Welcome

Project Name: Rocket Shot Game

Developed by: Riya Sunil Kharade

Contents

Abstract

The Rocket Shot Game is a 2D interactive computer game developed in C using the graphics.h library. The primary objective of the game is to control a rocket, move it horizontally, and fire bullets at a moving target to score points. The game incorporates a level-based system where the difficulty increases as the player progresses, with targets moving faster and requiring greater precision to hit. The game displays the player's score, remaining shots, and target score, providing real-time feedback to enhance user engagement.

The game utilizes basic computer graphics algorithms, including line drawing for the rocket and bullets, circle drawing for targets and decorative stars, and the flood fill algorithm for coloring objects. Animation is achieved by updating the positions of the rocket, bullets, and falling targets frame by frame, creating a smooth gaming experience. Random number generation is used to vary the target's horizontal position, ensuring unpredictability and increasing challenge.

Additionally, the game integrates sound effects for firing bullets and hitting the target, enhancing user interaction and gameplay immersion. The development of this game demonstrates fundamental concepts of 2D graphics programming, collision detection, user input handling, and basic game mechanics in C. It serves as a practical implementation for beginners to understand graphics programming, event-driven programming, and game design principles.

The Rocket Shot Game not only provides entertainment but also acts as an educational tool for students to learn programming logic, computer graphics, and interactive application development. This project can be extended by adding more levels, advanced graphics effects, and more complex target patterns to further improve the gaming experience.

➢ List of Abbreviations

Abbreviation	Full Form
CPU	Central Processing Unit
RAM	Random Access Memory
IDE	Integrated Development Environment
GUI	Graphical User Interface
DOS	Disk Operating System

➢ List of Figures

➢ List of Tables

➢ List of Symbols

Symbol	Meaning / Usage
→	Denotes movement to the right
←	Denotes movement to the left
F	Key used to fire bullets
Score	Player's accumulated points
Shots	Number of remaining bullets

1. Introduction

1.1 Introduction

The Rocket Shot Game is a 2D computer game developed in C using the graphics.h library. The player controls a rocket that can move horizontally and fire bullets at a moving target to score points. The game features a level-based system, where difficulty increases as targets move faster and require greater precision. Players earn extra shots for accurate hits, and the game ends when shots run out or targets are missed.

The game incorporates visual effects such as stars, moving rockets, and colored shapes using the flood fill algorithm. Line and circle drawing algorithms create the rocket, bullets, and targets, while random number generation makes the target's position unpredictable, keeping the gameplay engaging.

Additionally, sound effects indicate firing and successful hits, enhancing the interactive experience. The Rocket Shot Game serves as a practical project for beginners to learn graphics programming, animation, collision detection, and event-driven programming, providing both an educational and enjoyable learning experience.

1.2 Motivation

The motivation for developing the Rocket Shot Game is to provide a hands-on, interactive way to learn computer graphics concepts. Beginners often find algorithms for drawing shapes, animation, and user interaction difficult to grasp through theory alone.

By creating a simple 2D game, students can apply graphics algorithms like line drawing, circle drawing, and flood fill, while gaining experience with real-time input handling, collision detection, and game logic. The project makes learning fun and engaging, encourages problem-solving and creativity, and demonstrates how visual and audio effects enhance the experience.

Overall, it bridges the gap between theory and practical application in graphics programming.

1.3 Problem Statement & Objectives

In modern computer science education, beginners often struggle to understand computer graphics concepts such as shape drawing, animation, user interaction, and collision detection. Traditional theoretical methods alone are insufficient for hands-on learning.

This project creates a simple interactive 2D game to help beginners learn these concepts practically. The game includes:

1. **User-controlled objects –** rocket responds to arrow keys.
2. **Projectile mechanics –** bullets move in a straight path and interact with targets.
3. **Target interaction** – moving targets with random positions, requiring accuracy.
4. **Game logic –** scoring system, level progression, and end-game conditions.
5. **Visual and audio effects –** colours, shapes, and sounds enhance experience.

The goal is to simulate a real-time interactive environment using basic graphics techniques and teach beginner's how graphical applications work.

1.4 Organization of the Report

The report is structured to provide a systematic understanding of the Rocket Shot Game project:

1. **Abstract** – Summarizes the project, objectives, and key features of the game.
2. **Introduction** – Explains the purpose, objectives, motivation, and problem statement behind the project.
3. **Specific Requirements** – Lists the functional and non-functional requirements needed to implement the game.
4. **Technology Stack** – Describes the tools, programming language, and algorithms used in the project.
5. **Working and Algorithm Used** – Explains the game flow, logic, and algorithms applied for graphics and gameplay.
6. **Results and Discussion** – Showcases the game screens, features, and observations from testing.
7. **Conclusion and Future Scope** – Summarizes the outcomes and suggests possible enhancements.
8. **References** – Provides sources and materials referred to during the project development.

2. Literature Survey

2.1 Survey of Existing/Similar System

There are several 2D shooting and rocket games available today, mainly developed using modern game engines like Unity, Unreal Engine, SDL, or web-based platforms using JavaScript and HTML5. These games usually focus on entertainment, featuring interactive gameplay, moving targets, scoring systems, multiple levels, and visual effects to engage players. Some educational games also introduce basic concepts of programming or logic through interactive gameplay.

However, most existing systems are complex for beginners and often require prior knowledge of advanced graphics libraries or game engines. They rarely provide a clear demonstration of fundamental graphics algorithms such as line drawing, circle drawing, or flood fill in a simple programming environment like C. Additionally, these games may not emphasize real-time input handling, collision detection, or level-based difficulty, which are key concepts for learning computer graphics practically.

From the survey, it is evident that there is a gap in beginner-friendly educational games. A simple interactive game developed using C and graphics.h can provide students with hands-on experience in graphics programming, algorithm implementation, and basic game mechanics. Such a game can also integrate visual and audio effects to enhance learning while keeping the experience fun and engaging.

Furthermore, most existing games do not focus on educational objectives like helping beginners understand how graphics programming works. They often hide the implementation details behind pre-built libraries or complex engines, limiting the learning potential for students who want to study algorithms and logic step by step.

Finally, creating a simple 2D rocket shooting game in C allows learners to experiment with modifying code, adding new features, and observing the effects immediately. This approach reinforces conceptual understanding, encourages problem-solving skills, and builds confidence in implementing interactive graphics applications, which is often missing in conventional learning resources.

2.2 Limitation Existing/Similar system or research gap

While many existing 2D shooting or rocket games provide interactive gameplay and entertainment, they have several limitations, especially from an educational perspective:

1. Complexity for Beginners: Most games are developed using advanced engines like Unity, SDL, or web-based frameworks, which require prior knowledge and make them less accessible for beginners learning graphics in C.
2. Lack of Algorithm Transparency: These systems rarely explain or demonstrate basic graphics algorithms such as line drawing, circle drawing, and flood fill, which are essential for understanding how graphics work.
3. Limited Educational Focus: Existing games often prioritize visual appeal and gameplay over teaching core concepts like collision detection, real-time input handling, or level-based difficulty progression.
4. Restricted Hands-On Learning: Many games are pre-built with complex libraries, leaving little room for beginners to experiment, modify, or understand the underlying code logic.

Research Gap: There is a clear need for a beginner-friendly educational game developed using C and graphics.h that combines interactive gameplay with fundamental learning. Such a system should allow learners to apply graphics algorithms, handle inputs, detect collisions, and implement game logic practically, bridging the gap between theory and hands-on learning in computer graphics.

2.3 Mini Project Contribution

The contributions of each team member in the Rocket Shot Game project are as follows:

- Riya Kharade
  Developed the Rocket Shot Game including the game logic, graphics, and animation. Also designed the project poster.

All team members actively participated in testing, discussion, and final project completion, ensuring the project met its objectives successfully.

3. Proposed System

3.1 Introduction

The Rocket Shot Game is a simple 2D computer game developed in C using the graphics.h library. It simulates a rocket that can be moved horizontally and fired at a target to score points. The game is designed to provide an interactive experience while demonstrating the fundamentals of computer graphics programming, animation, and user input handling.

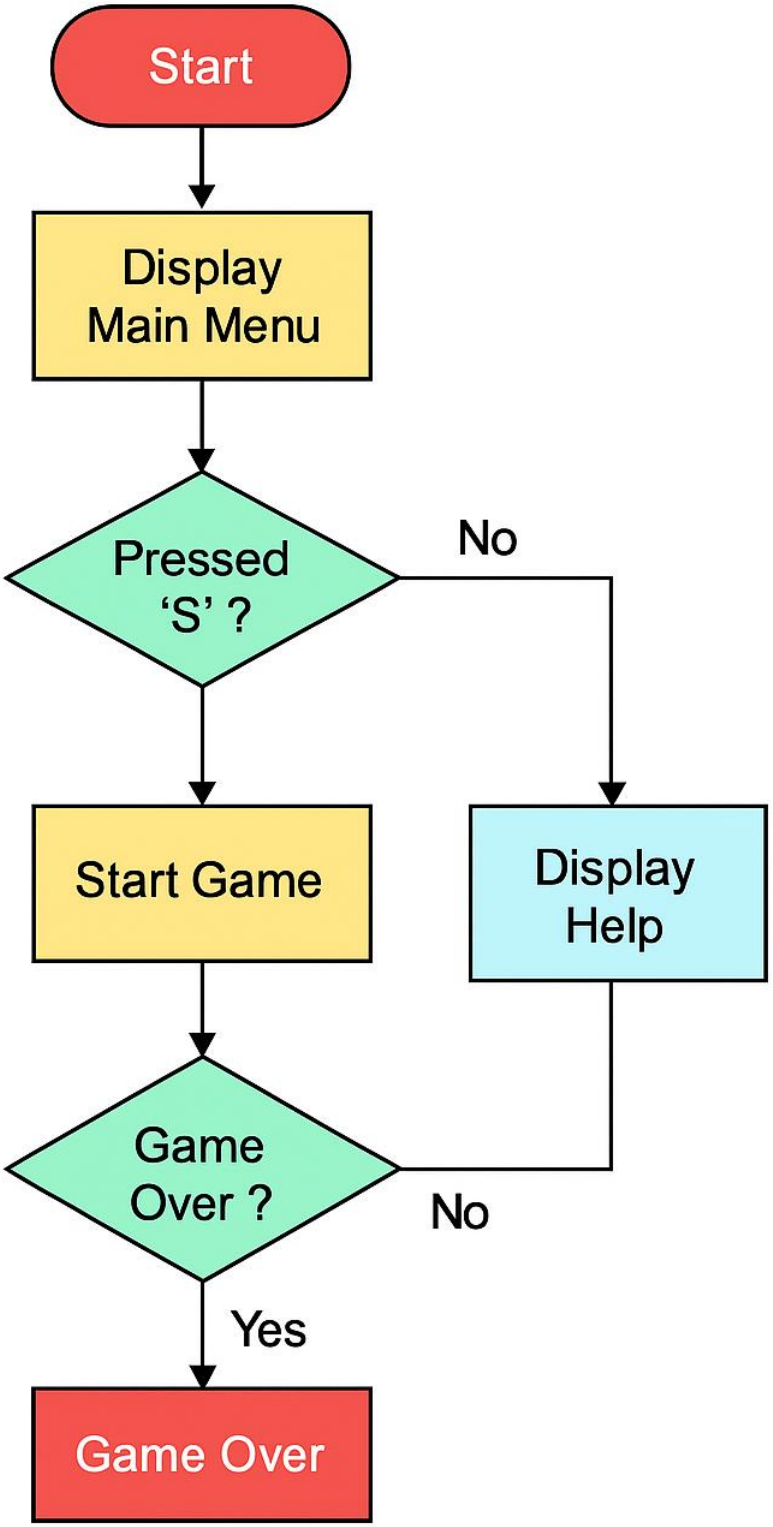
Players control the rocket using the arrow keys to move left or right and the F key to fire bullets. The primary goal is to hit a moving target and accumulate points. A level-based system increases difficulty progressively by making targets move faster and requiring greater accuracy to hit. Players are rewarded with additional shots when hitting targets accurately, while the game ends if all shots are used or the target is missed.

The game includes visual enhancements such as stars, moving rockets, and targets, which are created using line and circle drawing algorithms. The flood fill algorithm is applied for coloring shapes, providing an engaging and colorful environment. Random number generation is used to vary the target's horizontal position, ensuring unpredictable and challenging gameplay.

Audio features are also incorporated to improve user interaction. Sound effects indicate firing bullets and successful hits, adding an extra layer of engagement. These features make the game both fun and educational, helping users understand how visual and audio effects are implemented in 2D graphics applications.

The chosen technology stack and algorithms ensure that the project is beginner-friendly while providing a comprehensive learning experience. It allows students to practically explore graphics programming, collision detection, real-time input handling, and basic game mechanics, bridging the gap between theoretical learning and hands-on application. The project also serves as a foundation for further improvements, such as adding more levels, advanced effects, or multiplayer functionality.

3.2 Architecture/ Framework



3.3 Algorithms and Process Design

The Rocket Shot Game follows a structured design that ensures smooth gameplay, interactive controls, and proper scoring mechanisms. The development process integrates basic graphics algorithms, real-time input handling, and collision detection to create an engaging 2D gaming experience.

Process Flow of the Game

1. **Start Menu:** The game begins with a menu displaying options – Start (S), Help (H), and Exit (E). Players can choose to start the game, view instructions, or exit.

2. **Game Initialization:** Once the game starts, the rocket appears at the bottom of the screen. Initial parameters such as score, shots, target position, and level are set.

3. **Player Control & Shooting:**The player moves the rocket using arrow keys and fires bullets with the F key. Each bullet moves upward, and the system continuously checks for collisions with the target.

4. **Target Mechanics:** Targets move downward continuously, with their horizontal position randomly generated using the rand() function. The speed of movement increases with higher levels.

5. **Collision Detection:** The game checks if the bullet hits the target. If a hit occurs, the player's score increases, and an extra shot is awarded. Missing a target decreases the number of remaining shots.

6. **Level Progression:**When the player reaches a target score, the level increases, making the game progressively more challenging by increasing target speed and requiring more accuracy.

7. **End Condition:**
   The game ends when the rocket runs out of shots or the target reaches the bottom of the screen. A Game Over screen is displayed with the final score.

Algorithms Used

- **Line Drawing Algorithm:**Draws the rocket, bullets, and decorative elements on the screen using straight lines. This algorithm ensures that shapes are rendered accurately and efficiently, forming the basic structure of the game objects.

- **Circle Drawing Algorithm:** Creates circular targets and smaller visual details, like explosions or stars. It allows smooth and precise rendering of circular objects, enhancing the visual appeal of the game.

- **Flood Fill Algorithm:** Fills colors inside the rocket, targets, and stars to make objects visually distinguishable. This algorithm helps in adding vibrancy to the game and clearly separates different components.

- **Random Number Generation (rand()):** Determines random target positions to make gameplay unpredictable and challenging. This ensures that players cannot anticipate target locations, adding excitement and variability to each session.

- **Collision Detection Logic:** Checks bullet-target intersections to determine hits or misses. It updates the player's score and remaining shots, enabling accurate game mechanics and fair gameplay.

- **Keyboard Input Handling (getch(), kbhit()):** Captures real-time rocket movement and shooting actions from the player. This allows smooth control over the rocket, making the game interactive and responsive.

- **Delay Function (delay()):** Controls the speed of animations and smooth transitions of rockets, bullets, and targets. It ensures that movements appear natural and the gameplay is visually comfortable.

- **Sound Effects (sound(), nosound()):** Provides audio feedback for firing bullets and hitting targets, improving user engagement. These effects make the game more immersive and enhance the overall interactive experience.

This process and algorithm design ensures that the game operates efficiently, provides real-time interactivity, and progressively challenges the player, making it an effective educational tool for understanding graphics programming and game logic.

3.4 Details of Hardware & Software

Hardware Requirements

Component	Specification / Requirement	Purpose / Usage
Processor	Intel i3 or higher	To run the Turbo C/C++ IDE and game smoothly
RAM	4 GB or higher	Ensures smooth execution of the game
Storage	100 MB free space	To store program files and resources
Display	1024 x 768 resolution or higher	Proper visualization of graphics and animations
Keyboard & Mouse	Standard	To provide input for rocket movement and firing
Sound Card	Integrated or external	To play sound effects for firing and hits

Software Requirements

Component / Tool	Version / Specification	Purpose / Usage
Programming Language	C	Main language to develop game logic and graphics handling
Development Environment	Turbo C/C++ IDE	Provides support for graphics.h and DOS-based execution
Graphics Library	graphics.h	Used for drawing lines, circles, rockets, bullets, and targets
Sound Library	dos.h (sound(), nosound())	To add audio feedback during firing and target hits
Algorithms	Line, Circle, Flood Fill	For drawing and coloring objects, and creating animations
Randomization Function	rand()	To generate unpredictable target positions

3.5 Experiment and Results

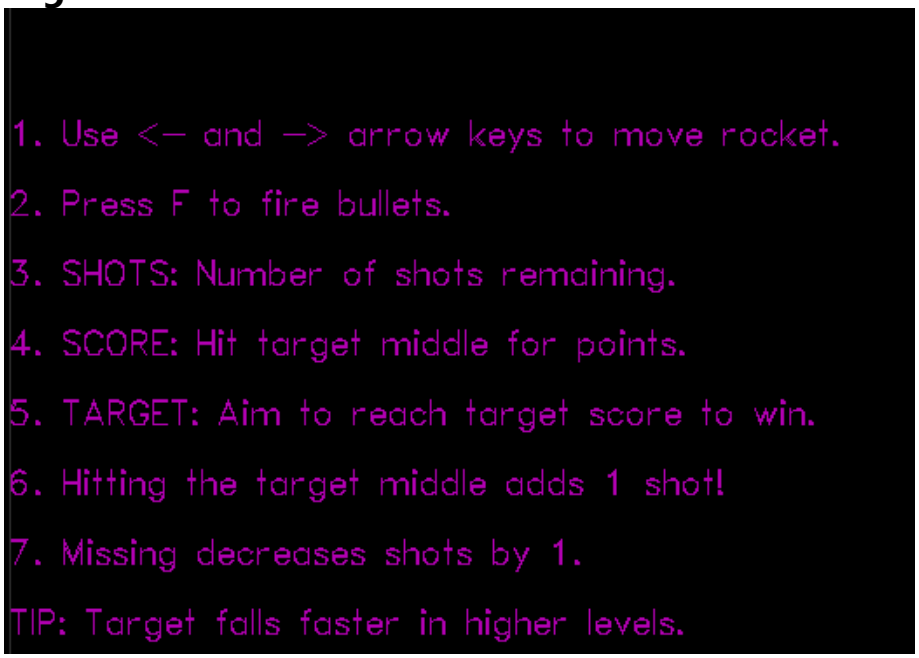
The "Shooting Game Project" was successfully implemented and executed using C language and the graphics.h library. The game offers an interactive experience with multiple levels, smooth controls, and visual effects. Below are the detailed results and discussion based on each stage of the game:

Stage	Description	Discussion
1. Main Page	The main page displays three options — Start (S), Help (H), and Exit (E).	The main menu provides a simple and user-friendly interface for the player to begin or understand the game.
2. Help Page	When the player clicks on H, the help page opens displaying the controls and rules of the game.	This helps new users understand how to play and improves the overall accessibility of the game.
3. Loading Screen	After pressing S, a loading animation appears before the game starts.	The loading screen gives the player a smooth transition into the game and improves the visual experience.
4. Game Level 1 (Gameplay Screenshot)	The player controls a shooter and aims to hit targets appearing randomly on the screen.	Level 1 introduces the basic gameplay mechanics and helps players get familiar with controls.
5. Level Up Display	When the player scores enough points, a "Level Up" message is shown on the screen.	This motivates the player and indicates progress in the game.
6. Level 6 Screenshot	The difficulty increases with higher levels, and enemies appear more frequently.	It adds challenge and engagement, keeping the player interested throughout the game.
7. Game Over Screen	Once the player loses, the "Game Over" message is displayed along with the final score.	It marks the end of gameplay and summarizes the player's performance.

- **Main Menu**

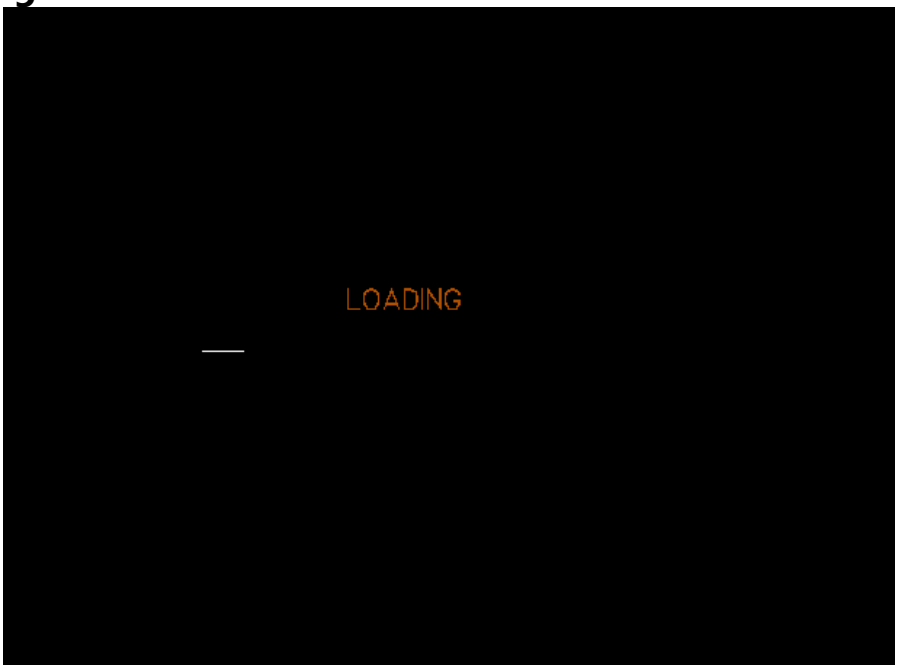
The main page of the game displays three options Start (S), Help (H), and Exit (E). It acts as the entry point for the player. The interface is simple and user-friendly, allowing easy navigation.

- **Help Page**



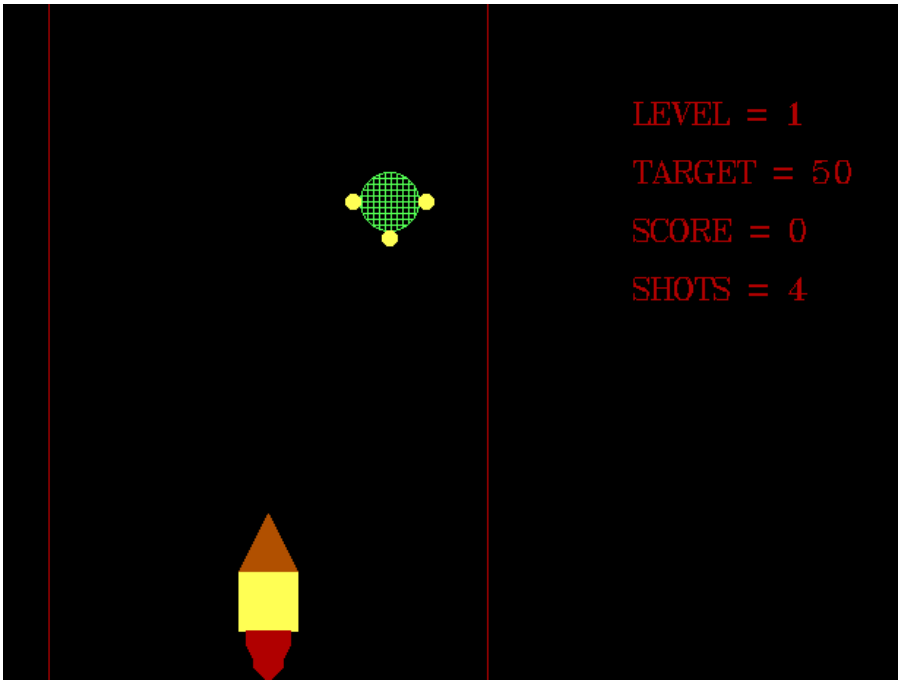
When the player presses H, the help page opens and provides all the necessary information about how to play the game, including controls and rules. This feature helps beginners understand the gameplay before starting.

- **Loading Screen**



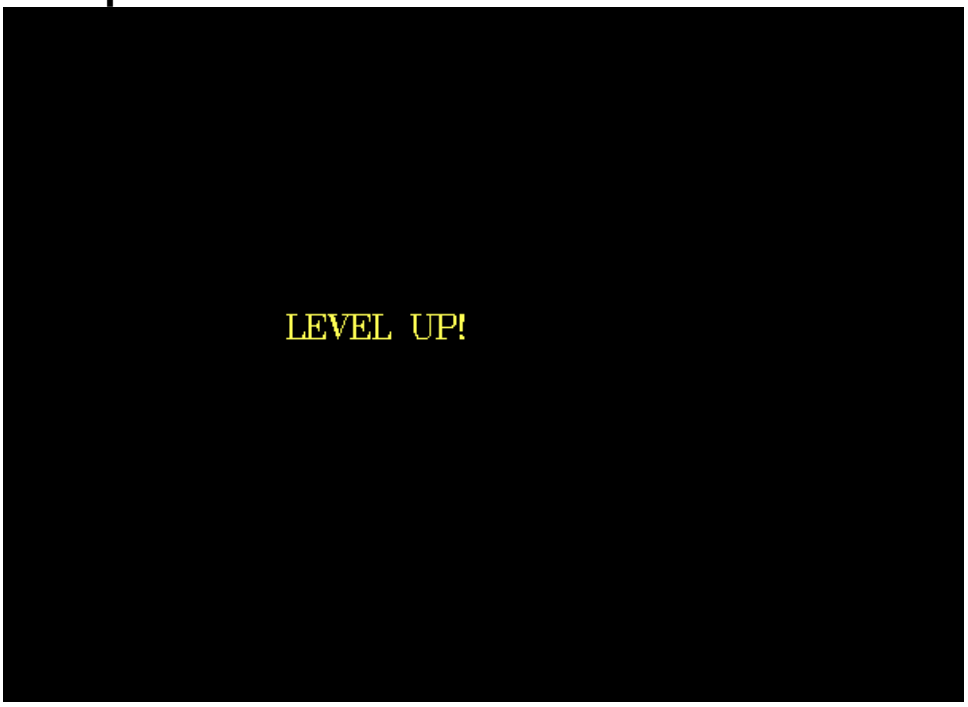
After pressing S, a loading animation is shown before the game begins. This creates a smooth transition from the main menu to the gameplay, making the user experience better.

- **Game Level 1:**



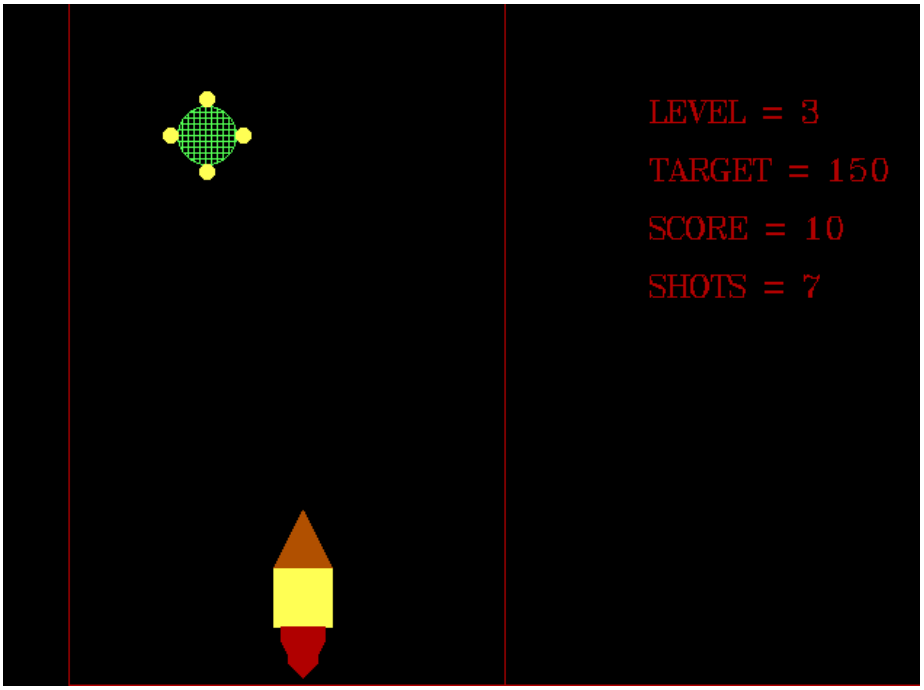
Once the game starts, the player controls the shooter and fires bullets at targets appearing on the screen. The graphics and timing functions make the gameplay interactive and engaging.

Level Up screen



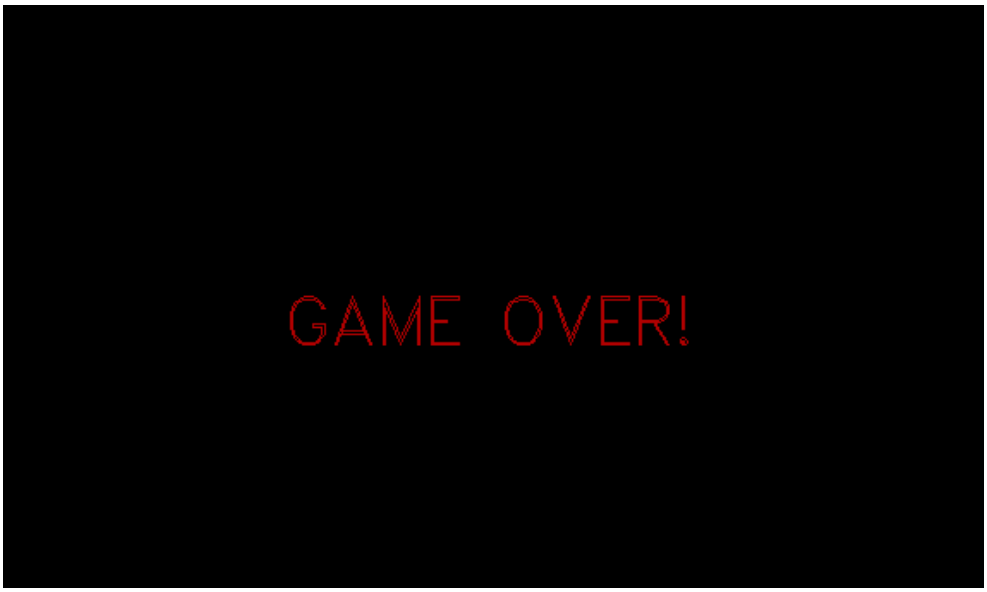
When the player achieves a specific score, a “Level Up” message appears, showing the progress to the next level. This motivates the player and makes the game more interesting.

- **Level 3**



As the player advances, the difficulty increases. Targets move faster, making it more challenging. This keeps the player focused and improves engagement.

- **Game Over screen**



When the player loses all chances, a “Game Over” message appears along with the final score. This marks the end of the game and helps evaluate the player’s performance.

3.6 Conclusion and Future work

Conclusion

The Rocket Shot Game successfully demonstrates the practical application of basic computer graphics concepts, including line and circle drawing, flood fill, and randomization. The project effectively integrates real-time user input, collision detection, scoring, and level progression, providing an engaging interactive experience. By combining visual effects, audio feedback, and smooth animations, the game serves as a valuable learning tool for beginners to understand graphics programming and game design principles in a hands-on manner.

Additionally, the project highlights how simple algorithms can be combined to create a fully functional 2D game. It reinforces logical thinking, problem-solving, and creativity while offering an enjoyable learning experience. The level-based system and progressively increasing difficulty ensure that players remain challenged and engaged throughout gameplay.

Future Scope

The Rocket Shot Game can be further enhanced by upgrading to modern graphics libraries like SDL, OpenGL, or SFML for richer visual effects and smoother animations. Additional features such as more levels, moving obstacles, power-ups, background music, multiplayer mode, and cross-platform support can be integrated to improve gameplay and make it suitable for a wider audience. These enhancements will also provide deeper learning opportunities for students in advanced graphics and interactive application development.

References

- E. Balagurusamy, *Computer Graphics and Multimedia*, McGraw Hill Education, 2018.
- Donald Hearn and M. Pauline Baker, *Computer Graphics: C Version*, Pearson Education, 2017.
- Bjarne Stroustrup, *The C++ Programming Language*, Addison-Wesley, 2013.
- TutorialsPoint, *Computer Graphics in C*, https://www.tutorialspoint.com/computer_graphics/index.htm
- GeeksforGeeks, *Graphics in C*, https://www.geeksforgeeks.org/graphics-in-c/
- Stack Overflow, *Discussions on Turbo C graphics.h usage*, https://stackoverflow.com
- Online resources for BGI graphics library functions and sound() / nosound() implementations
- https://github.com/Riya-Kharade/Rocket-Shot-Game