INTERIM REPORT

PLATFORMER GAME

*Dissertation submitted in fulfilment of the requirements for the Degree of*

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE AND ENGINEERING**

**– DATA SCIENCE WITH MACHINE LEARNING**

By

**NIHAL NAZEER**

**Registration No: 12311441  
Section: K23CH**

**Roll No: 11  
15/11/2024**

Supervisor

**Aman Kumar**



**School of Computer Science and Engineering**

Lovely Professional University Phagwara, Punjab (India) October 2024

# ACKNOWLEDGEMENT

I at this moment declare that the research work reported in the dissertation/dissertation proposal entitled “PLATFORMER GAME” in partial fulfillment of the requirement for the award of Degree for Master of Technology in Computer Science and Engineering at Lovely Professional University, Phagwara, Punjab is an authentic work carried out under the supervision of my research supervisor Mr. Aman Kumar. I have not submitted this work elsewhere for any degree or diploma.

I understand that the work presented here with directly complies with Lovely Professional University’s Policy on plagiarism, intellectual property rights, and the highest standards of moral and ethical conduct. Therefore, to the best of my knowledge, the content of this dissertation represents an authentic and honest research effort conducted, in its entirety, by me. I am fully responsible for the contents of my dissertation work.

NIHAL NAZEER  
REG.NO - 12311441

\*\*Table of Contents\*\*

1. Introduction
   * 1.1 Project Overview
   * 1.2 Purpose and Significance
   * 1.3 Registration Details
2. Objectives and Scope of the Project

* 2.1 Project Objectives
* 2.2 Project Scope

1. Application Tools

* 3.1 Software Applications
* 3.2 Programming Languages

1. Project Structure of the Student Management System (SMS)

* 4.1 Main Components of the Project
* 4.2 Classes and Their Functions
* 4.3. Interaction Between Components

1. Flow chart
   * 5.1 Explanation of the Flowchart

**I. INTRODUCTION**

**1.1 Project Overview**

The Platformer Game is a Python-based interactive application that leverages the robust capabilities of the Pygame library to create an engaging gaming experience. This project challenges players to navigate through meticulously designed levels, where they must overcome obstacles, collect rewards, and achieve objectives by skillfully maneuvering a character. Designed to demonstrate the practical application of Python in game development, the project is a testament to the simplicity and power of programming in creating interactive experiences.

The game revolves around controlling a responsive character that moves smoothly across a vibrant 2D environment. Key features include dynamic jumping mechanics, accurate collision detection, and a comprehensive scoring system to track player progress. Each level is tailored to present unique challenges, gradually increasing in complexity to test the player’s problem-solving abilities and reflexes.

By focusing on efficient code structure and user-friendly design, the Platformer Game serves as a learning tool and a source of entertainment. Its modular design allows future enhancements, making it a scalable and adaptable platform for budding game developers and enthusiasts.

**1.2 Purpose and Significance**

The Platformer Game was developed with the dual purpose of education and entertainment. At its core, the project seeks to highlight Python's versatility in application development, particularly in the gaming domain. By utilizing the Pygame library, the game introduces essential programming concepts like graphical rendering, event-driven logic, and basic physics simulation in an accessible manner.

The game’s educational value is immense, serving as a foundation for beginners to understand programming basics and for advanced learners to experiment with creative mechanics. Its significance extends beyond learning, as it also provides a practical demonstration of optimizing performance, designing intuitive interfaces, and handling user interactions seamlessly.

Scalability is a key aspect of the project. Developers can expand its capabilities by introducing features such as multiplayer gameplay, enhanced visuals, or more complex levels. This adaptability underscores its importance as a resource for fostering creativity and innovation within the gaming community.

**1.3 Gameplay Details**

The gameplay of the Platformer Game focuses on a blend of strategy, agility, and precision. The following elements define the core experience:

Player Character:

* A 2D avatar controlled by the player, capable of running, jumping, and interacting with objects in the environment.
* Includes smooth animations for states such as idle, movement, and jump, enhancing the visual appeal and responsiveness.

Platforms and Obstacles:

* Platforms of varying sizes and types, including static and moving ones, create dynamic gameplay.
* Hazards like spikes, pits, and moving barriers test players’ reflexes and decision-making skills.

Scoring System:

* Players accumulate points by collecting rewards such as coins or stars scattered across levels.
* A live scoreboard displays progress, while a high-score feature motivates continued engagement.

Level Design:

* A series of levels designed to gradually increase in difficulty, each with a unique layout.
* Challenges include timed jumps, avoiding obstacles, and reaching the endpoint within specific constraints.

Audio and Graphics:

* 2D graphics with vibrant colors create an engaging aesthetic.
* Background music and sound effects enhance the immersive experience, providing feedback for player actions.

**II. OBJECTIVES AND SCOPE OF THE PROJECT**

**2.1 Project Objectives**

The primary objectives of the Platformer Game are as follows:

Developing Core Mechanics:

1. Implement intuitive controls for fluid character navigation and interaction.
2. Design responsive jumping mechanics with realistic physics to ensure precision.
3. Integrate robust collision detection for accurate interactions with platforms, obstacles, and rewards.

Creating an Engaging User Experience:

1. Develop levels that balance challenge and entertainment to maintain player interest.
2. Optimize game performance to ensure a seamless experience across devices.
3. Incorporate visual and audio feedback to enhance player immersion.

Rewarding Player Progress:

1. Introduce a scoring system that tracks player achievements and displays them dynamically.
2. Motivate players with a leaderboard or similar recognition feature.

Scalable Design for Future Enhancements:

1. Establish a modular structure that facilitates adding new features, such as multiplayer modes or advanced levels.
2. Design an extensible framework for incorporating more sophisticated game mechanics in subsequent iterations.

**2.2 Project Scope**

The scope of this project includes:

Gameplay Elements:

* A single-player experience focusing on platforming mechanics and level completion.
* Diverse levels with unique layouts and escalating challenges.

Visual and Audio Design:

* Simple but effective 2D graphics to depict the game world.
* Audio elements, including sound effects and background music, to enrich the gameplay.

Limitations:

* Multiplayer functionality is not included in this version.
* Advanced features like 3D graphics and persistent save systems are excluded to prioritize simplicity.

**III. APPLICATION TOOLS**

**3.1 Software Applications**

Several tools were utilized to develop the Platformer Game, ensuring an efficient workflow:

Python:

* The core programming language for this project. Its simplicity, extensive libraries, and active community support made it an ideal choice.

Pygame Library:

* Used for graphical rendering, audio integration, and event handling, enabling smooth game mechanics.

Code Editor (e.g., Visual Studio Code):

* Provided an integrated development environment for writing and debugging the code effectively.

Version Control Tools (Git, GitHub):

* Facilitated collaborative development and tracked changes systematically.

Graphics Tools (e.g., GIMP):

* Allowed the creation and editing of visual assets like characters and backgrounds.

**3.2 Programming Languages**

Python is the primary language used for this project, with Pygame offering specialized functionalities for game development.

**IV. PROJECT STRUCTURE OF THE PLATFORMER GAME**

**4.1 Main Components**

The project comprises the following key components:

Player Character:

* Represents the player’s in-game avatar, capable of movement and interaction.

Environment:

* Consists of platforms, obstacles, and rewards that define the game’s challenges.

Game Logic:

* Handles core gameplay functions, including scoring, level progression, and game state management.

Graphics and Sound:

* Visual and auditory feedback enhance the gaming experience, providing cues and maintaining immersion.

**4.2 Classes**

Player Class:

* Attributes: Position, velocity, and animation states.
* Methods: Manage movement, collision detection, and interaction with the environment.

Platform Class:

* Attributes: Size, position, and movement type (static or dynamic).
* Methods: Define how platforms interact with the player and environment.

Obstacle Class:

* Attributes: Position and hazard type (e.g., spikes, pits).
* Methods: Handle interactions that penalize the player for errors.

Game Class:

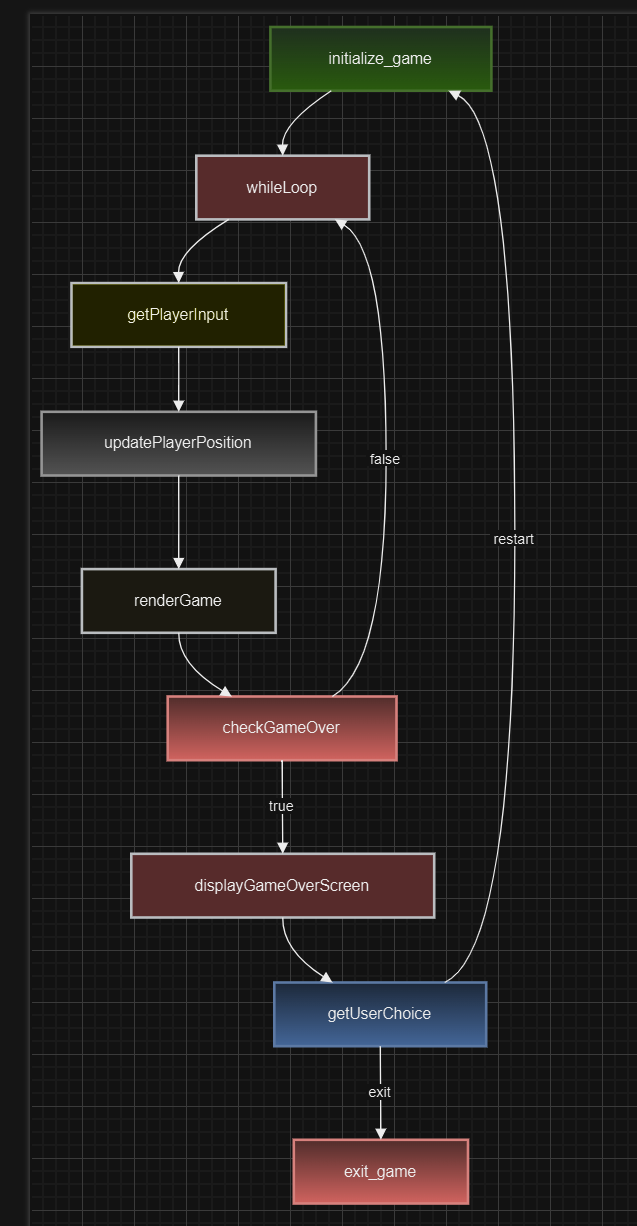
* Attributes: Score, current level, and game state.
* Methods: Manage the game loop, transitions, and feedback mechanisms.

4.3 Interaction Between Components

* Player Movement and Interaction: Players control the character via keyboard inputs, navigating platforms and avoiding hazards.
* Level Progression: Successfully completing levels unlocks new challenges.
* Scoring and Feedback: Dynamic score updates and visual/audio cues enhance engagement.

# V.Flowchart or Algorithm of the Project

Below is a simplified flowchart that illustrates the process flow of the Platformer Game



* 1. **Explanation of the Flowchart**

1. Initialize Game:

Sets up the initial state of the game, including loading game assets (images, sounds, etc.) and creating game objects (player, enemies, platforms, etc.).

2. While Loop:

The main game loop, which continuously runs until the game ends.

3. Get Player Input:

Reads input from the player, such as keyboard presses or joystick movements.

4. Update Player Position:

Updates the player's position based on the input received.

5. Render Game:

Draws the game world to the screen, including the player, enemies, platforms, and any other visual elements.

6. Check Game Over:

Determines if the game is over, such as if the player loses all their lives or completes the game.

7. Display Game Over Screen:

If the game is over, displays a game over screen with relevant information like the final score.

8. Get User Choice:

Prompts the user to choose whether to restart the game or exit.

9. Exit Game:

Exits the game, cleaning up any resources used.

10. Restart:

Restarts the game by going back to the "Initialize Game" step.

This flowchart outlines the basic structure of a platformer game, focusing on the core game loop and its essential components.