**Thapar Institute of Engineering And Technology, Patiala**



COMPUTER GRAPHICS

MINI PROJECT

Named

**Space\_Shooterz**

By

Lakshay Bhatti 102017057

Tanveer Singh 102017066

To

**CERTIFICATE**

**Signature of the**

**Dr.**

CSE, TIET

**DECLARATION**

**ACKNOWLEDGEMENT**

**Lakshay Bhatti (102017057)**

**Tanveer Singh (102017066)**

6th Sem (3CS3)

**ABSTRACT**

The project is two player shooting game, where there will be two spaceship

representing two players and objective is to destroy opponent’s ship first.

This project implements the 2D model viewing and translation of spaceship

and various openGL functions to create 2D model of ship and various transformation to perform movement of ship and display it on the

screen. This project uses quadratic equation to solve shooting function.

Here we are using mouse for selecting option from menu and keyboard for moving the spaceship and shooting beams.

**TABLE OF CONTENTS**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Titles** | | | |  | | **Page No.** |
| **List of Figures** | | | |  | | **ⅠI** |
| **CHAPTER 1 INTRODUCTION** | | | |  | | **01** |
|  | 1.1 About Computer Graphics | | | | | 01 |
|  | 1.2 About OpenGL | | | | | 01 |
|  | | 1.2.1 OpenGL Architecture | | | | 01 |
|  | 1.3 OpenGL Library Organization | | | | | 02 |
|  | 1.4 OpenGL in the Project | | | | | 02 |
|  | 1.5 OpenGL Functions | | | | | 03 |
|  | | 1.5.1 GLUT Call-backs | | | | 03 |
|  | | 1.5.2 OpenGL Command | | | | 03 |
|  | | | |  | |  |
| **CHAPTER 2 SYSTEM REQUIREMENTS** | | | | |  | **04** |
|  | 2.1 Software Requirements | | | | | 04 |
|  | 2.2 Hardware Requirements | | | | | 04 |
|  | | | |  | |  |
| **CHAPTER 3 DESIGN** | | | |  | | **05** |
|  | | | |  | |  |
| **CHAPTER 4 IMPLEMENTATION** | | |  | | | **06** |
|  | 4.1 In-Built OpenGL Functions | | | | | 06 |
|  | 4.2 Standard Header and Library Functions | | | | | 10 |
|  | 4.3 User-Defined Functions | | | | | 11 |
|  | 4.4 Algorithm to Shoot at aTarget | | | | | 13 |
|  | | | |  | |  |
| **CHAPTER 5 RESULTS** | | | |  | | **14** |
| 5.1 Screenshots | | | |  | | **14** |
|  | | | |  | |  |
| **CONCLUSION** | | | |  | | **17** |
| **REFERENCES** | | | |  | | **18** |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Fig. No.** | **Description** | **Page No** |
| 5.1.2.1 | Random Screenshot from game | 14 |
| 5.1.2.2 | Spaceship Shooting | 15 |
| 5.1.2.3 | Instruction Menu | 15 |
| 5.1.3 | Displaying menu | 16 |
|  |  |  |
|  |  |  |

**CHAPTER 1**

**INTRODUCTION**

**1.1 About Computer Graphics**

Computer Graphics in today’s world is one of the most widely used powerful and interesting features of the computer which gives us the power to handle the graphical data very efficiently and also process them rapidly and effectively Computer graphics started with a display of data on hard copy plotters and cathode ray tube screens soon after the introduction of computers themselves. The development of computer graphics has been driven both by the needs of the user community and by advances in hardware and software. Computer graphics today is largely interactive. The user controls the contents, structures and appearances of the object and of their displayed images by using input devices, such as keyboard, mouse or touch-screen. Due to the close relationships between the input devices and the display, the handling of such devices is included in the study of computer graphics.

**1.2 About OpenGL**

OpenGL is a software interface to graphics hardware. OpenGL is designed to work efficiently even if the computer that displays the graphics you create isn’t the computer that runs your graphics program. OpenGL is designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms. OpenGL doesn’t provide high-level commands for describing models of three-dimensional objects.

**1.2.1 OpenGL Architecture**

This is a diagram representing the flow of graphical information, as it is processed from CPU to the frame buffer, in OpenGL. The upper pipeline is for geometric, vertex-based primitives. The lower pipeline is for pixel-based, image primitives.

Texturing combines the two types of primitives together. There are several APIs related to OpenGL:

* + - * AGL, WGL, GLX: OpenGL & windowing systems interfaces
      * GLU: 2D & 3D Geometry, tessellations, etc.
      * GLUT: portable windowing API
  1. **OpenGL Library Organization**

Support provided by a graphics API:

1. Primitive functions: points, line segments, polygons, text, curves, surfaces.
2. Attribute functions: colour, fill, type face.
3. Viewing functions: attributes of the synthetic camera.
4. Transformation functions: rotation, translation, scaling. Matrix computations.
5. Input functions: keyboard, pointing device.
6. System communication: window system, OS, other workstations, other users.

OpenGL Library Organization:

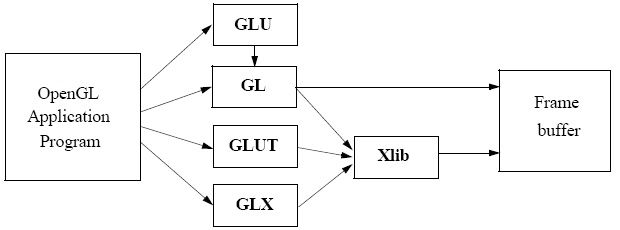


Fig 1.3.1: OpenGL Library Organization

**1.4 OpenGL in the Project**

In the project, we use a particular graphics software system, OpenGL. The applications are designed to access OpenGL directly through functions in three libraries. The first is the GL library in which the function name begins with gl. The second is the OpenGL utility library (GLU) which uses only GL functions but contains code for creating common objects and simplify viewing and these functions that begin with glu. To interface with the windows system and to get input from the external devices into our programs we use the third library, the OpenGL Utility Toolkit (GLUT) whose functions are prefixed as glut.

* 1. **OpenGL Functions**

**1.5.1 GLUT Callbacks**

OpenGL has a wide variety of functions in GL, GLU, GLUT, and GLE. GLUT uses a callback mechanism to do its event processing. Callbacks simplify event processing for the application developer. Given below are some of the functions used in this project:

glutDisplayFunc()- called when pixels in the window need to be refreshed.

glutMouseFunc()- called when the user presses a mouse button on the mouse. glutKeyboardFunc()- called when user enters a value (input) though Keyboard.

glutIdleFunc()- called when nothing else is going on.

glutReshapeFunc()- sets the reshape callback for the current window.

**1.5.2 OpenGL Commad**

The OpenGL API calls are designed to accept almost any basic data type, which is reflected in the calls name. Knowing how the calls are structured makes it easy to determine which call should be used for a particular data format and size. For instance, vertices from most commercial models are stored as three component floating point vectors.

As such, the appropriate OpenGL command to use is glVertex3fv (co-ordinates). For glVertex\*() calls Selective Repeat ARQ which don’t specify all the coordinates i.e., glvertex2f (), OpenGL will take default z=0.0

**CHAPTER 2**

**SYSTEM REQUIREMENT**

The package is designed such that users with a computer having minimum configuration can also use it, which does not require complex graphics packages.

The package requires simple in-built functions found in the header file along with a few users defined functions.

* 1. **Software Requirements**
     + Operating System – Windows 7(minimum)
     + Language Tool - C Compiler - GNU C Compiler
     + Libraries - freeglut
     + Documentation Tool - MS-Word
  2. **Hardware Requirements**
     + Processor - Intel Pentium onwards Compatible Hardware
     + RAM - 256Mb RAM (minimum)
     + Hard Disk - 3 GB (minimum)
     + Monitor – VGA Compatible
     + Keyboard - Standard 101 keys Keyboard

**CHAPTER 3**

**DESIGN**

The project “Star Wars” is designed in C++ language using some standard header and library functions. When we run the program, a window appears then you press “Enter” it will take you to a Menu.

Start game menu option.

* It will directly take you to the game.

Instruction menu option.

* It is a help window, which will show key configuration.
* And it will show game objective.

Quit menu option.

* This option quits the game and stop program execution

Keyboard control for PLAYER 1:

* W - UP
* S - DOWN
* A - LEFT
* D - RIGHT
* C - to shoot, Use 'w' and 's' to change direction.

Keyboard control for PLAYER 2:

* I - UP
* K - DOWN
* J - LEFT
* L - RIGHT
* M - to shoot, Use 'I' and 'K' to change direction.

**CHAPTER 4**

**IMPLEMENTATION**

**4.1 OpenGL In-built Function**

**4.1.1 Initialize**

This Function is used to initialize all argument variables.

Code: **glutInit(&argc, argv);**

**4.1.2 WindowPosition**

This Function is used to set window position.

Code: **glutInitWindowPosition(0, 0);**

**4.1.3 CreateWindow**

This Function is used to create a window.

Code: **glutCreateWindow("Star Wars");**

**4.1.4 DisplayFunc**

This Function is used to register display function.

Code: **glutDisplayFunc(display);**

**4.1.5 ReshapeFunc**

This Function is used to register reshape function.

Code: **glutReshapeFunc(myReshape);**

**4.1.6 KeyboardFunc**

This Function is used to register character keys interaction function.

Code: **glutKeyboardFunc(keys);**

**4.1.7 glutBitmapCharacter**

This Function is used to renders a bitmap character using OpenGL.

Code: **glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24 , \*c);**

**4.1.8 MouseFunc**

This Function is used to register mouse function.

Code: **glutMouseFunc(mouse);**

**4.1.9 LineWidth**

This Function is used to specify the width of rasterized lines.

Code: **glLineWidth(5);**

**4.1.10 Scale**

This Function is used to multiply the current matrix by a general scaling matrix.

Code: **glScalef(2, 2 ,0);**

**4.1.11 SwapBuffers**

This Function swaps the buffers of the current window if double buffered.

Code: **glutSwapBuffers();**

**4.1.12 Colour**

This Function is used to colour the displaying objects

Code: **glColor3f (0.7, 0.0, 0.0);**

**4.1.13 Vertex**

This function is used to draw a vertex in 3 dimensions.

Code: **glVertex3f(1,.8,-1);**

**4.1.14 PostRedisplay**

This function is used to marks the current window as needing to be redisplayed.

Code: **glutPostRedisplay();**

**4.1.15 Projection matrix**

This function is used for the projection of matrix.

Code: **glMatrixMode();**

**4.1.16 Reset Matrix**

This function is used to reset the matrix.

Code: **glLoadIdentity();**

**4.1.17 Viewport**

This function is used for the viewport

Code: **glViewport(0, 0, w, h);**

**4.1.18 Modelview**

This function is used for to get back to the modelview

Code: **glMatrixMode(GL\_MODELVIEW);**

**4.1.19 MainLoop**

This function call causes the program to enter an event processing loop. It should be the last statement in main.

Code: **void glutMainLoop();**

**4.1.20 ClearColor**

This function call sets the present RGBA clear color used when clearing the color buffer.

Code: **glClearColor(0.2, 0.2, 0.2, 1.0);**

**4.1.21 DisplayMode**

This function call requests a display with the properties that are specified in mode.

Code: **glutInitDisplayMode (GLUT\_SINGLE|GLUT\_RGB);**

**4.1.22 WindowSize**

This function call specifies the initial height and width of window in pixels.

Code: **glutInitWindowSize(1200, 600);**

**4.1.23 ClearColor**

This function call forces any buffered openGL command to execute.

Code: **glFlush();**

**4.2 Standard Header and Library Functions**

**4.2.1 stdio.h**

This header file is used for standard input and output and stream manipulation function.

**4.2.2 stdlib.h**

Stdlib.h is the header of the general-purpose standard library of C programming language which include functions involving memory allocation, process control, conversions and others.

**4.2.3 GL/gl.h**

Library utility of gl functions in OpenGL program.

**4.2.4 GL/glu.h**

Library utility of glu functions in OpenGL program.

**4.2.5 GL/glut.h**

Library utility of glut functions in OpenGL program.

**4.2.6 math.h**

math.h is the header of functions used to perform mathematical operations.

**4.3 User-Defined Functions**

**4.3.1 Alien**

These functions defined to draw alien.

* void DrawAlienBody(bool isPlayer1)
* void DrawAlienCollar()
* void DrawAlienFace(bool isPlayer1)
* void DrawAlienBeak()
* void DrawAlienEyes(isPlayer1)

**4.3.2 Alien Ship**

These functions defined to draw Alien ship.

* void DrawSpaceshipDoom()
* void DrawSteeringWheel()

**4.3.3 Laser**

These functions defined to draw laser beam.

* void DrawLaser(int x, int y, bool dir[])

**4.3.4 Menu**

These functions defined to create main menu.

* void introScreen()
* void startScreenDisplay()
* void instructionsScreenDisplay()
* void gameScreenDisplay()

**4.3.5 Checking Laser Contact**

This function defined to find out if the laser intersects with spaceship.

* void checkLaserContact(int x, int y, bool dir[], int xp, int yp, bool player1)

**4.3.6 Display**

This function defined to display the result which is invoked by built-in function glutDisplayFunc() in main.

* void display(void)

**4.3.7 Mouse**

This function defined to perform mouse button interaction which is invoked by built-in function glutMouseFunc() in main.

* void mouse(int btn, int state, int x, int y)

**4.3.8 Keyboard**

This function defined to perform character keys of keyboard interaction which is invoked by built-in function glutKeyboardFunc(keys) in main.

* void keys(unsigned char key, int x, int y)

**4.3.9 Key Operation**

This function defined to perform all the key operation during the movement and shooting of the spaceship.

* void keyOperations()

**4.3.10 Reshape**

This function defined to reshape result for changing screen size which is invoked by built-in function glutReshapeFunc(display) in main.

* void myReshape(int w, int h)

**4.3.11 PassiveMotionFunc**

This function sets passive motion callbacks respectively for the current window. The passive motion callback for a window is called when the mouse moves within the window while no mouse buttons are pressed.

* void passiveMotionFunc(int x,int y)

**4.4 Algorithm to Shoot at a Target**

Let the target be at the position A and moving with velocity VA, and the shooter be stationary at the position B and can fire bullets with speed s. Let the shooter fire at time 0. The bullet hits at time t such that |A − B + t VA| = t s. This is a straightforward quadratic equation in t, which you should be easily able to solve (or determine that there is no solution).

**CHAPTER 5**

**RESULTS**

**5.1 Screenshots**

**5.1.1 Execution**

To execute the project

* Open codeBlocks in Windows.
* Open project as new project (Glut Project).
* Click “Built and Run” button on top tab.

**5.1.2 Random Screenshot from game**

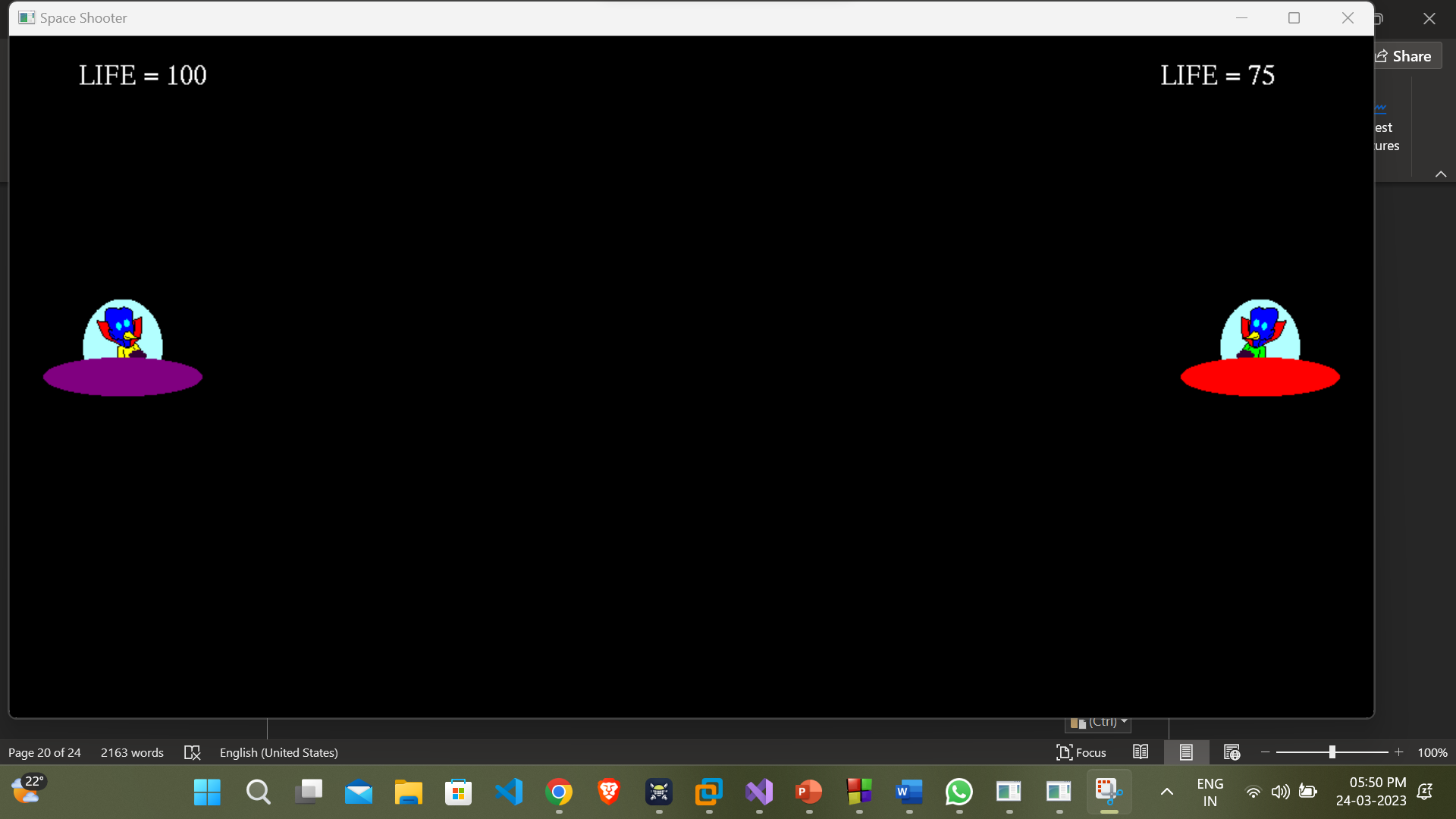
****

Figure 5.1.2.1 spaceship at initial position

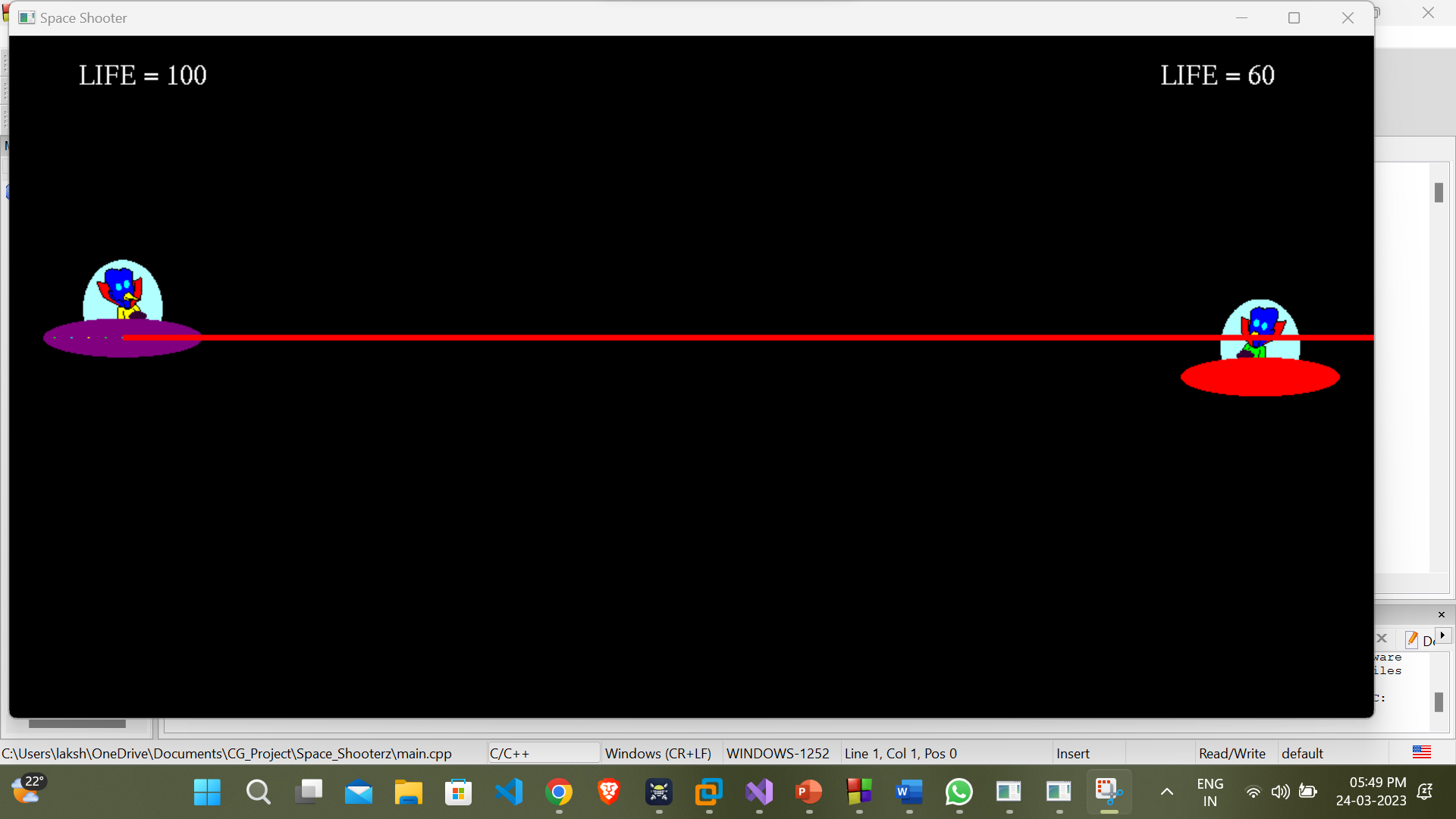
****

Figure 5.1.2.2 Spaceship Shooting

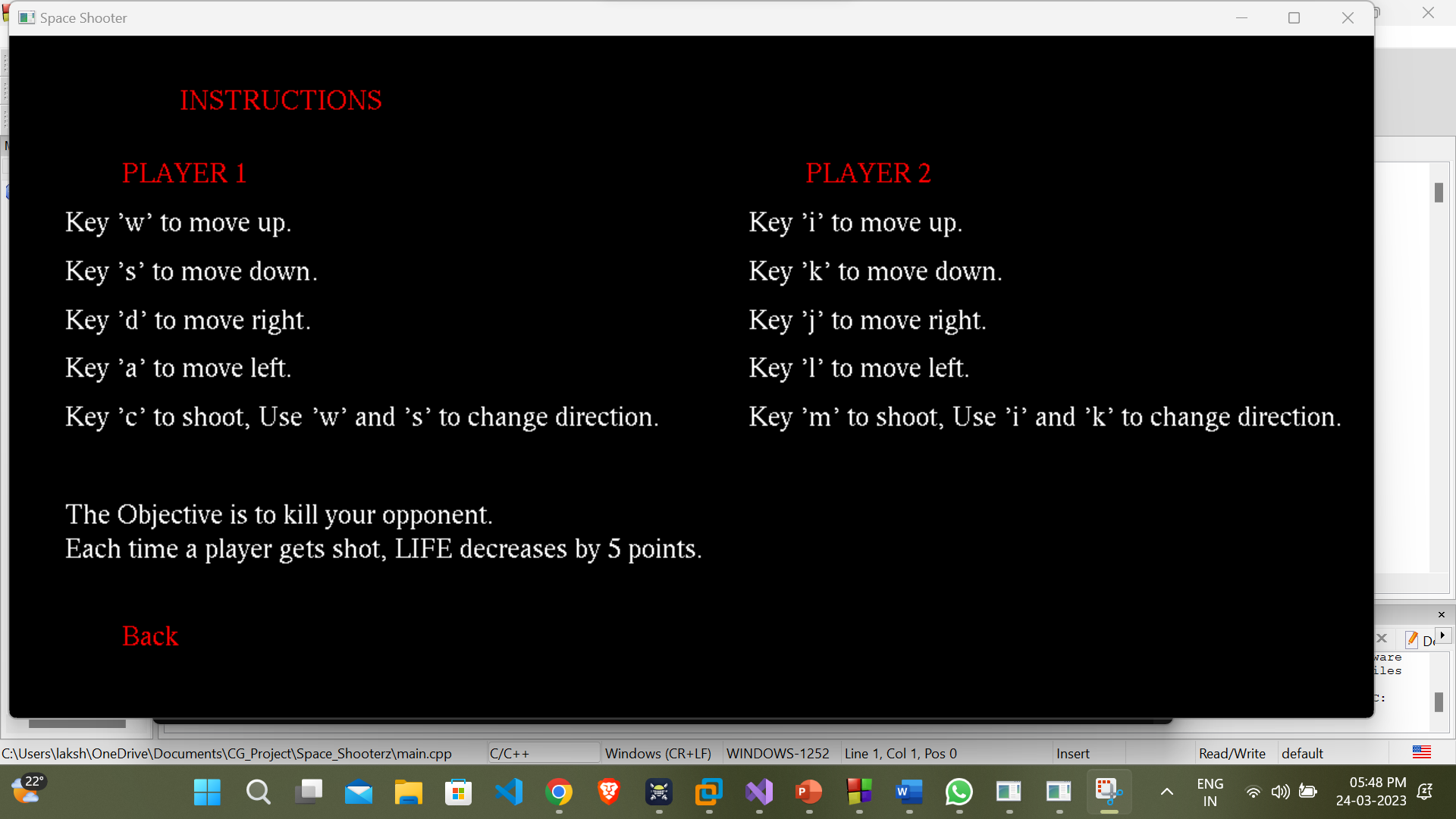
****

Figure 5.1.2.3 Instruction Menu

**5.2.2 Menu**

****

**Figure 5.1.3** **Displaying menu**

**CONCLUSION**

The wonderful experience we have gained developing this project. By working on the project, we gained hands- on experience of using the OpenGL software which made us feel the boundless applications it can bring. We have made use of mouse and keyboard driven interface, thus to reduce the complexity and make it user friendly.

The project helped in understanding the working of computer graphics using OpenGL and various concepts, functions and methodologies for the development of a graphics packages. The proposed project will serve its purpose without any hassles.

**REFERENCES**

1. Edward Angel, Interactive Computer Graphics a Top-Down Approach UsingOpenGL, 5th edition, Dorling Kindersley India Pvt. Ltd., UP, India, 2009.
2. Computer Graphics Using OpenGL – F.S. Hill, Jr. 2nd Edition, Pearson 1. Education,2001.
3. Computer Graphics – James D Foley, Andres Van Dam, Steven K Feigner, John F Hughes, Addison-Wesley 1997.
4. Computer Graphics - OpenGL Version – Donald Hearn and Pauline Baker, 2nd Edition, Pearson Education.
5. [www.geeksforgeeks.org](http://www.geeksforgeeks.org)
6. [www.stackoverflow.com](http://www.stackoverflow.com)