A Project Report

On

Space War

Submitted in partial fulfillment of the requirement of

project subject

BCE 3009

OF

BACHELOR IN COMPUTER ENGINEERING

Submitted To

Purbanchal University

Biratnagar, Nepal

Submitted By

Tulsa Bhat

Payal Joshi

Gaurab Aryal

KANTIPUR CITY COLLEGE

Putalisadak, Kathmandu

DATE:(03/2025)

A Project Report

on

Space War

Submitted in partial fulfillment of the requirement of

project subject

(subject code)

BACHELOR COMPUTER ENGINEERING

Submitted to

Purbanchal University

Biratnagar, Nepal

Submitted By

Tulsa Bhat

Payal Joshi

Gaurab Aryal

Project Supervisor

Er. Pramish Karki

KANTIPUR CITY COLLEGE

Putalisadak, Kathmandu

DATE:(03/2025)

Abstract

The purpose of this project is to design and develop a space war game using C/C++ programming languages, incorporating basic graphical features for an enhanced visual experience, without relying on any external libraries or AI algorithms. This team-based project focuses on creating a graphical user interface (GUI) that allows a player to control a spaceship, battle static enemy ships, and navigate obstacles. The motivation behind this project is to understand the fundamentals of game development, including user input handling, graphical rendering, and game mechanics, all while avoiding the use of advanced libraries or AI-driven enemies. Instead, the game features predefined static enemy patterns that are activated at set intervals. The game is implemented using custom C/C++ code for handling user input, rendering basic graphics, controlling game flow, and detecting collisions. The graphical elements are drawn from scratch using the built-in graphics capabilities of C/C++, providing a visually engaging experience without relying on external libraries. The project involves a comprehensive team effort to design and develop a fully functional game where the player must destroy enemies and avoid obstacles while progressing through levels. Evidence of the project includes the final source code, a working game demo, and a report detailing the development process, design decisions, and the challenges faced by the team.

Acknowledgements

I would like to express my sincere gratitude to everyone who contributed to the successful completion of this project and the preparation of this report.

First and foremost, I extend my heartfelt thanks to **Er. Pramish Karki**, our supervisor, for his constant guidance, valuable feedback, and unwavering support throughout the duration of the project. His insights and encouragement were instrumental in shaping our work and ensuring its success.

I would also like to express my deep appreciation to my teammates: **Tulsa Bhat**, **Gaurab Aryal**, and **Payal Joshi**, for their collaboration, hard work, and dedication. Our collective effort, teamwork, and commitment were essential to the completion of this project.

Additionally, I would like to acknowledge the support provided by the faculty and staff of Computer Engineering, who helped guide us through various stages of this project and provided the resources necessary to complete it. Special thanks to the entire **Computer Engineering** for fostering an environment conducive to learning and growth.

I would also like to thank my family for their consistent encouragement, support, and understanding throughout the course of this project. Their belief in me kept me motivated and focused on the task at hand.

Lastly, I extend my gratitude to all my friends for their moral support and motivation. Their encouragement was invaluable in helping me stay focused on completing the project.

The Team Members

Tulsa Bhat

Payal Joshi

Gaurab Aryal

Preface

This report presents the design, development, and implementation of the **Space War Game**, a console-based graphical game created as part of our group project for the **Game Development** course under the guidance of **Er. Pramish Karki**. The main objective of this project was to gain practical experience in game development using **C/C++ programming languages** and explore essential concepts such as user input handling, graphical rendering, and collision detection.

In this project, we developed a game in which the player controls a spaceship, navigating through space, avoiding obstacles, and battling static enemy ships. The game does not rely on AI-driven enemies but instead uses predefined enemy patterns to provide the player with a challenging experience.

The project was executed by a team of three members: **Tulsa Bhat**, **Gaurab Aryal** and **Payal Joshi**. Each team member contributed to the project, working collaboratively on various tasks such as Coding, File Management , GUI and Documentation. This project is a reflection of our teamwork, creativity, and dedication to learning.

We hope this project demonstrates our ability to apply the knowledge we've gained and serves as an example of our commitment to software development.

DECLARATION

I declare that this project report titled **Space War Game** submitted in partial fulfillment of the **Bachelor in Computer Engineering** is a record of original work carried out by me under the supervision of **Er. Pramish Karki**, and has not been submitted for the award of any other degree or diploma, in this or any other institution or university. In keeping with ethical practices in reporting academic work, due acknowledgements have been made wherever the findings of others have been cited.

Tulsa Bhat,

Gaurab Aryal,

Payal Joshi

31st march,2025

SUPERVISOR'S APPROVAL

Space War Game undertaken and demonstrated by **Tulsa Bhat**, **Gaurab Aryal**, and **Payal Joshi** has been successfully completed under my supervision as a partial fulfillment of the requirements for the degree of **BE Computer, 3rd Semester** under **Purbanchal University**. I, henceforth, approve this project to be awarded the certificate by the concerned authority.

During supervision, I found the students hardworking, skilled, and ready to undertake any professional work related to this field in the future.

Er. Pramish Karki
Project Supervisor
Kantipur City College

Date:

CERTIFICATE FROM DEPARTMENT

Following the Supervisor's Approval and Examiners' Acceptance, the project entitled **Space War Game** submitted by **Tulsa Bhat**, **Gaurab Aryal**, and **Payal Joshi** as a partial fulfillment of the requirements for the degree of **BE Computer, 3rd Semester** under **Purbanchal University**, has been officially awarded this certificate.

I wish the students all the best for their future endeavors.

[Official Signature + Stamp]

Er. Subash Rajkarnikar

Senior Assistant Professor

Computer's HOD

Date: **[Date]**

Table of Contents

1. Introduction:

The **Space War Game** is an interactive 2D **console-based game** developed using both **C and C++** programming languages, aimed at providing an engaging and exciting gaming experience. In this game, the player controls a spaceship navigating through space, with the goal of avoiding obstacles and defeating enemy ships. The game features simple yet challenging mechanics, where the player must maneuver their spaceship while dealing with static enemies that pose a threat. The project primarily focuses on creating a basic graphical interface and implementing core gameplay functions such as **user input handling**, **collision detection**, and **basic animation**.

One of the key decisions in the development process was the **choice of a console-based interface over a GUI-based one**. While GUI-based games offer more visually appealing graphics and modern user interfaces, console-based games are simpler to implement, more resource-efficient, and ideal for understanding **fundamental programming concepts**. This trade-off allows the development team to focus on **game logic, data structures, and memory management** rather than interface design, making the console-based approach more appropriate for an educational and foundational project like this.

This project was undertaken to explore **fundamental game development concepts** such as **programming logic**, **game flow**, and **graphical rendering**. By building the Space War Game, the team gained valuable experience in **coding**, **debugging**, and creating an interactive application using **client-side programming**, without relying on external libraries or AI-driven features. The game's design emphasizes **simplicity** while providing an **enjoyable challenge** for players, showcasing the team's understanding of **game development processes** and **problem-solving techniques** within the scope of C/C++.

1.1 Overview

The Space War Game is a 2D arcade-style game developed using both C and C++ programming languages, where the player controls a spaceship navigating through space and battling enemy ships. The goal of the game is to survive and defeat the enemies while avoiding obstacles. The game's design features simple yet engaging gameplay mechanics, such as player movement, projectile firing, and enemy interaction. Players must use their skills to navigate the spaceship, destroy enemies, and avoid damage.
Developed using basic graphical elements without the use of external libraries or advanced AI, the game highlights fundamental programming concepts like loops, conditionals, collision detection, and object-oriented principles in C++. The project demonstrates the implementation of a console-based interactive game while emphasizing the use of core game development techniques, providing a fun and challenging experience for players. The combination of C and C++ allows for efficient memory management and clean code, making it an excellent learning tool for game development basics.

1.2 Problem Statement

In the current landscape of game development, many space-themed games rely on complex AI-driven enemies and advanced graphics libraries, making them difficult to understand and develop for beginners. These existing systems often require deep knowledge of external libraries, sophisticated algorithms, and graphical engines to implement interactive and visually engaging features.

For many developers—**especially beginners in game development or learners of C and C++ programming languages**—this complexity presents a steep learning curve. The challenge lies in creating a **simple,**

understandable, yet engaging game that reinforces fundamental programming skills without being overwhelmed by advanced tools and technologies.

This project aims to address this gap by developing a **Space War Game** that focuses on **core game development concepts**, using **C and C++** exclusively, and **avoiding external libraries or AI-driven systems**. The objective is to create an **accessible, educational platform** that introduces beginners to the **fundamentals of game design**, **user interaction**, and **basic graphical rendering** in a manageable environment.

By adopting a **console-based approach**, the game offers an **intuitive and hands-on learning experience**, helping learners grasp essential coding principles while enjoying the process of building a playable game. The result is a project that not only entertains but also **serves as a strong learning tool** for newcomers to the world of programming and game development.

1.3 Objective

The major objectives of the Space War Game project are as follows:

- To design and develop a 2D space-themed game using C and C++ programming languages.
- To implement core game mechanics such as spaceship movement, shooting, and enemy ship behaviors.
- To create an interactive gameplay experience while ensuring the game remains simple and accessible.
- To implement collision detection and response between the player's spaceship and enemy objects.
- To enhance the understanding of basic game development principles such as graphics rendering and handling user inputs.

- To develop the game without relying on external libraries or AI, focusing on fundamental programming techniques.
- To provide a hands-on learning experience in coding, debugging, and interactive game design.

1.4 Features

To effectively manage the development process, the features of the **Space War Game** are categorized into **Core Features** and **Stretch Features**:

Core Features (Must-Have)

These features are essential for the game to function and meet its primary educational goals:

- **Player Movement**: Control the spaceship's movement (left, right) using keyboard input.
- **Projectile Shooting**: Allow the player to fire bullets from the spaceship to destroy enemies.
- **Enemy Ships**: Introduce static enemy ships that follow simple movement patterns.
- **Collision Detection**: Detect and respond to collisions between player bullets and enemies, or between enemies and the player's spaceship.
- **Score Tracking**: Display and update the player's score based on destroyed enemies.
- **Health System**: Include a basic health/lives system, reducing health on collisions.
- **Game Over Logic**: End the game when player health reaches zero, and display a game over message.

- **Basic Console Graphics**: Use ASCII or character-based rendering to display game elements.
- **User Input Handling**: Efficient keyboard input handling for real-time gameplay.
- **Target Frame Rate**: Maintain a **minimum of 30 FPS** to ensure smooth and responsive gameplay.

Stretch Features (Nice-to-Have)

These features enhance the gameplay experience but are not critical to the game's basic functionality:

- **Increasing Difficulty**: Gradually increase enemy speed or quantity as the player's score increases.
- **High Score Saving**: Store and display the highest score achieved along with the player's name.
- **Restart Option**: Provide an option to restart the game after game over without restarting the application.
- **User Login/Records**: Allow basic user profiles to track individual high scores or game history.
- **Simple Animation Effects**: Add movement or flicker animations using ASCII characters.
- **Modular Code Design**: Implement modular, reusable functions or classes to improve code maintainability.

1.5 Significance

The Space War Game project holds significant educational value, providing hands-on experience in the fundamental concepts of game development, programming, and problem-solving. By developing this game using C and

C++, the project helps enhance understanding of basic programming concepts such as loops, conditionals, collision detection, and user input handling. This simple game design offers an accessible way for beginners to learn how games are built from the ground up, making it an excellent starting point for those interested in pursuing further studies in game development.

Moreover, the project emphasizes the importance of efficient coding, memory management, and creating interactive user experiences. The absence of AI and external libraries encourages developers to focus on the core mechanics and logic of the game, fostering creativity and resourcefulness in programming. Overall, the Space War Game serves as both an engaging and educational tool for learners to explore game development in a manageable yet impactful way.

1.6 Scope and Limitations

The scope of the Space War Game project is focused on developing a simple 2D arcade-style game using C and C++ programming languages. The project aims to provide a platform for understanding and implementing core game development concepts such as user input handling, graphics rendering, collision detection, and basic game mechanics. This game will be console-based, using text-based graphics and focusing on fundamental programming skills without relying on external libraries or AI-driven features. The project is designed to serve as an educational tool for beginners in game development, offering a solid foundation for further exploration in more complex game creation.

Here are the some limitations of the Space War:

- No AI Implementation: The enemies in the game follow simple predefined patterns, and there is no advanced AI for enemy movement or behavior.

- Basic Graphics: The game uses simple text-based graphics and does not feature advanced visual effects or 3D rendering.
- Limited Gameplay Features: The game is focused on basic mechanics such as movement, shooting, and enemy destruction without additional features like multiplayer support or complex game levels.
- No Sound Effects: The game may not include advanced sound or music, as the focus is on core programming elements.
- Platform Dependent: The game is designed to run on a console-based system and may not be compatible with graphical user interfaces (GUIs) or web-based platforms.
- Lack of Advanced Game Levels: The game does not feature multiple complex levels or dynamic environments; it remains simple to allow users to focus on fundamental coding skills.

1.7 Organization of the Document

This document is structured to provide a comprehensive overview of the Space War Game project, from the initial concept to the final implementation. The report begins with an Introduction, which outlines the purpose, objectives, and significance of the project. Following this, the Problem Statement section discusses the challenges addressed by the game, such as the need for a simple, accessible game development experience for beginners. The Features section highlights the key elements that make up the gameplay, followed by the Scope and Limitations of the project, explaining the boundaries and constraints of the game.

The document then delves into the core technical aspects, such as the design methodology, coding structure, and development process. The Conclusion summarizes the key takeaways from the project and offers insights into potential future improvements. Throughout the report, readers will gain an

understanding of both the theoretical and practical aspects of building a simple yet engaging game using C and C++ programming languages.

2. Literature Review

Space war games have been popular for decades, with many different versions developed over time. These games typically involve a player controlling a spaceship to fight off enemies in space. Below are a few examples of existing space war games, their operation, and their strengths and weaknesses.

2.1 Space Invaders (1978)

Overview: Space Invaders is one of the first space shooter games, where players control a cannon to shoot down waves of descending aliens.

Pros:

- Easy to understand and play.
- Simple but addictive gameplay.
- Can run on low hardware.

Cons:

- Repetitive gameplay with predictable enemies.
- No power-ups or upgrades.
- Limited graphics and features

2.2 Galaga (1981)

Overview: Galaga is a more advanced version of Space Invaders, where players shoot down alien ships, capture enemies, and collect power-ups.

Pros:

- Engaging with dynamic enemy movement and patterns.
- Power-ups add strategy.
- Better graphics than Space Invaders.

Cons:

- Can be overwhelming for new players.
- No multiplayer options.
- Gameplay can still become repetitive

2.3 Star Fox (1993)

Overview: Star Fox is a 3D space shooter that lets players fly a spacecraft and battle enemies in a 3D environment.

Pros:

- Immersive 3D graphics.
- Multiple aircraft options.
- Challenging AI and diverse missions.

Cons:

- Requires more powerful hardware.
- Can be hard to learn for new players.
- Limited multiplayer options

2.4 Asteroids (1979)

Overview: In Asteroids, players control a spaceship and destroy asteroids in space. The game uses simple vector graphics.

Pros:

- Simple and addictive gameplay.
- Endless replayability.
- Classic design.

Cons:

- No enemies, just asteroids.
- Lack of power-ups or upgrades.
- Can become repetitive.

3. Methodology

The development of the **Space War Game** follows a structured, yet flexible methodology suited to the nature of game development. It integrates iterative testing, rapid feedback, and modular coding. The team adopted a **Prototyping-Based Agile Model**, which emphasizes continuous improvement through user testing and multiple working builds.

3.1 Development Approach: Agile + Prototyping

The game was developed over multiple short iterations, or sprints, each focused on implementing and refining specific game features. Early prototypes helped validate gameplay mechanics before moving on to more complex features.

Key Phases in Development:

I. Requirement Planning:

- Core features such as player controls, enemy behavior, collision detection, and scoring were identified and prioritized.
- Features were categorized as:
- **Core Features**: Player movement, shooting, enemy interaction, collision detection.
- **Stretch Features**: Game-over screen, progressive difficulty, high-score tracking.

II. Rapid Prototyping:

- Simple working versions of the game were created early on.
- The team tested and refined movement, shooting, and visual layout using console output.

III. Incremental Feature Development:

- Game mechanics were built in phases:
- Player spaceship movement and screen boundaries.
- Projectile firing logic and enemy ship behavior.
- Collision detection and scoring.
- C++ classes were used to encapsulate game entities like Player, Enemy, and Bullet.

IV. Testing and Feedback:

- Each prototype was tested for logical correctness, user input response, and screen rendering.
- Informal peer feedback helped fine-tune the difficulty level and gameplay responsiveness.

V. Integration and Optimization:

- All core modules were combined into a full game loop.
- Focus was placed on achieving a **minimum frame rate of 30 FPS** for smooth gameplay.
- Visuals and animations were enhanced using ASCII art to suit the console environment.

VI. Final Polishing and Maintenance:

- The final version was cleaned of bugs, optimized, and prepared for deployment.
- Future updates may include new enemies, scoring levels, or saving features based on feedback.

3.2 Technologies and Tools Used

The development of the Space War Game will require the use of specific programming languages, software tools, and file management practices. The tools selected will help streamline the development process and ensure the project is completed efficiently.

3.2.1 Programming Language: C and C++

- C and C++ are chosen for their ability to handle low-level system processes and manage resources effectively.
- C will be used for basic gameplay mechanics and game logic, while C++ will enable the use of Object-Oriented Programming (OOP) principles to organize the code and manage the player, enemy, and bullet objects.

3.2.2 File Management

- Git will be used for version control and collaboration. The project will be hosted on GitHub, allowing multiple team members to work on different parts of the game without overwriting each other's work.
- File Structure: Source code files will be organized into directories for different components (e.g., one for player mechanics, one for enemy behavior, etc.), which helps maintain an organized structure.

3.2.3 Software Used: Dev C++

- Dev C++ will be used as the integrated development environment (IDE) for writing, debugging, and testing the C and C++ code. It is a lightweight, user-friendly IDE that supports compiling and executing C and C++ programs, making it suitable for developing the game.
- GCC (GNU Compiler Collection) will be used to compile the C and C++ code into executable files. Dev C++ integrates seamlessly with GCC to build the game on Windows systems.
- Windows Console will be used to run and display the game, utilizing text-based graphics and basic ASCII characters.

3.3 Assignment of Roles and Responsibilities

Member	**Role**	**Responsibilities**
Gaurab Aryal	Game Operation	- Responsible for core gameplay mechanics:

		player movement, shooting, enemy behavior, and collision detection. - Manages the game loop and overall game logic.
Payal Joshi	GUI (Graphical User Interface)	- Designs and implements the user interface, including game status, score, health display, and "Game Over" screens.
Tulsa Bhat	File Handling & User Records	- Manages file handling for saving and loading player history, usernames, and high scores. - Documentation: Handles documentation for the project report, including file handling and user record systems for future updates and maintenance.

4. System Analysis

4.1 Requirements Analysis

In this phase, the Space War Game project's system requirements were thoroughly examined. The primary goal was to ensure that all necessary features, performance standards, and constraints were clearly defined and achievable within the project's scope.

4.2. Requirements Gathering

Requirement Category	**Details**
Objective	Define core game functionalities and features.
Stakeholders	Team members, supervisors, and users.

Game Features	Space shooting, enemy behavior, score tracking, etc.
Technical Constraints	Develop using C/C++ with minimal external dependencies.
User Experience	Easy controls, intuitive gameplay, and responsive UI.

4.3. Functional Requirements

Feature	Description
Player Movement	Move the spaceship left, right .
Shooting Mechanism	Shoot bullets using a key (e.g., spacebar).
Enemy Behavior	Enemies move and shoot, and follow set patterns.
Scorekeeping	Track and display score when enemies are destroyed.
Health Bar	Show player's health, which decreases on enemy contact.
Game Over	Display "Game Over" when health reaches zero.
High Scores	Save highest scores with usernames.
Restart Option	Provide a restart option after the game

Feature	Description
Player Movement	Move the spaceship left, right .
Shooting Mechanism	Shoot bullets using a key (e.g., spacebar).
Enemy Behavior	Enemies move and shoot, and follow set patterns.
Scorekeeping	Track and display score when enemies are destroyed.
	ends.

4.1.3 Non-Functional Requirements

Requirement Category	Description

Performance	Game should run smoothly with minimal lag or delays.
Usability	Intuitive controls and easy-to-navigate UI.
Portability	Compatible across platforms (Windows, Linux, etc.).
Reliability	Stable with no frequent crashes or errors.
Security	Secure storage of user data (e.g., high scores).
Maintainability	Code should be clean, documented, and easy to modify.
Scalability	Easily supports future feature additions.
Availability	Game should load and restart quickly without delay.
Safety	Safe gameplay suitable for all users.

4.4 Feasibility Study

4.1.1 Technical Feasibility

Technical feasibility examines whether the proposed system can be implemented with the available technology, tools, and resources.

- Development Tools: The game will be developed using C/C++ programming languages, which are widely used for game development due to their performance capabilities. Dev-C++ will be used as the Integrated Development Environment (IDE).
- Hardware Requirements: The game can run on standard PCs with basic hardware requirements, ensuring it is accessible to a wide audience.
- Game Design: The gameplay mechanics, such as player movement, shooting, and enemy behavior, are simple to implement using the available technical knowledge of C/C++.

4.1.2 Economic Feasibility

Economic feasibility assesses the financial aspects of the project, including the cost of development, resources, and long-term maintenance.

- Cost of Development: The project is cost-effective since it utilizes open-source tools such as Dev-C++ for development, with no additional licensing fees required for development software.
- No External Libraries: By not relying on external libraries, we minimize potential costs associated with licensing and integration.

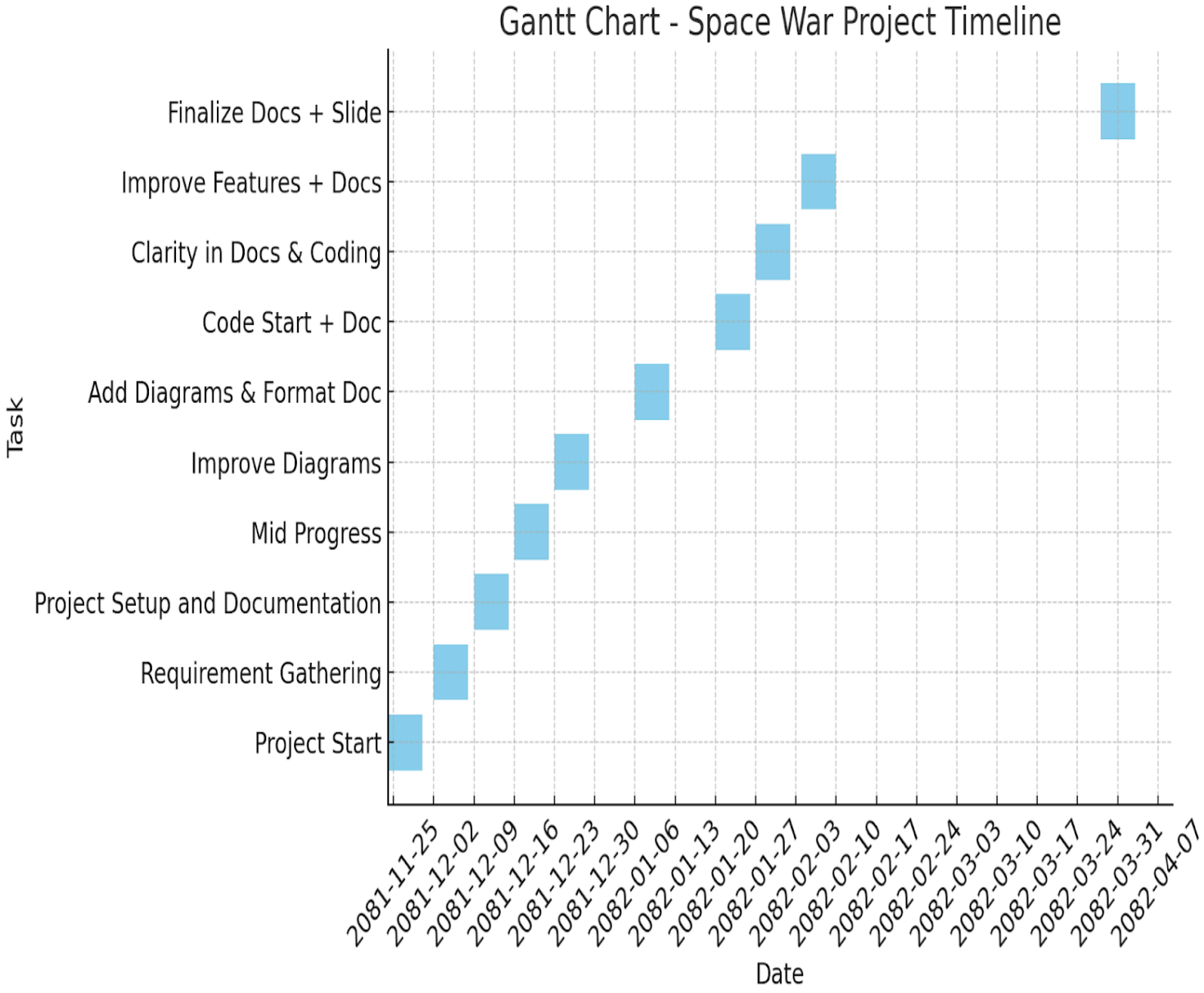
- Low Maintenance Costs: Once the game is developed, the ongoing costs for maintenance and updates will be minimal.

4.1.3 Schedule Feasibility

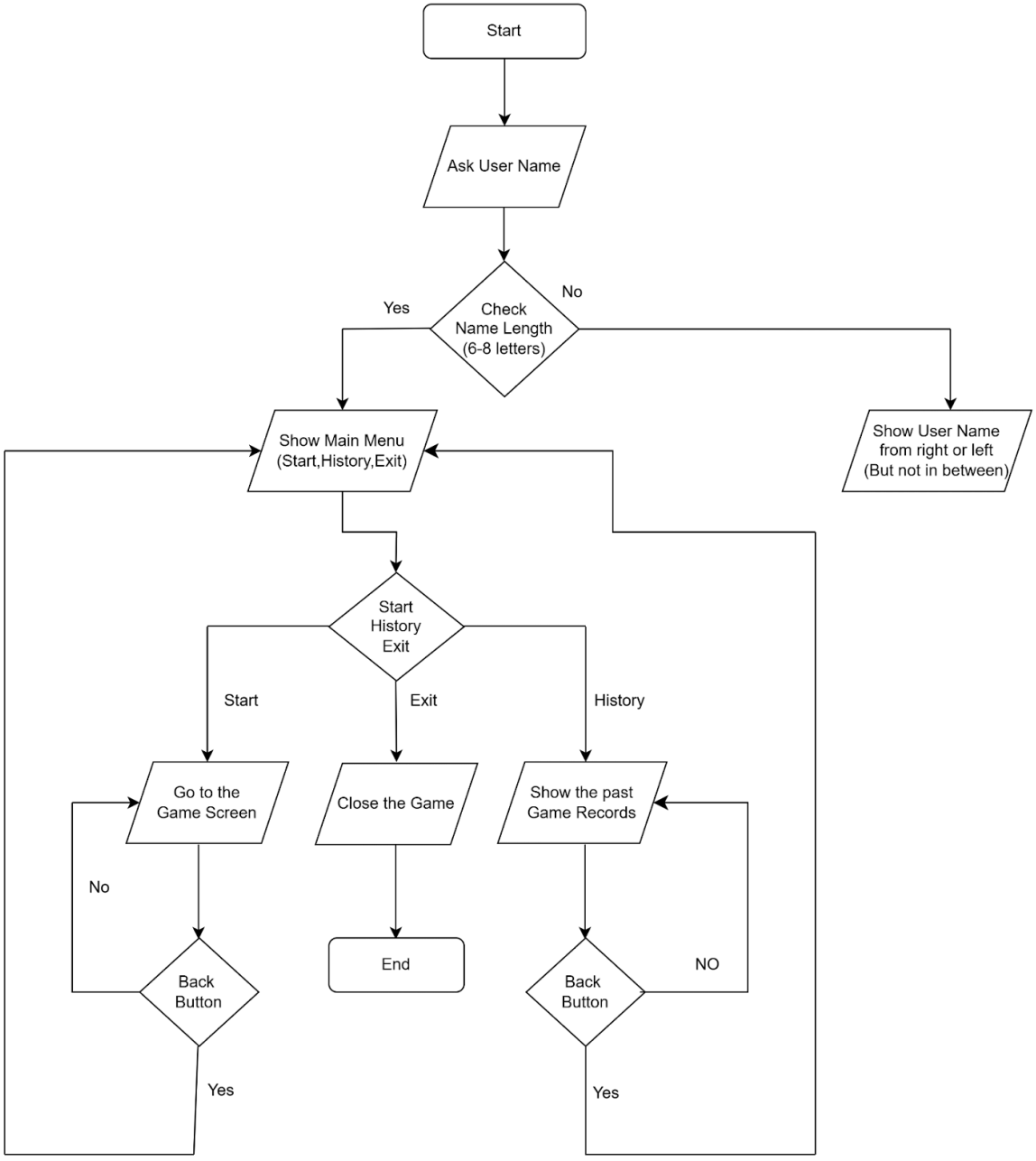
Schedule feasibility analyzes whether the project can be completed within a reasonable timeframe, based on the available resources.

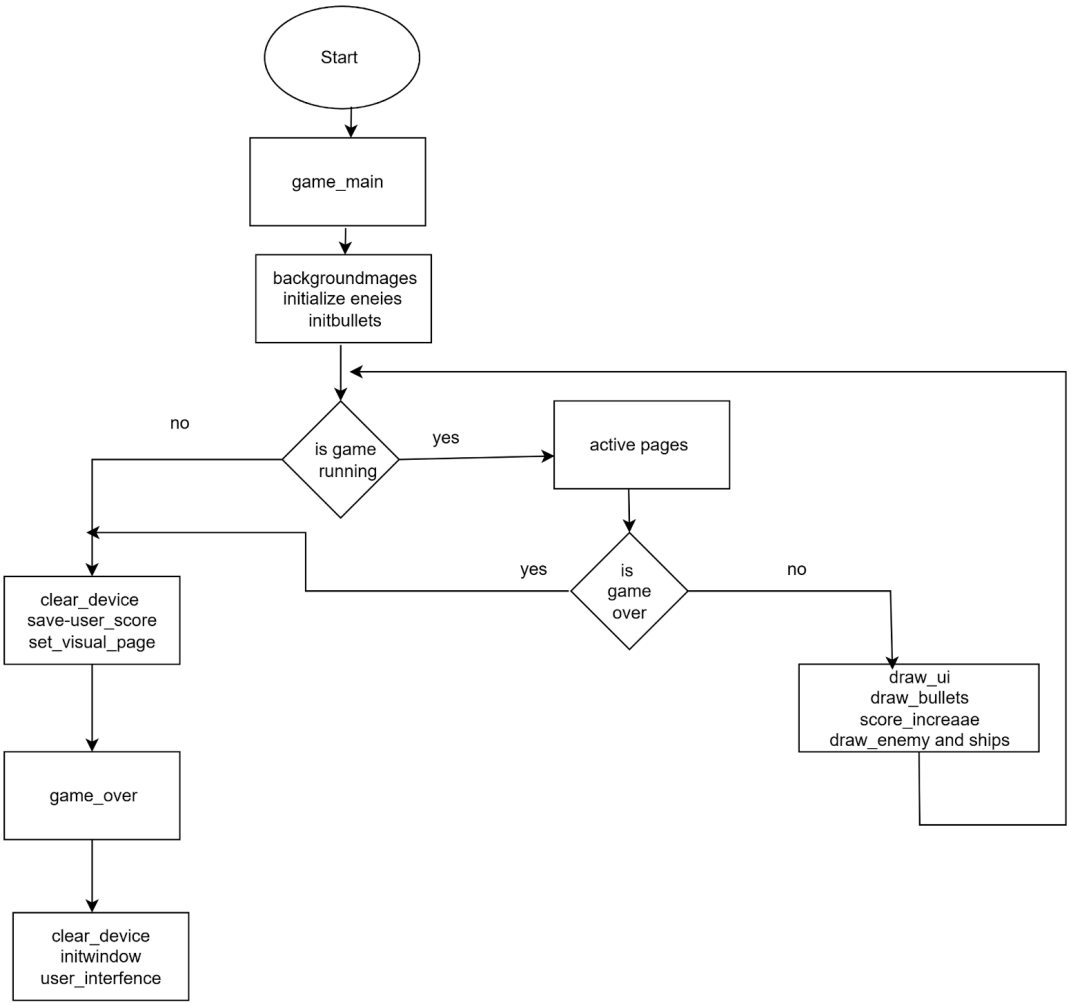
- Timeline: Given the project's scope, the development should be completed within the allotted time frame, with major milestones achieved in the following stages:
    - Initial Development: 2-3 weeks for basic gameplay mechanics (player movement, shooting).
    - Game Design and Graphics: 1-2 weeks for integrating simple graphics and sound.
    - Testing and Debugging: 1 week for testing game functionality and fixing bugs.
    - Final Polish and Documentation: 1 week for finalizing the game and preparing project documentation.
- Resource Availability: With a team of 3 members, the workload is evenly distributed and manageable within the project's duration.

4.5 Gantt Chart



5. Flowchart


Start
Ask User Name
Check
Name Length
(6-8 letters)
Yes
No
Show User Name
from right or left
(But not in between)
Show Main Menu
(Start,History,Exit)
Start
History
Exit
Start
Exit
History
Go to the
Game Screen
Close the Game
Show the past
Game Records
No
Back
Button
End
Back
Button
NO
Yes
Yes



6. System Design

6.1 System Architecture

The system architecture of the *Space War Game* is designed using a **layered and modular approach**, ensuring a clear separation of concerns across different functional components. This design supports simplicity, maintainability, and easy debugging — crucial for a console-based game developed using C and C++ without external libraries.

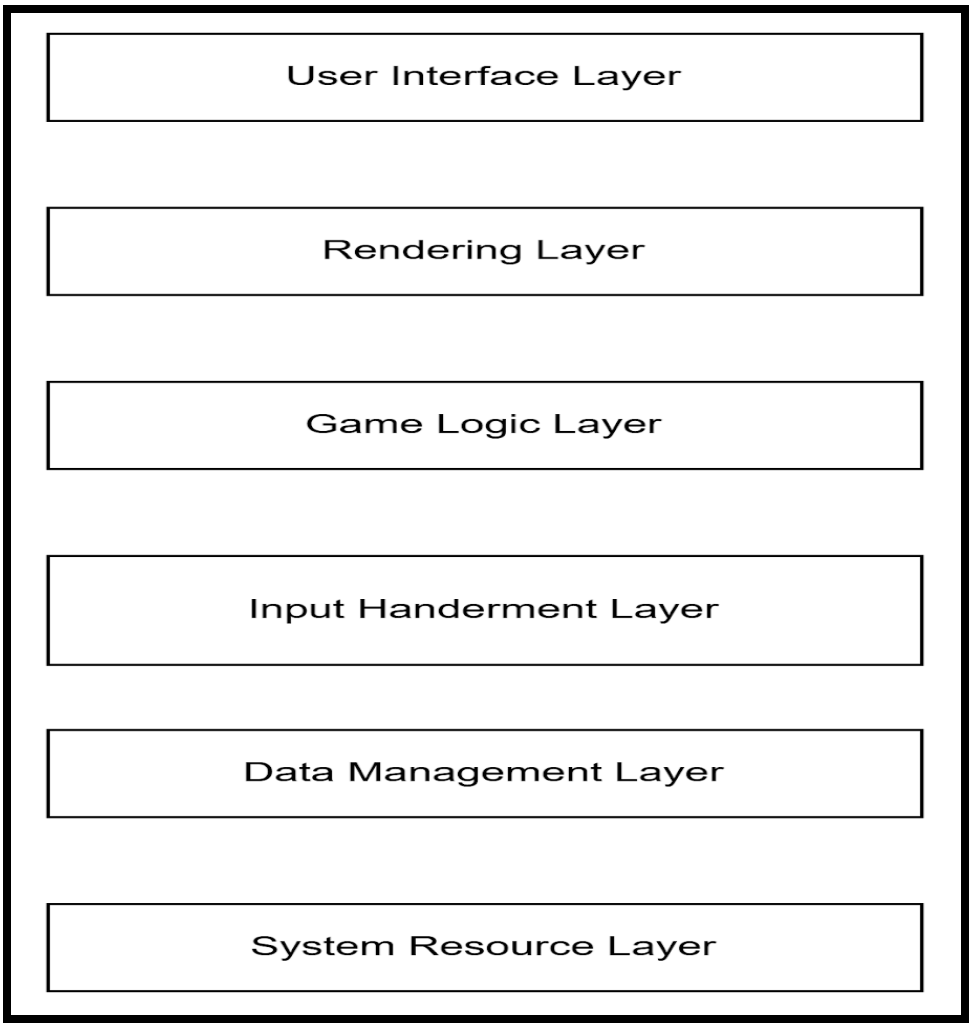


Fig 6.1: System Architecture

6.2 Procedure-Oriented

In a **Procedure-Oriented**, the focus is primarily on creating a sequence of steps or functions that will achieve the desired outcome. In the context of the Space War Game, the design is structured around a series of functions that perform specific tasks, such as moving the spaceship, detecting collisions, handling enemy behavior, and managing user input.

6.2.1 Data Flow Diagram

A **Data Flow Diagram (DFD)** represents the flow of data within the *Space War Game* system. It shows how input is transformed into output through various processes, data stores, and external entities. The DFD focuses on how the data moves rather than how it is processed procedurally.

6.2.2 Context Level(Level 0)DFD

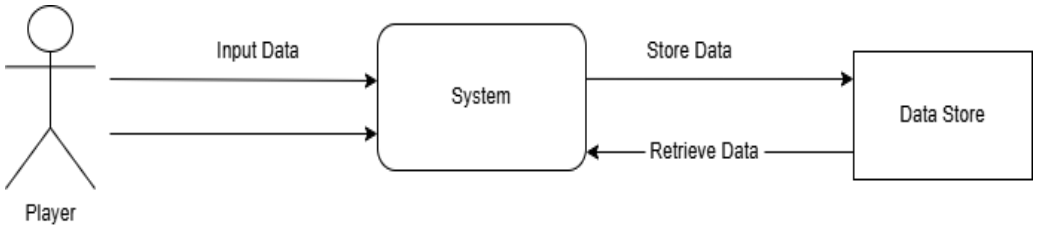


Fig 6.2.2 Context Level(Level 0)DFD

6.2.3 Context Level(Level 1)DFD

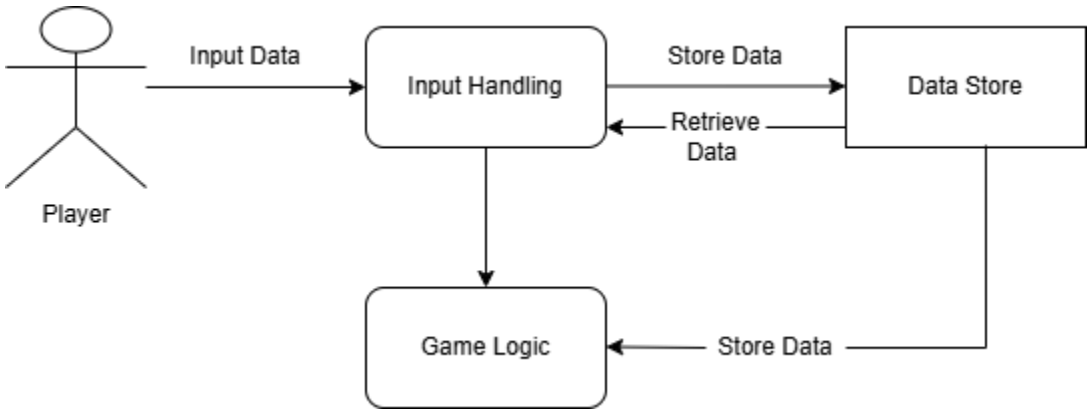


Fig 6.2.3 Context Level(Level 1)DFD

6.2.4 Context Level(Level 2)DFD

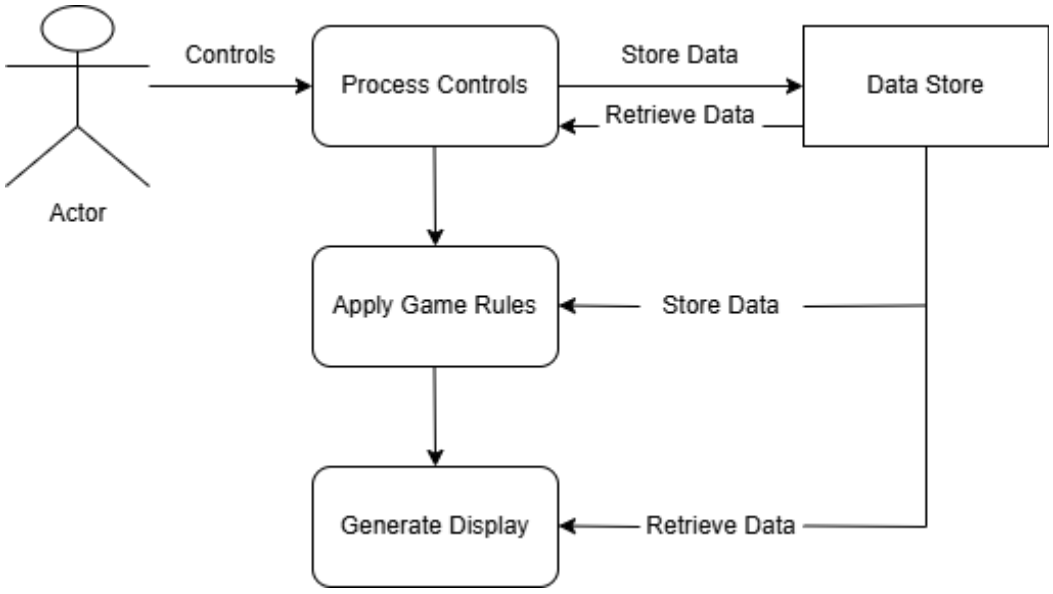


Fig 6.2.4 Context Level(Level 2)DFD

6.2.5 Use Case Diagram

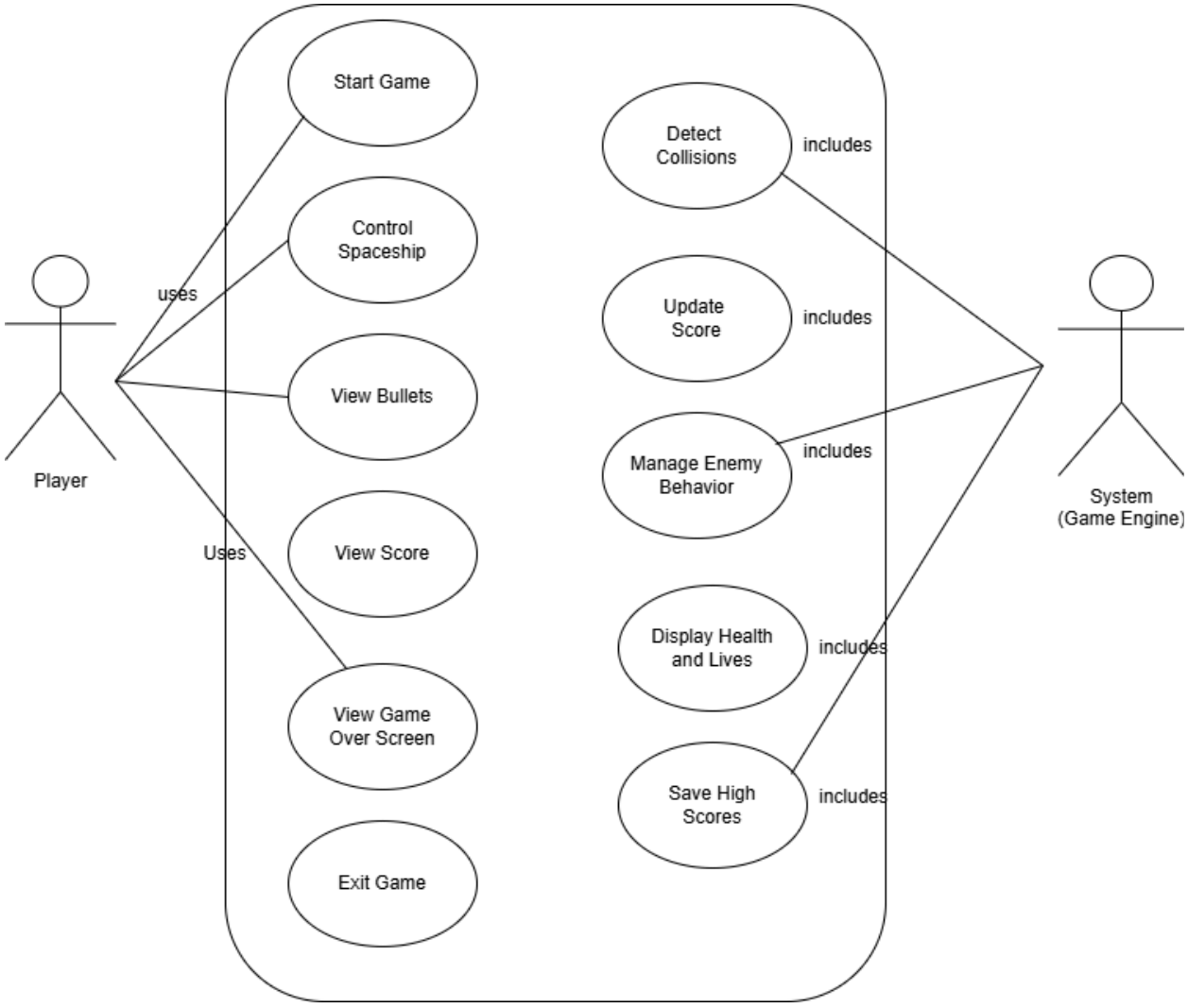


Fig 6.2.5 Use Case Diagram

6.3 Object Oriented

The **Space War Game** has been developed using an **Object-Oriented Programming (OOP)** approach in C++, allowing for better organization, modularity, and reusability of code. The design centers around key OOP principles such as encapsulation, inheritance, abstraction, and polymorphism.

This approach is suitable for representing different in-game entities like players, enemies, bullets, and game mechanics as distinct classes with their own attributes and behaviors.

6.3.1 Class Diagram

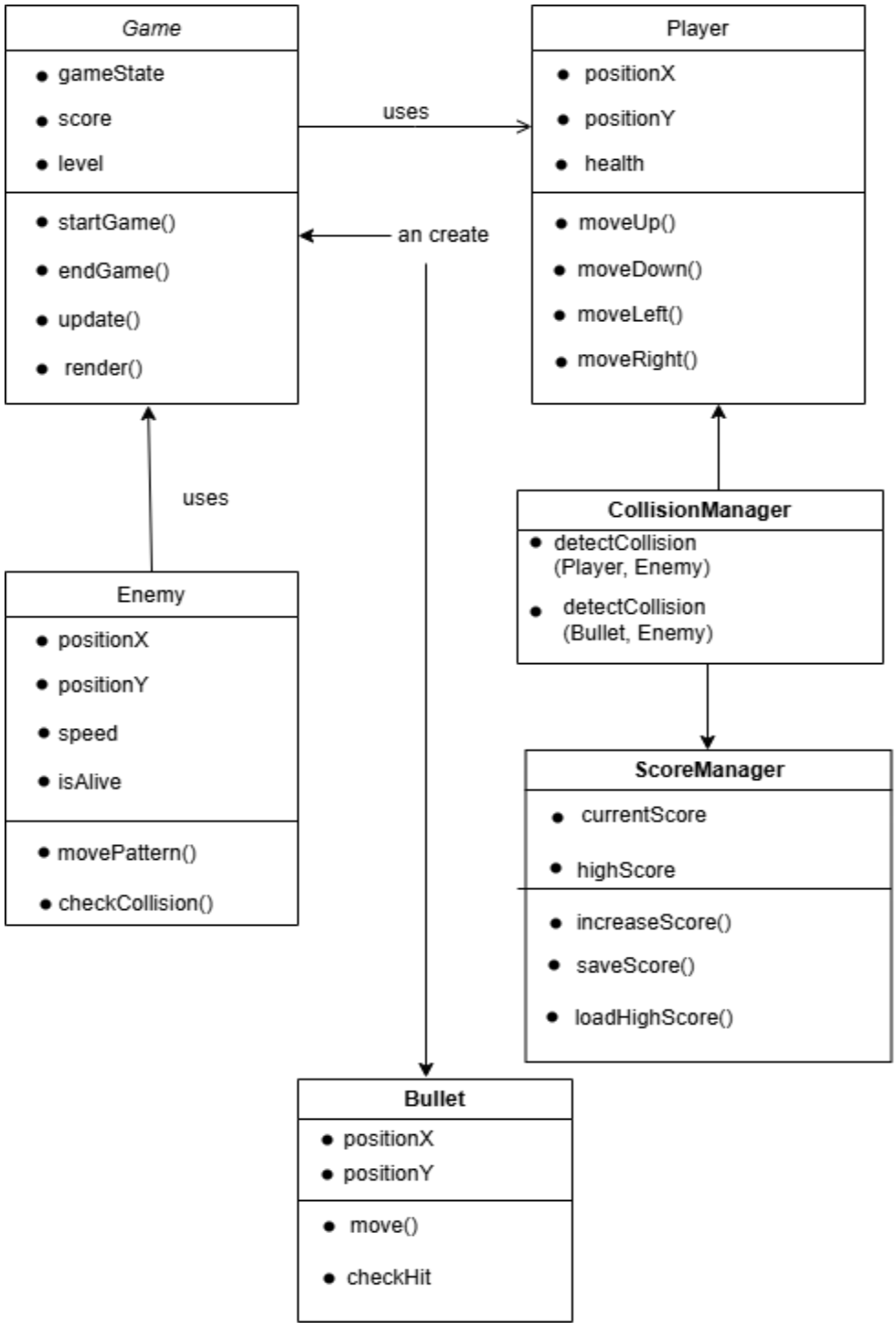


Fig 6.3.1 Class Diagram

6.3.2 Component Diagram

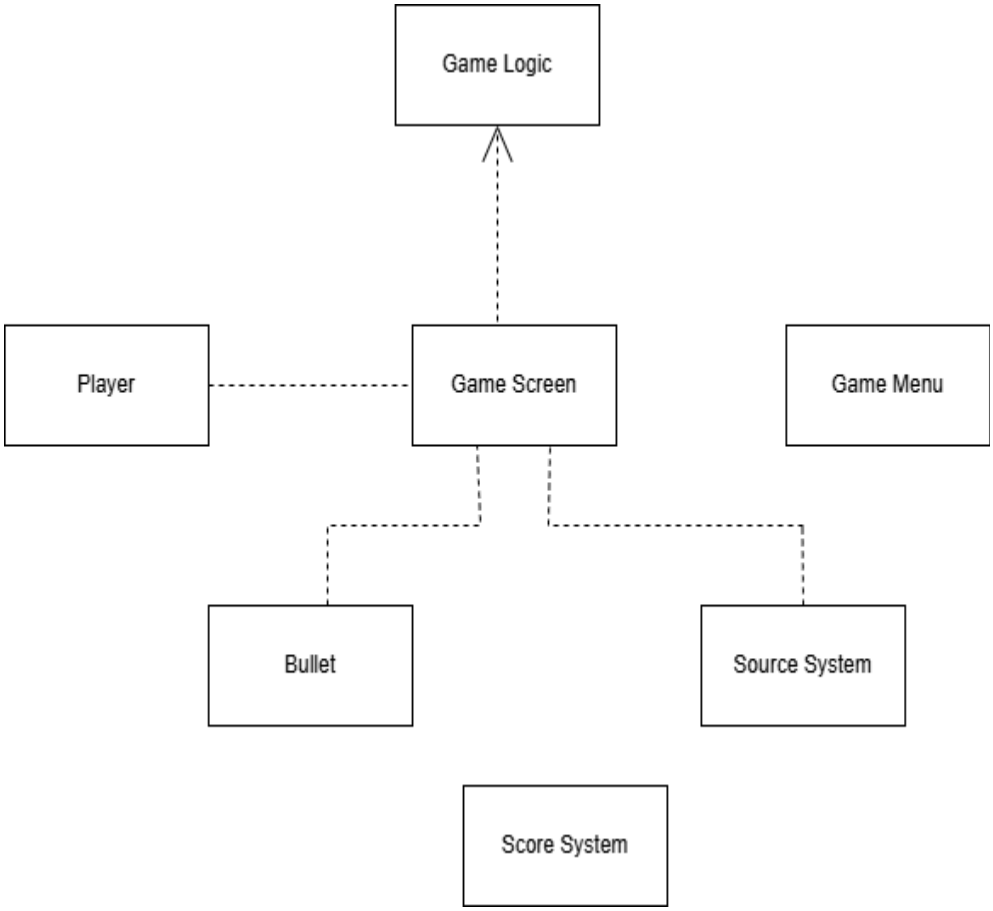


Fig 6.3.2 Component Diagram

6.3.3 Sequence Diagram

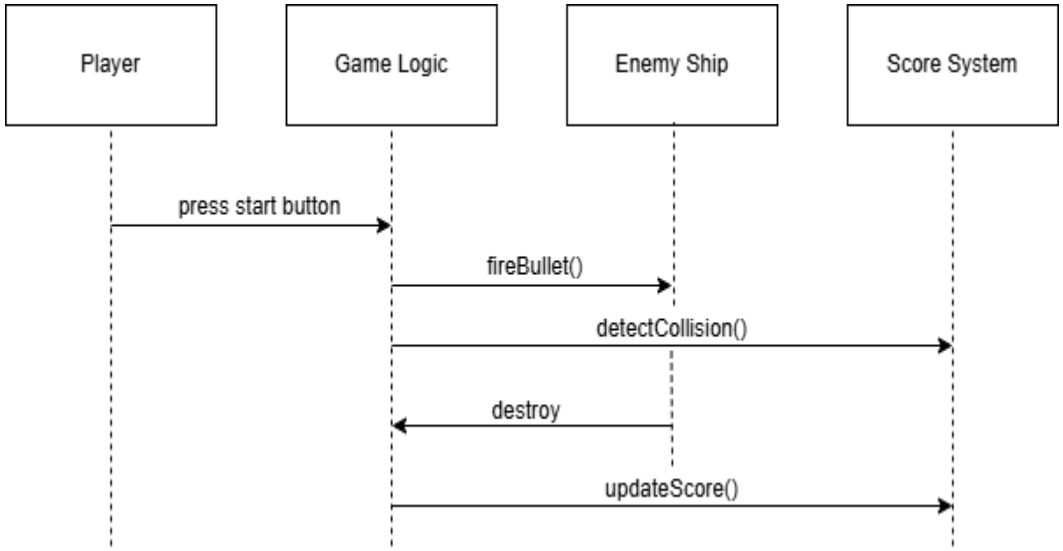


Fig 6.3.3 Sequence Diagram

6.3.4 Communication Diagram

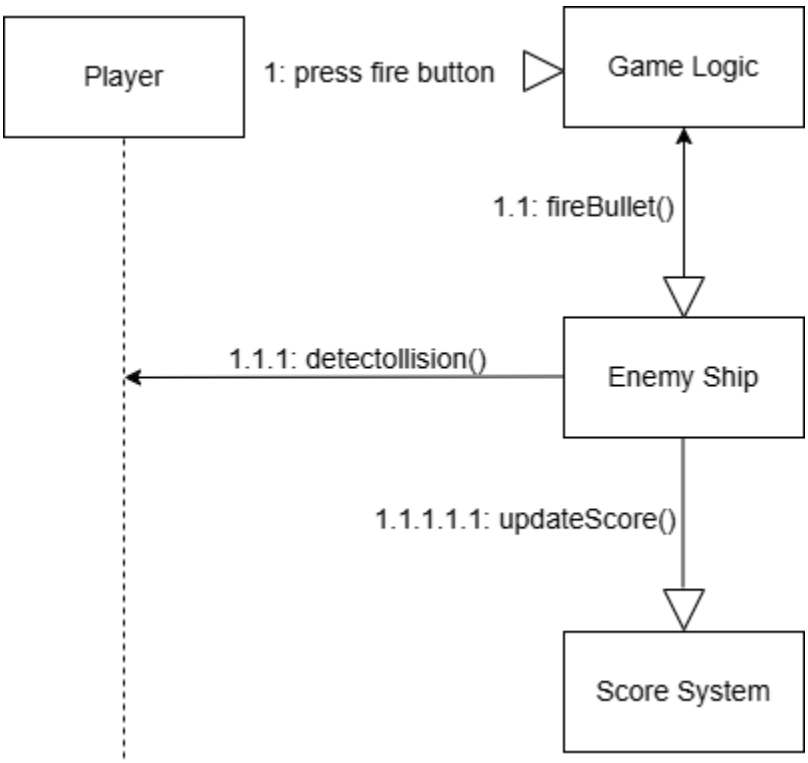


Fig 6.3.4 Communication Diagram

7. Testing and Debugging

Testing and debugging are critical components of the Space War Game project to ensure the system runs smoothly and meets all functional and non-functional requirements. The purpose of this phase is to identify and eliminate bugs, enhance performance, and guarantee a seamless gameplay experience for the user.

7.1 Testing Strategy

Testing in this project was performed in a structured and multi-layered approach. The major types of testing involved include:

1. Unit Testing: Each module such as player movement, bullet firing, collision detection, and enemy behavior was tested individually to verify their expected output.

2. Integration Testing: After unit testing, individual components were integrated, and the complete flow of the game (movement → shooting → enemy collision → scoring) was tested to identify issues arising from inter-module interaction.

3. System Testing: System-wide testing was conducted to ensure the entire game operates correctly under expected and edge-case conditions. The testing validated aspects like:

- Smooth rendering of gameplay elements
- Accurate collision detection
- Score tracking and display
- Game over condition when health reaches zero

4. Regression Testing: Each time new features were added (e.g., score tracking or restart mechanism), prior functionalities were retested to ensure no existing behavior broke.

5. Performance Testing: To ensure a smooth user experience, the frame rate was monitored, targeting a minimum of 30 FPS. Gameplay responsiveness was tested across different systems with minimal hardware configurations.

7.2 Debugging Process

Debugging was an ongoing process during the entire development lifecycle. Common debugging techniques used:

- **Print Statements:** Used to trace execution flow and variable values at runtime (e.g., to check bullet positions or player health).
- **Step-by-Step Execution:** Utilizing Dev-C++'s inbuilt debugger to go through the code line-by-line.
- **Breakpoint Analysis:** Setting breakpoints in key areas like collision detection logic to identify the exact location of unexpected behavior.
- **Edge Case Simulation:** For example, simulating simultaneous key presses, or extreme user inputs to test stability.

7.3 Common Bugs Encountered and Resolved

Bug Description	Cause	Solution
Bullet passes through enemy	Collision box mismatch	Standardized bounding box logic
Score not updating after enemy hit	Missed function call on collision	Updated scoring logic in loop
Player movement lag	Inefficient input polling	Optimized input handling code
Game over not triggered properly	Health not updating on collision	Fixed damage detection method

Game restarts with old score	Score variable not reset	Added resetScore() on restart

7.4 Final Validation Checklist

This rigorous testing and debugging phase ensured the Space War Game is not only functional but also provides a smooth and reliable user experience. It plays a vital role in achieving the educational objective of reinforcing core programming and game development skills.