window (and the depth buffer) and draws the moon as viewed from the current position of the orbiter.

}

void reshape()

{

reshape() fixes up the projection matrix so that we always see a sphere instead of an ellipsoid.

}

void init()

{

Enables depth testing, enables lighting for a bright yellowish diffuse light, and creates a moon.

}

int main()

{

The application code that are essential to create window to display graphics.

}

**4.2 Flowchart:**

main()

front()

display()

Twin Tower ()

Front view

Rear view

Top view

Go near to object

Go far away from object

Switch lighting between day and night

Fig 4.2.1: Flowchart of the project

**4.3 Function used:**

**4.3.1 User defined functions used in the project:**

* void init()

{

This function is used to initialize the viewing of the output.

}

* void reshape()

{

This function is used to define what to do when the window is resized

}

* void timer()

{

This function is used to register a timer callback to be triggered in a specified number of milliseconds.

}

* void display()

{

This function sets background color to black and opaque

This function also clears the color buffer (Background)

}

* void advance()

{

This function is used to provide with a tool called uniform buffer objects that allows to declare a set of global uniform variables that remain the same.

}

* void draw()

{

This function is used to provide the user to draw an object.

}

**4.3.2 Standard library functions used in the project:**

* **glutInit()**

glutInit is used to initialize the GLUT library

* **glutInitDisplayMode()**

glutInitDisplayMode sets the initial display mode.

* **glutInitWindowPosition()**

glutInitWindowPosition is used to set initial window position.

* **glutInitWindowSize()**

glutInitWindowSize is used to set initial window size.

* **glutCreateWindow()**

glutCreateWindow is used to provide the name to the window system as a window’s name.

* **glutDisplayFunc()**

glutDisplayFunc sets the display callback for the current window.

* **glutTimerFunc()**

glutTimerFunc registers a timer callback to be triggered in a specific number of milliseconds.

* **glutReshapeFunc()**

glutReshapeFunc sets the reshape callback for tht current window.

* **glutMainLoop()**

glutMainLoop enters the event processing loop.

* **glutPostRedisplay()**

glutPostRedisplay is used to request the display callback be executed after the callback returns.

* **glutSwapBuffers()**

glutSwapBuffer is to swap buffers of current window if double buffered

* **glutSolidSphere()**

glutSwapBuffer renders a solid or wireframe sphere.

* **glEnable()**

glEnable specifies a symbolic constant indicating capabilities.

* **glLightfv()**

**glLightfv** function returns light source parameter values.

* **glViewport()**

glViewport specifies the affine transformation of x and y from normalized device coordinates to window coordinates.

* **glMatrixMode()**

glMatrixMode is used to specify which matrix is the current matrix

* **glLoadIdentity()**

glLoadIdentity replaces the current matrix with the identity matrix.

* **glClear()**

glClearfunction clears buffers to preset values.

* **glPushMatrix()**

glPushMatrix pushes matrixstack down by one, duplicating current matrix.

* **glPopMatrix()**

glPopMatrix pops the current matrix stack, replacing the current matrix with the one below it on the stack.

* **glEndList()**

glEndList displays the list of commands that have been stored for subsequent execution.

* **glViewport ()**

**glViewport** function sets the viewport.

* **gluPerspective():**

gluPerspective defines a perspective viewing volume using the y direction field of view fov measured in degrees ,the aspect ratio of the front clipping plane and the near and far distances.

**CHAPTER 5**

**SNAPSHOT**

The Twin Tower Project consists of a 3-dimensional view with 360 degree walkthrough. These can be further explained in the snapshots provided below,

i.e.,

|  |
| --- |
|  |
|  |
| Figure 5.1: Front view of The Twin Tower. |

|  |
| --- |
|  |
| Figure 5.2: Bottom view of The Twin Tower. |
|  |
|  |
| Figure 5.3: Top view of The Twin Tower. |
|  |

|  |
| --- |
|  |
| Figure 5.4: Right rotation of The Twin Tower. |
|  |
|  |

**CONCLUSION AND FUTURE ENHANCEMENT**

**Conclusion:**

We can conclude that “The Twin Towers” constitute a three-dimensional urban community of interlocking horizontal and vertical towers. The project ‘The Twin Tower’ has been successfully implemented using OpenGL features for creating a place of natural beauty and have a healthy environment. The illustration of graphical principles and OpenGL features are included and application program is efficiently developed. In the first view, to create a place of natural beauty and the we have shown the left view. In the second view, we have the top view of Twin Tower. This package is very useful for the user since it provides their basic information about various OpenGL functions and its component utilities. This is an interactive project which has user friendly interaction given through keyboard. Thus, this project meets the basic requirements successfully and is flexible in all respects to one and all

**Future Enhancements:**

* Including more detailed interior view.
* Include more objects around towers to make it more attractive.

**REFERENCES**

**Reference Books:**

1. Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version,3rd / 4th Edition, Pearson Education,2011
2. James D Foley, Andris Van Dam, Steven K Feigner, John F Hughes Computer graphics with OpenGL: Pearson education

**Website Links:**

1. https://www.youtube.com/
2. https://www.udemy.com/
3. https://www.edureka.co/
4. https://www.geeksforgeeks.org