****

**AMERICAN INTERNATIONAL UNIVERSITY–BANGLADESH (AIUB)**

**Dept. of Computer Science**

**Faculty of Science and Technology**

**CSC4118: COMPUTER GRAPHICS**

**Spring: 2024-2025**

**Section: [L]**

**Group No: 09**

**Project Report On**

**Project Name [AIRPLANE CRASHER]**

**Submitted By:**

|  |  |  |
| --- | --- | --- |
| Sl# | **Name** | **ID** |
| 1 | **NAIYER NUR FAIROZ** | **23-50227-1** |
| 2 | **MD RAKIBUL ISLAM AYON** | **23-50236-1** |

**Submitted to: NOBORANJAN DEY**

# Introduction

We implemented this project using **C++ with OpenGL and GLUT**, which are popular libraries for creating 2D and 3D graphics. The core idea was to design a simple interactive game where a player controls a fighter jet. We created different drawing functions for the jet, enemy planes, clouds, sun, and fuel cans. A timer-based update function controls movement, scoring, and animations, while keyboard input allows the player to move the jet. By combining these components, we built a real-time interactive environment where gameplay feels smooth and engaging.

We implemented this project to practice and demonstrate the fundamentals of **computer graphics, game design, and user interaction**. By working on this game, we applied theoretical knowledge of transformations, translations, scaling and animation into a hands-on project. It also gave us experience with structuring a program that uses both continuous updates (for movement and collision) and event-driven programming (for user controls).

It demonstrates how simple geometric shapes and transformations can represent complex objects like airplanes and clouds. The scoring, lives system, and power-ups highlight how logical rules can be integrated with visual elements to enhance gameplay.

The target audience for this project includes **students, beginners in computer graphics, and anyone interested in learning game development**. Casual players may enjoy it as a simple arcade-style game.

**Technologies used:**  
 C++  
 OpenGL (GL, GLUT)  
 Windows Platform (Win32)

# List of Functions /Tools:

| **Tool / Function** | **Usage in Project** |
| --- | --- |

* **glBegin(mode) / glEnd()** – Defines the start and end of drawing a primitive shape (like GL\_QUADS, GL\_TRIANGLES, GL\_POLYGON).
* **glVertex2f(x, y)** – Specifies 2D vertex coordinates for shapes.
* **glColor3f(r, g, b)** – Sets the current drawing color (RGB values between 0–1).
* **glRasterPos2f(x, y)** – Sets the position where bitmap text will appear.
* **glutBitmapCharacter(font, char)** – Renders individual characters of on-screen text (for score, lives, Game Over).
* **glLoadIdentity()** – Resets the current transformation matrix (used before applying new transformations).
* **glTranslatef(x, y, z)** – Translates objects to a new position (used for moving clouds, planes, fuel).
* **glRotatef(angle, x, y, z)** – Rotates objects around an axis.
* **glScalef(x, y, z)** – Scales objects bigger or smaller.
* **glPushMatrix()** – Saves the current transformation matrix.
* **glPopMatrix()** – Restores the most recently saved transformation matrix (keeps transformations local to objects).
* **glClearColor(r, g, b, a)** – Sets the background (clear) color.
* **glClear(GL\_COLOR\_BUFFER\_BIT)** – Clears the screen before drawing.
* **glFlush()** – Forces execution of OpenGL commands (used in single-buffer mode).
* **glutSwapBuffers()** – Swaps front and back buffers to display the drawn frame (prevents flickering in double-buffer mode).
* **glutPostRedisplay()** – Marks the window for redrawing, ensuring continuous updates.
* **glutSpecialFunc(func)** – Registers callback for special keys (arrow keys in this case).
* **glutKeyboardFunc(func)** – Registers callback for normal key inputs.
* **glutDisplayFunc(func)** – Registers the display function for rendering.
* **glutTimerFunc(ms, func, value)** – Calls a function (like update()) after a specified number of milliseconds (used for the game loop).
* **glutInit(&argc, argv)** – Initializes the GLUT library.
* **glutInitDisplayMode(mode)** – Sets display mode (like GLUT\_DOUBLE for double buffering, GLUT\_RGB for color).
* **glutInitWindowSize(width, height)** – Defines the window size.
* **glutInitWindowPosition(x, y)** – Sets the initial position of the window on screen.
* **glutCreateWindow(title)** – Creates the window with a given title.
* **glutMainLoop()** – Starts the infinite loop that keeps processing rendering and input events.

**KNOWLEDGE APPLIED:**

1. **Jobmarket purpose:** The development of this project reflects the application of fundamental and advanced concepts in **computer graphics and software engineering.** With OpenGL and GLUT, it demonstrates proficiency in **real-time rendering, event-driven programming, and graphical user interface design,** which are essential skills in the professional domains **of game development, simulation software, and interactive visualization systems**. These abilities are directly transferable to industry roles in **software engineering, graphics programming** making the project highly relevant to current job market demands.

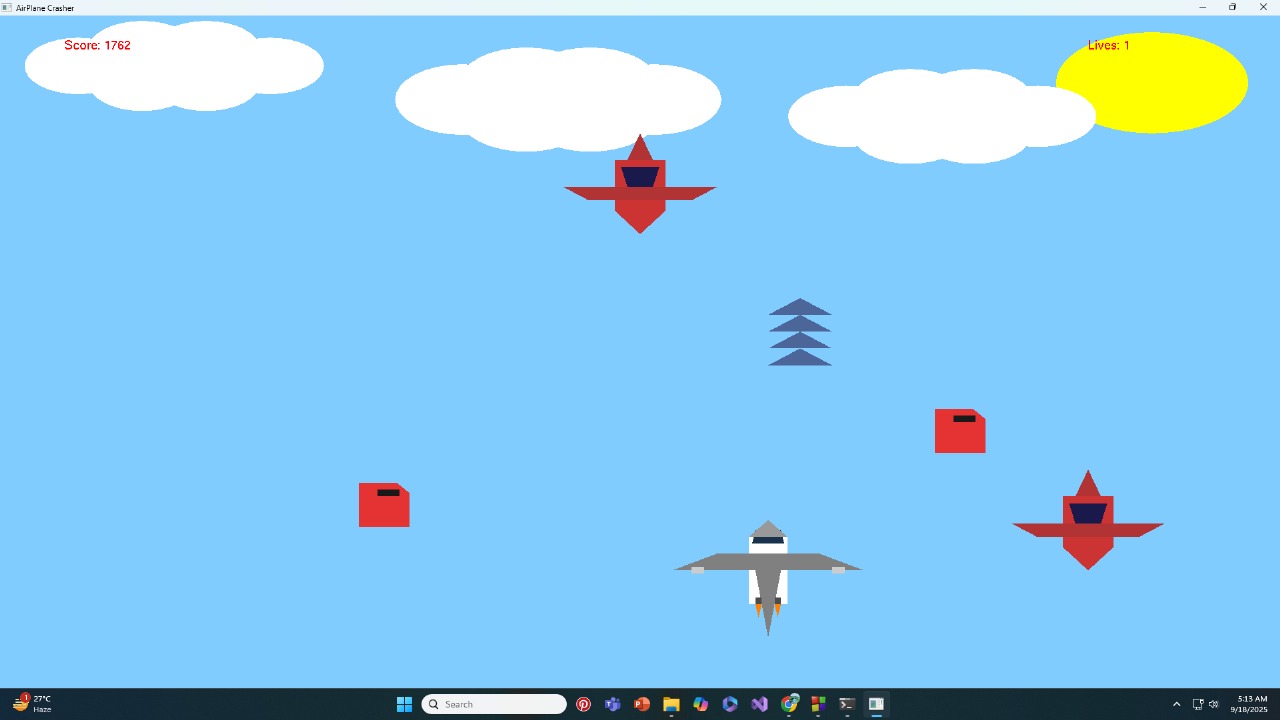
2. **Higher study purpose:** The use of geometric transformations, rendering techniques, and dynamic state management highlights a clear understanding of core concepts that form the foundation of higher-level study. This project is particularly significant for advanced research or postgraduate studies in areas such as **2D graphics rendering, GPU programming, virtual reality and simulation modeling.**

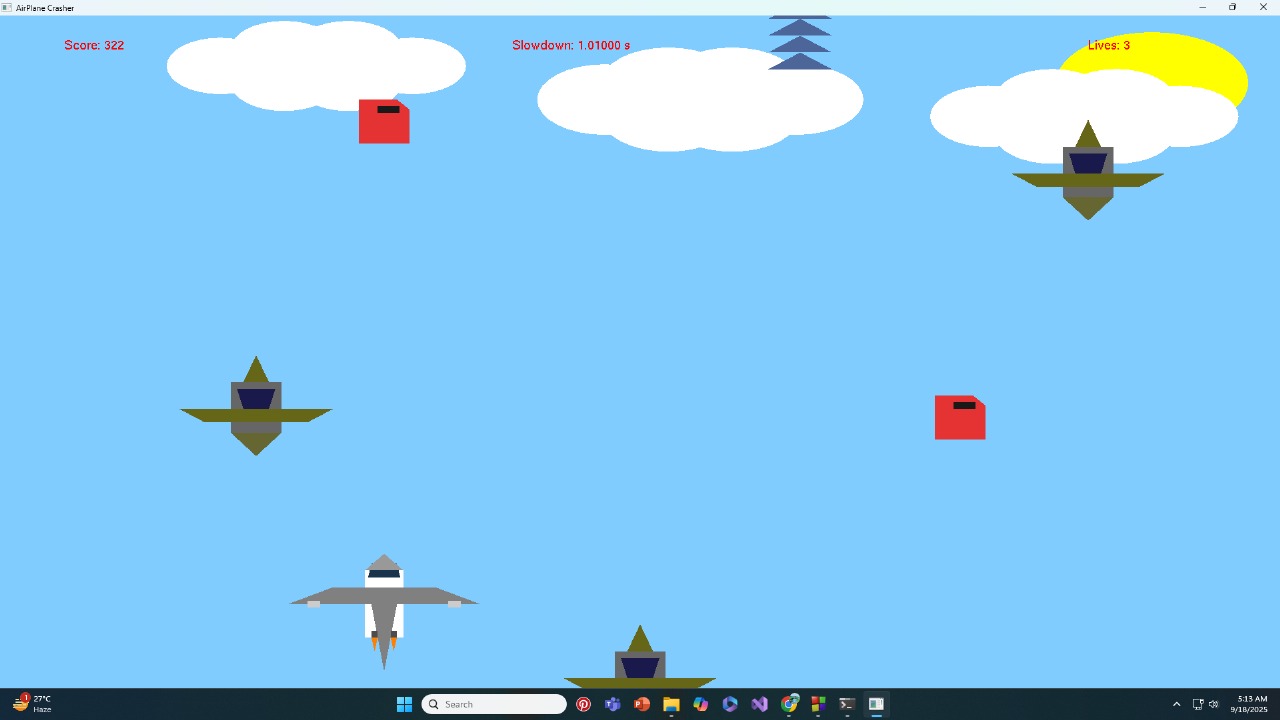
**Reference Material:**

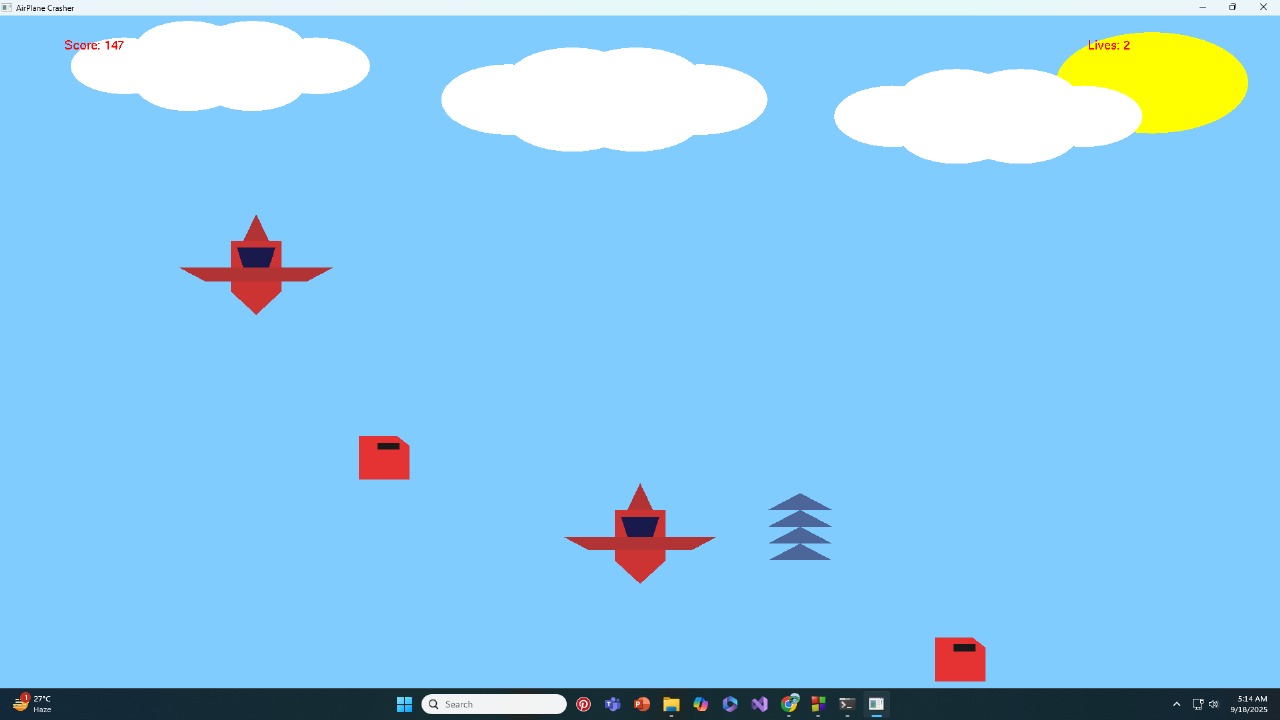
**1.Github link:** <https://github.com/Ayon-Islam1216/AirPlane-Crasher.git>

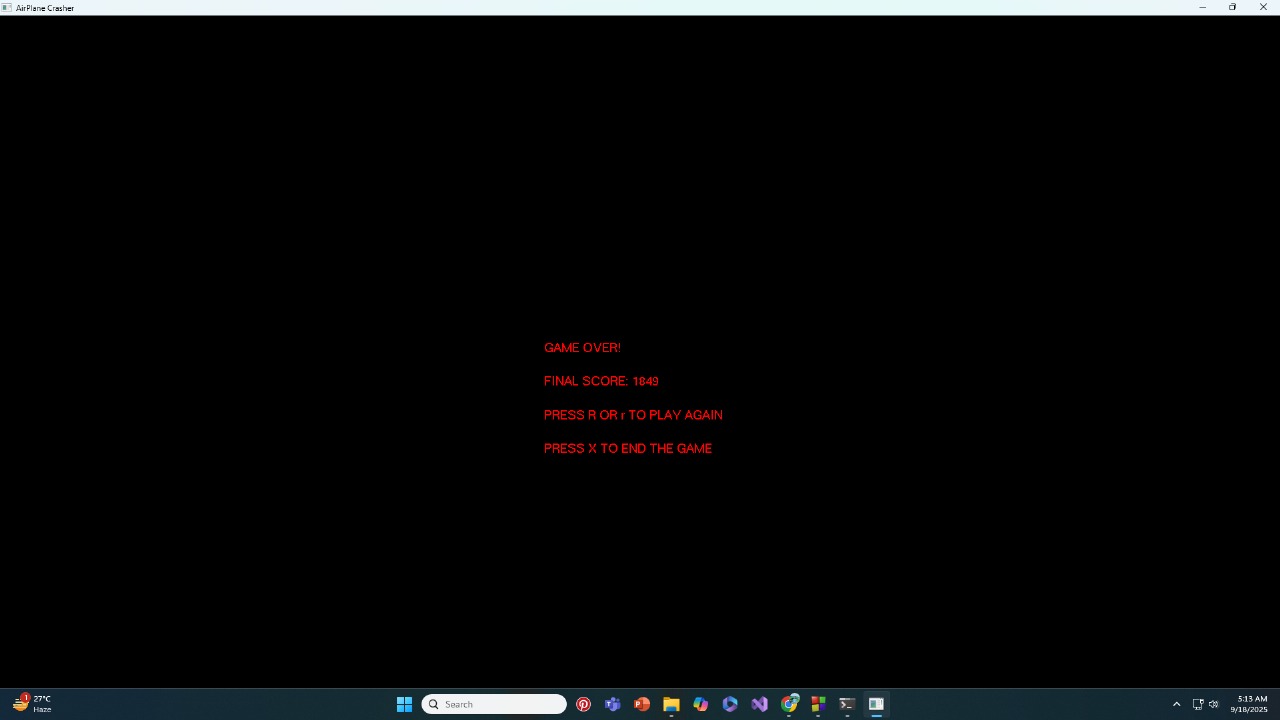
**2.Graph:** (attached)

**Screenshot:**

****

****

****

****