Project Report

Developing A 2D Asteroid Avoiding Game using OpenGL

Course Code: CSE2114

Course Title: Software Development Project-I and Industrial Tour

Submitted By	Supervised By
Jahid Hasan ID: CE22014 Nafis Ahmed ID: CE22015 Session: 2021-22	Dr. Mostofa Kamal Nasir Professor, Dept. of CSE, MBSTU

Department of Computer Science and Engineering,

Mawlana Bhashani Science and Technology University

Santosh, Tangail-1902

Table of Contents

1. Introduction **3**
1.1 Technical Aspects and Background **3**
1.2 Objective of the Project **3**
1.3 Scope of the Project **4**
1.4 Benefit or Significance of the Project **4**
2. Requirement Analysis **4**
2.1 Feasibility Study **5**
2.2 System Requirements **6**
2.3 User Requirements **6**
3. System Design **7**
3.1 Architecture of the Project **7**
3.2 Workflow of the Project **8**
4. Implementation and Testing **9**
4.1 Implementation of the Project **9**
4.2 Result[screenshot] **13**
4.3 Testing **13**
5. Conclusion **14**
6. Codebase and Resources on GitHub **14**

List of Figures:

Figure 1: Block Diagram of Project.....7

Figure 2**:** Workflow ofProject.....8

Figure 3**:** Home Window 10

Figure 4: Score Increasing…….10

Figure 5: Level increasing……...11

Figure 6:Collision 11

Figure 7:Game Over 12

Figure 8:Restart the Game 12

Figure 9:High Score Updating.....13

1. Introduction

1.1 Technical Aspects and Background

The Rocket and Asteroid is a software application developed in C, a powerful and versatile programming language widely used for system programming, embedded systems, and more. C was chosen for its efficiency, control over system resources, and the availability of robust libraries that streamline development and OpenGL for providing graphics library.

The 2D asteroid avoidance game is a beginner-friendly project designed to demonstrate fundamental concepts in computer graphics and game development using OpenGL and GLUT. It provides a hands-on way to understand the basics of rendering, collision detection, and interactive programming.

1.2 Objective of the Project

Player Interaction: Allow users to control a rocket and enable intuitive controls for movement, pausing, and restarting the game.

 Dynamic Gameplay:

- Implement progressive difficulty where asteroid speed and complexity increase as the game progresses.
- Offer a scoring system to track performance and encourage competition.

 Collision Detection and Response:

- Incorporate accurate collision detection to determine game-over scenarios when the rocket collides with asteroids.

 Visual Appeal:

- Create visually pleasing graphics and animations using OpenGL to enhance user engagement.
- Design realistic-looking asteroids and rockets with smooth movements.

 High Score Tracking:

- Implement a system to save and display high scores, encouraging players to improve their performance.

 Educational Objective:

- Demonstrate key concepts of computer graphics, including 2D rendering, real-time updates, and animation loops.

The primary objective of this project is to design and implement a 2D arcade-style game using OpenGL that demonstrates real-time rendering, interactive gameplay mechanics, and basic file handling for persistent data storage. The game focuses on simplicity, fun, and challenging gameplay while highlighting key concepts of computer graphics and programming.

1.3 Scope of the Project

 Gameplay Mechanics:

- Provide players with a straightforward objective to navigate a rocket while avoiding falling asteroids.
- Offer intuitive controls to move the rocket left or right and features such as pausing and restarting the game.

 Dynamic Features:

- Implement progressively challenging levels, where the speed and frequency of asteroids increase as the player's score rises.
- Integrate real-time collision detection and response for accurate gameplay outcomes.

 Visual and Graphical Components:

- Develop visually appealing graphics, including a realistic representation of asteroids, rockets, and backgrounds.

 Educational Purpose:

- Serve as a practical demonstration of computer graphics principles, such as 2D rendering, transformations, and event handling.
- Act as a foundational project for understanding game loops, input handling, and modular programming techniques.

 Scalability and Future Enhancements:

- The game can be expanded to include additional features such as power-ups, new levels, or multiplayer functionality.
- Provides a base for integrating advanced concepts like physics-based movements, AI-controlled elements, or 3D graphics.

1.4 Benefit or Significance of the Project

1. Educational Significance

- **Understanding Graphics Programming**:
  This project provides hands-on experience with OpenGL, helping to understand the fundamentals of 2D rendering, transformations, and animations.
- **Real-Time System Development**:
  It offers insights into creating real-time systems with continuous updates, collision detection, and interactive controls.
- **Modular Programming Skills**:
  The project promotes writing modular and reusable code, essential for developing large-scale applications.

2. Entertainment Value

- **Engaging Gameplay**:
  The game provides an engaging and challenging experience, motivating players to improve their skills and achieve higher scores.
- **User-Friendly Interface**:
  Intuitive controls and a simple interface make the game accessible to players of all ages.

3. Practical Application

- **Portfolio Project**:
  As a completed project, it can be included in the developer's portfolio to showcase programming and game development skills.
- **Foundation for Advanced Games**:
  The project lays the groundwork for exploring advanced game features, such as physics-based mechanics, AI, and 3D environments.

4. Scalability and Customization

- **Expandable Features**:
  The project can be enhanced with additional features like power-ups, new levels, multiplayer options, or aesthetic improvements.
- **Learning Platform**:
  It serves as a base for experimenting with more complex game development concepts and tools.

5. Significance in Academia

- **Demonstration of Skills**:
  The project allows students to demonstrate their understanding of computer graphics, programming concepts, and game logic.
- **Team Collaboration**:
  If developed collaboratively, it teaches the importance of teamwork, code management, and project planning.

2. Requirement Analysis

2.1 Feasibility Study

 Technical Feasibility:

- Uses free and widely supported tools (OpenGL, GLUT).
- Requires basic programming and graphics skills to be available to most developers.

 Economic Feasibility:

- Minimal cost since all tools are Open - source.
- No licensing fees or additional hardware requirements.

 Operational Feasibility:

- User-friendly interface and simple controls.
- Easily scalable with potential for adding new features and levels.

2.2 System Requirements

 Hardware:

- **Processor**: 1 GHz or faster processor.
- **RAM**: At least 2 GB.
- **Graphics**: OpenGL-compatible GPU.
- **Storage**: 10 MB for the game files and high-score persistence.

 Operating System:

- Windows or any OS capable of running GLUT (OpenGL Utility Toolkit).

 Software Dependencies:

- OpenGL library for rendering.
- GLUT library for handling input and display.

 File System:

- A file named highscore.txt in the game directory to store and load the high score.

 Development Tools (for modifications):

- GCC or any C++ compiler supporting OpenGL.
- An IDE or text editor for C++ development (e.g., Visual Studio, Code::Blocks).

 Runtime Requirements:

- A screen resolution of at least 800x600.
- Keyboard input for controlling the rocket and pausing/resuming the game.

2.3 User Requirements

- **Basic Computer Literacy:** Understanding fundamental computer operations, file management, and navigating through a command-line interface.
- **Understanding of Input Mechanisms:** Knowledge of how to input data and respond to system prompts using the keyboard and other input devices.

3. System Design

3.1 Architecture of the Project

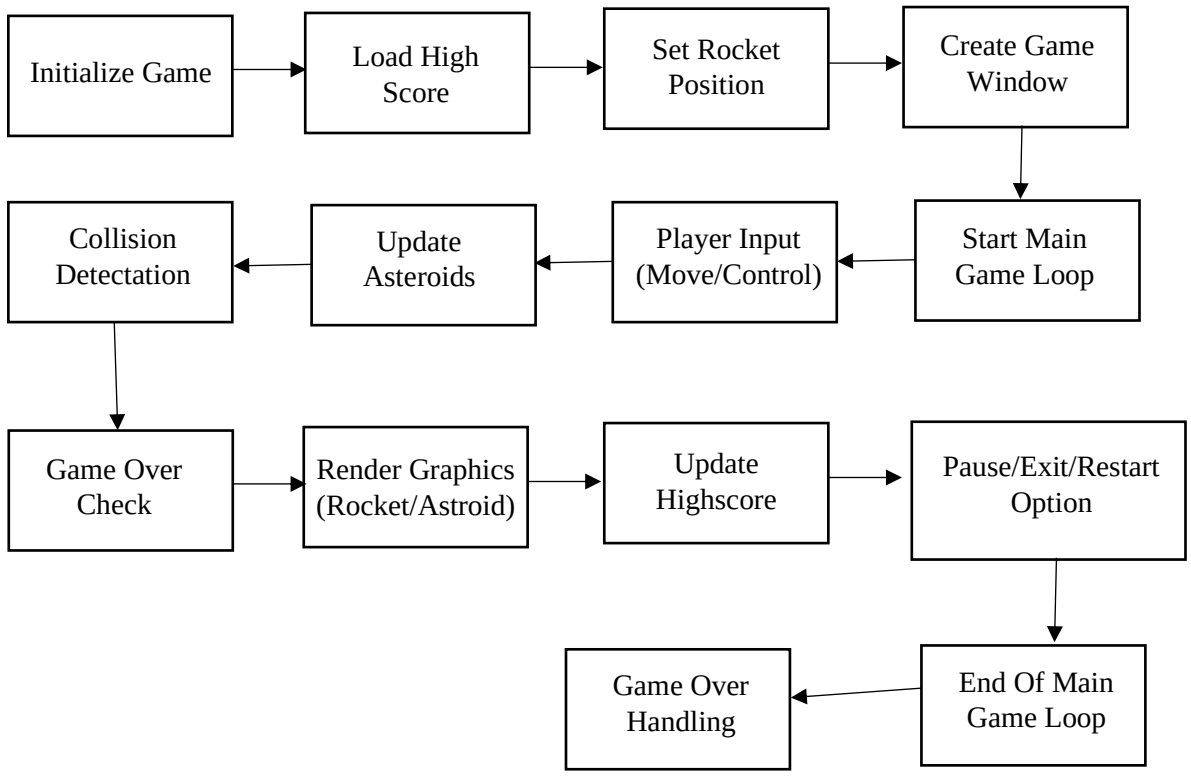

Figure 1: Block Diagram of project

3.2 Workflow of the Project

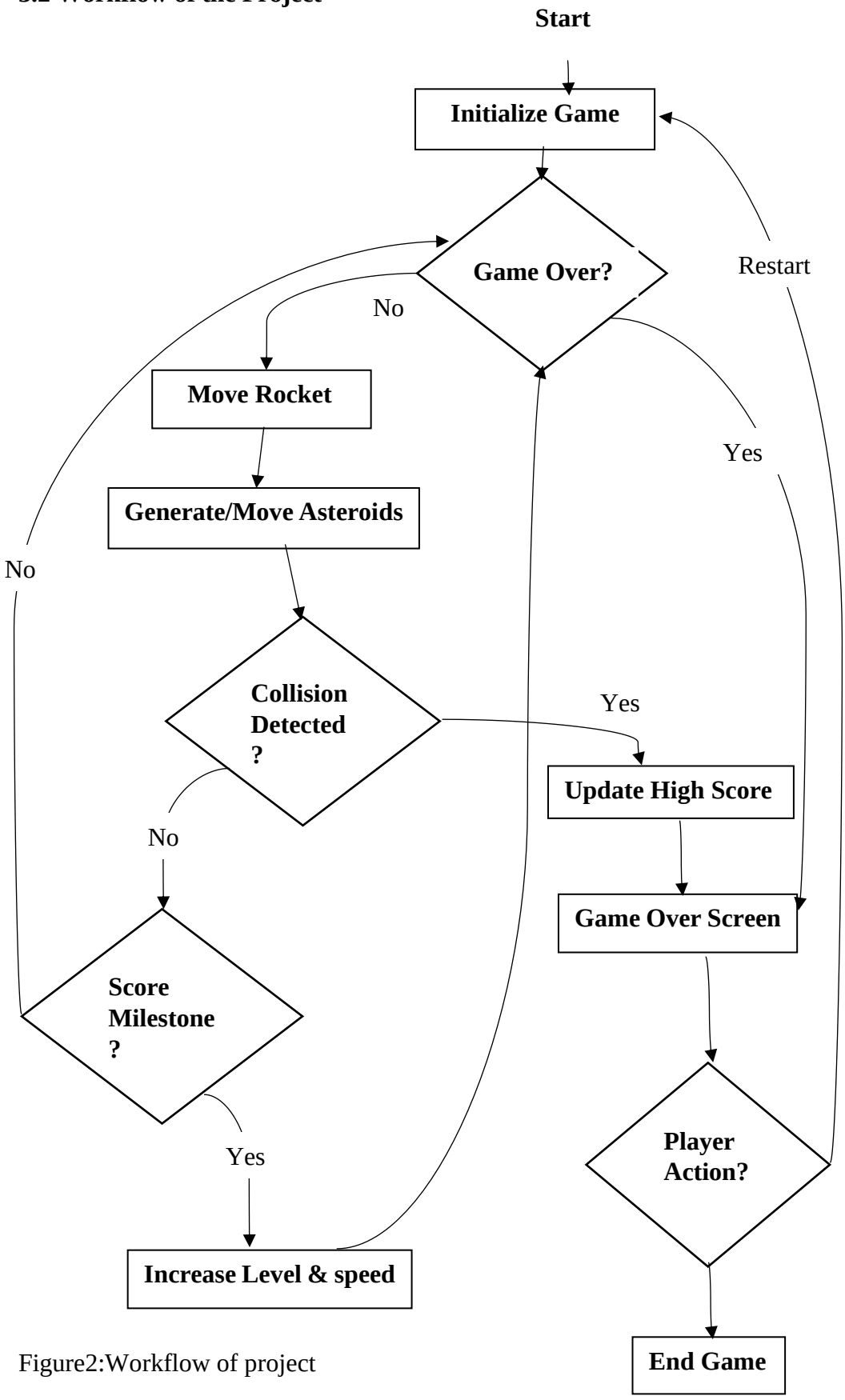

Figure2:Workflow of project

4. Implementation and Testing

4.1 Implementation of the Project

Developing the project is implemented in C, a robust programming language renowned for its low-level memory manipulation capabilities and efficient system interaction. The implementation involves several key aspects:

- **OpenGL**: This is used for rendering 2D graphics such as the rocket, asteroids, background, and game UI. OpenGL provides the functions to draw shapes (like rectangles and circles) and manage the display on the screen.
- **GLUT (OpenGL Utility Toolkit)**: This library is used to handle window management, user input (keyboard/mouse events), and managing the game's main loop. It also handles the creation of windows and context for rendering.

- **Standard C File Handling**: Used for saving and loading the high score to/from a text file (highscore.txt), so players can keep track of their highest score across sessions.

- **Collision Detection**: Implemented manually using simple geometric checks, like bounding box or rectangle collision detection, to determine if the rocket collides with any falling asteroid.
- **Game Loop**: A timer-based loop that updates game objects like the rocket and asteroids, checks for collisions, and manages the score, levels, and speed.

- **IDE/Compiler**: You can use an Integrated Development Environment (IDE) like Code::Blocks, Visual Studio, or Dev-C++ for writing and compiling the C/C++ code.
- **GCC (GNU Compiler Collection)**: If using Linux or other platforms, the GCC compiler can be used to compile the C/C++ source code.

- **Windows/Linux**: The project can be implemented on both Windows and Linux, as OpenGL and GLUT are cross-platform. However, setup might differ slightly between the two.

4.2 Result [Screenshots]

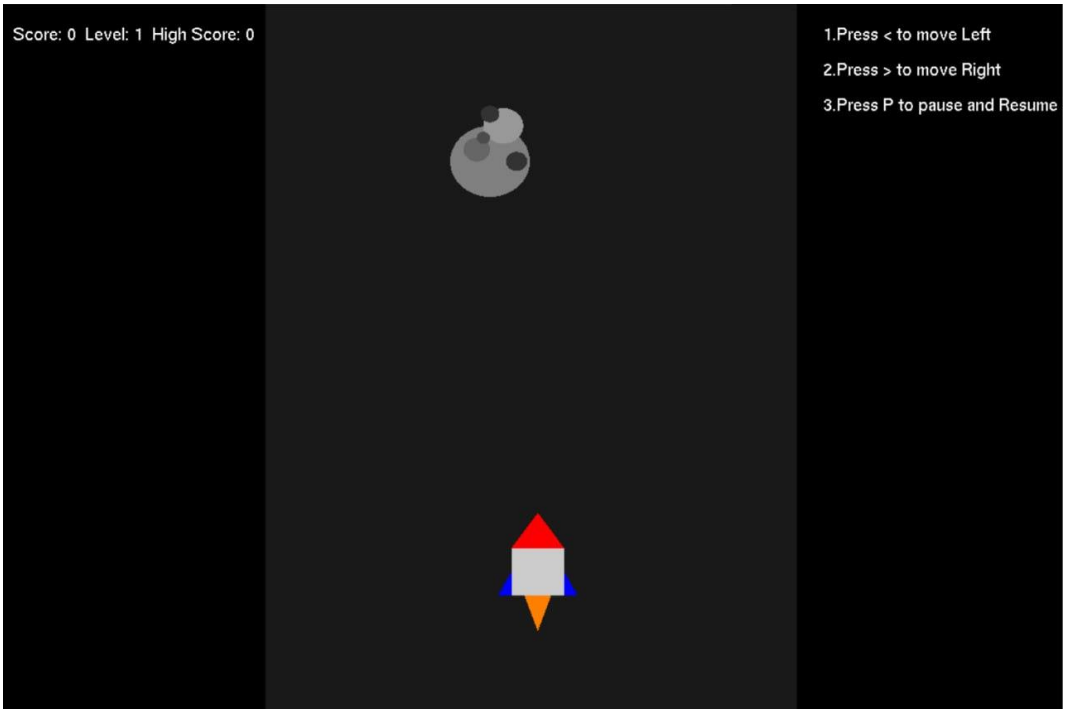

Figure 2**:** Home Window

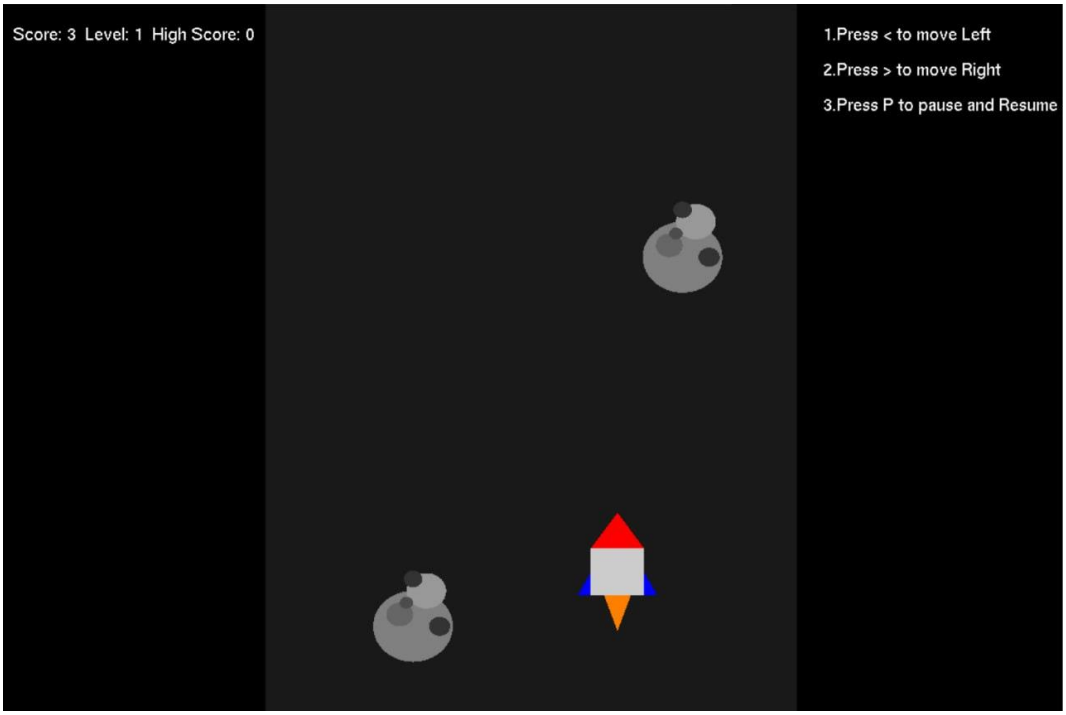

Figure 3**:** Score Increasing

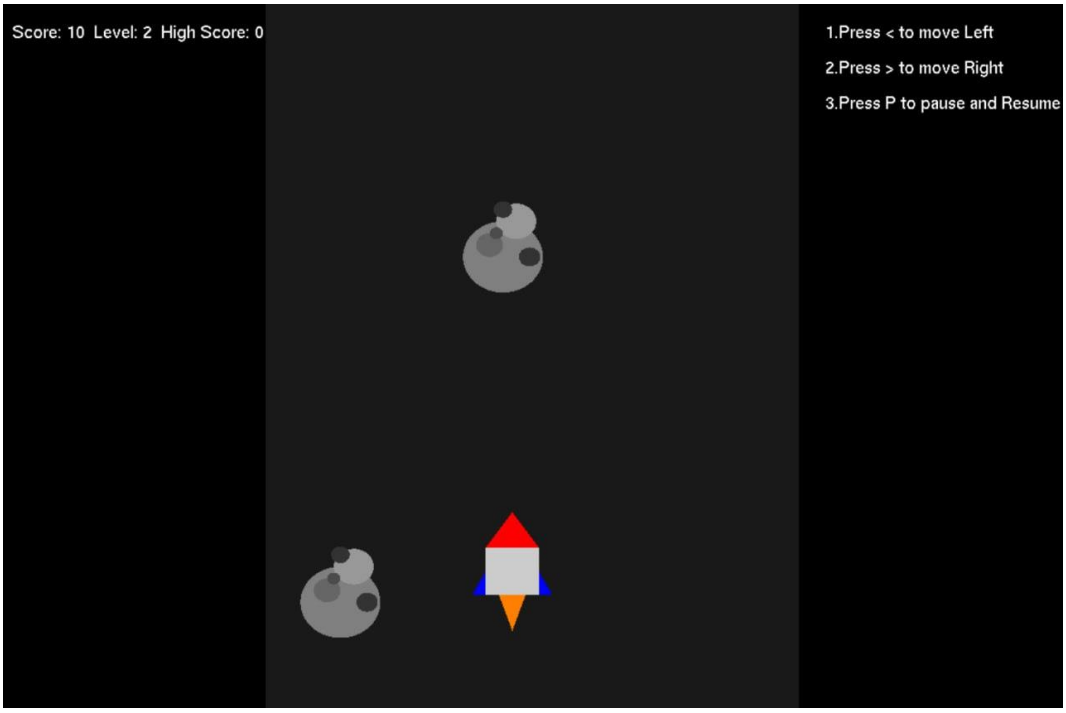

Figure 4: Level Increasing

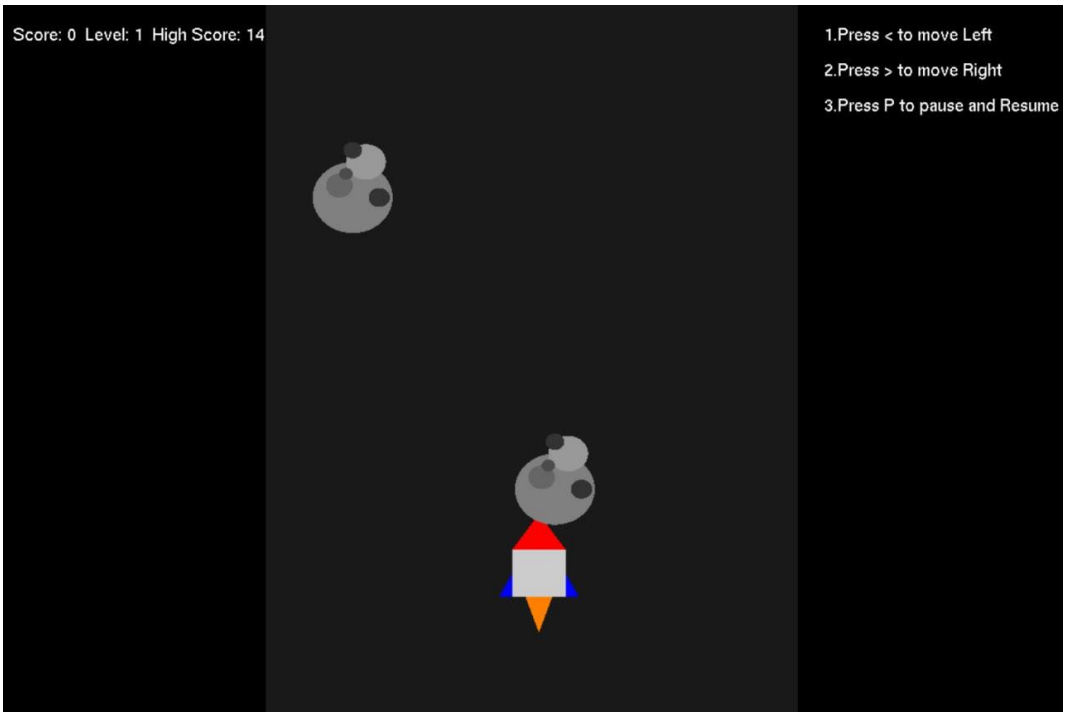

Figure 5: Collision

Figure 6: Game over

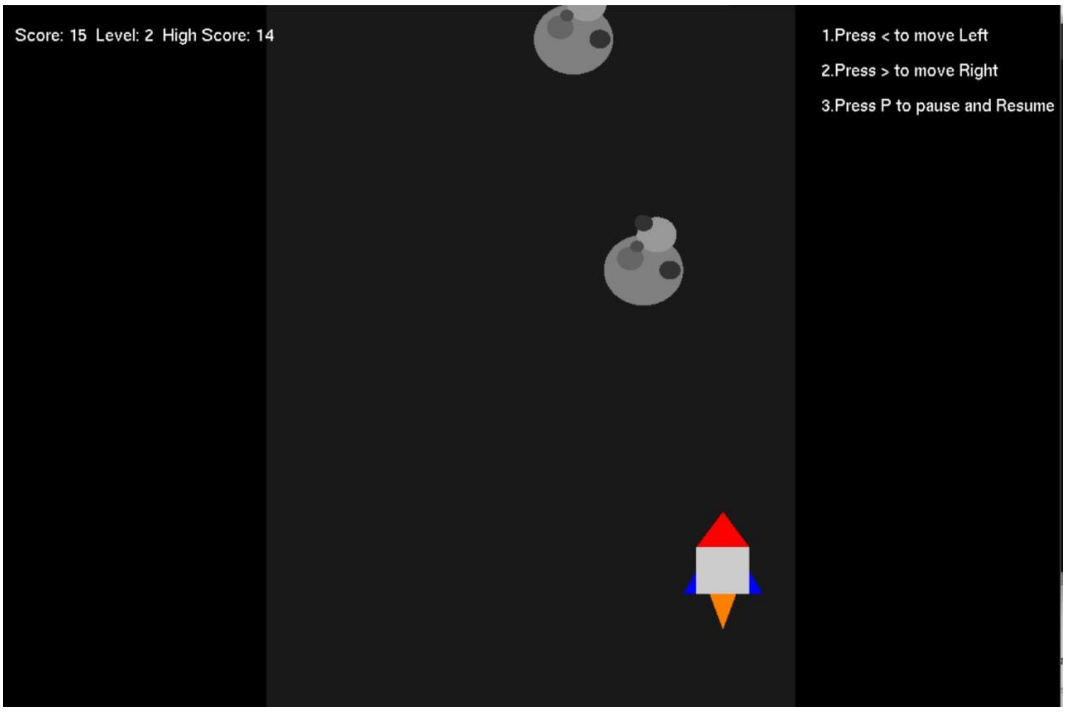

Figure 7: Restart the Game

Figure 8: High Score Updating

4.3 Testing

The 2D Asteroid Avoidance Game was thoroughly tested to ensure smooth functionality and an engaging user experience. The testing included:

1. **Functional Testing**: Verified rocket movement, asteroid spawning, collision detection, score progression, level-up logic, and high score management. All features worked as intended.
2. **Performance Testing**: Ensured the game maintained smooth rendering and responsiveness without lag or crashes, even during extended gameplay.
3. **Usability Testing**: Confirmed intuitive controls, clear on-screen instructions, and ease of use for pause, resume, and restart functions.
4. **Edge Case Testing**: Handled scenarios like asteroid overlap and boundary conditions without issues.
5. **Cross-Platform Testing**: Verified compatibility on Windows and Linux systems, ensuring consistent performance.
6. **Bug Fixes**: Addressed minor issues such as collision detection errors and improved gameplay responsiveness.

5. Conclusion

The 2D Asteroid Avoidance Game project successfully demonstrates fundamental concepts in game development, including graphics rendering, user input handling, collision detection, and game state management. By using C**,** OpenGL, and GLUT, the project showcases how these tools can be integrated to create an engaging, interactive game.

Key achievements of the project include:

1. **Smooth Graphics Rendering**: The use of OpenGL enabled the creation of visually appealing 2D game objects, such as rocket and asteroids, while maintaining smooth performance.
2. **Interactive Gameplay**: The game provides an interactive experience where the player controls the rocket and avoids incoming asteroids, with increasing difficulty as the game progresses.
3. **Score Tracking**: The game tracks the player's score and high score, providing motivation for players to continue playing and beat their previous records.
4. **Pause and Restart Functionality**: The game allows users to pause, resume, and restart, enhancing user experience by offering flexibility.
5. **File Handling**: The ability to save and load high scores adds a persistent element to the game, allowing players to track their progress over time.

6. Codebase and Resources on GitHub:

https://github.com/Nafis1e18/Developing-A-2D-Asteroid-Avoiding-Game-using-OpenGL

Signature of Supervisor