**PLATFORMER GAME**

* **A Python-based Game where players Jumping through platforms and Overcome obstacles, etc.**

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

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE AND ENGINEERING**

By

**NIHAL NAZEER**

**12311441**

Supervisor

**AMAN KUMAR**



**School of Computer Science and Engineering**

Lovely Professional University

Phagwara, Punjab (India)

November,2024

@ Copyright LOVELY PROFESSIONAL UNIVERSITY, Punjab (INDIA)

November,2024

ALL RIGHTS RESERVED

**DECLARATION STATEMENT**

I I hereby declare that the research work reported in the dissertation/dissertation proposal entitled “PLATFORMER GAME” in partial fulfilment of the requirement for the award of a Degree for Bachelor 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 haven’t submitted this work elsewhere for any degree or diploma.

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

Signature of Candidate

NIHAL NAZEER

Roll. No: 11

Regd. No: 12311441

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| S. No | Contents | Page No. |
| **1.** | **Inner first page – Same as cover** | **01** |
| **2.** | **Declaration by the Scholar** | **02** |
| **3.** | **Table of Contents** | **04** |
| **4.** | **Introduction** | **04** |
| **5.** | **Objectives And Scope Of The Project** | **04** |
| **6.** | **Application Tools** | **06** |
| **7.** | **Project Design** | **06** |
| **8.** | **Flow Chart** | **07** |
| **9.** | **Project Implementations** | **09** |
| **10.** | **Screenshots** | **12** |
| **11.** | **Testing And Validation** | **17** |
| **12.** | **Summary** | **19** |
| **13.** | **Bibliography** | **20** |

**I. INTRODUCTION**

The Platformer Game is a 2D video game designed where players navigate jump between platforms, overcoming obstacles to advance through stages. The project is developed using Python with the Pygame library, offering a simple yet engaging gameplay experience. The game is built around a set of challenges where players control a character that must avoid obstacles and complete levels in a series of increasingly complex stages.

The game environment is designed to be dynamic. Pygame serves as the core tool to render graphics, handle user input, and implement game logic. This project focuses on creating a smooth user experience by leveraging the interactive features provided by the library, such as keyboard input for movement and real-time physics for gravity and collisions.

The platformer game includes essential features such as character movement, jumping, scoring, etc. As players move forward, the difficulty increases, keeping the game engaging and challenging. Although basic in nature, the game provides a foundation for future expansion, more complex mechanics, and additional features to be added over time.

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

**OBJECTIVES:**

1. Game Development: Create a playable platformer game with interactive levels and challenges.
2. Character Control: Implement smooth and responsive character movement, including running and jumping.
3. Physics and Collisions: Incorporate realistic physics, such as gravity, and ensure accurate collision detection.
4. Score System: Implement a system to track player progress and score based on items collected or enemies defeated.
5. Visual Effects: Visual elements to enhance the gaming experience.

**SCOPE:**

The platformer game is primarily designed for a single player, focusing on character movement, obstacle avoidance, etc. While the core mechanics center around platforming and simple physics, the game is structured to support scalability. Future enhancements can include adding new power-ups, enemies, and multiplayer capabilities.

The scope includes:

* 2D platformer game environment
* Player movement and jumping mechanics
* Platforms, obstacles, and collectibles
* Score tracking system
* Visual effects

Future Scope:

* Advanced Game Mechanics: Adding power-ups, new abilities, and more complex enemies.
* Multiplayer Support: Implementing a cooperative or competitive multiplayer mode.
* Mobile Version: Adapting the game for mobile platforms for wider accessibility.
* AI Integration: Introducing AI-controlled enemies with dynamic behavior.

**III. APPLICATION TOOLS**

1. Programming Language: Python
2. IDE: Python IDLE, PyCharm
3. Libraries/Packages:
   * Pygame: Primary library for game development, handling graphics, sound, and game logic.
   * Tkinter: Optional for creating a simple menu or interface.
   * Pillow: For loading and displaying images (characters, backgrounds, etc.).
   * PyInstaller: For packaging the game into a standalone executable.
4. Additional Tools:
   * Tiled Map Editor: For designing and exporting game levels (optional).
   * GIMP or Photoshop: For creating and editing game sprites and textures.

**IV. PROJECT DESIGN**

The design of the Platformer Game follows a modular structure that separates different functionalities into manageable components:

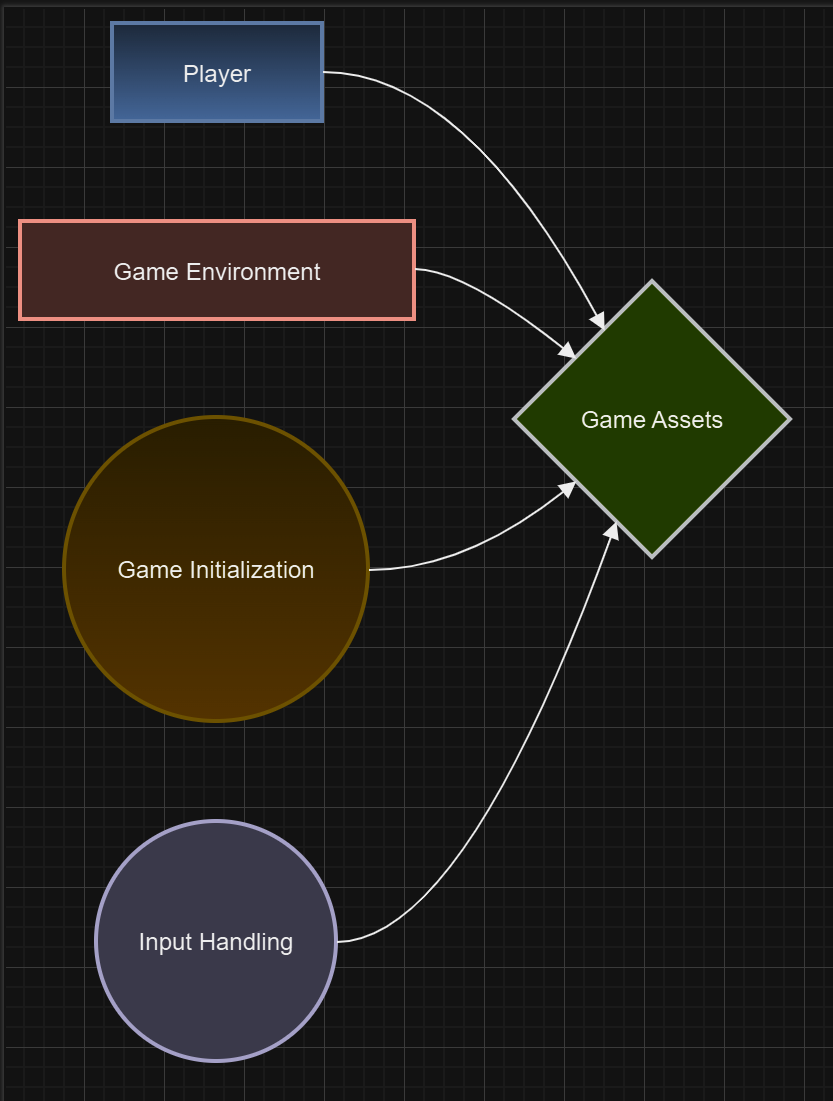
1. Game Initialization:
   * Setup the game window, initialize variables, and load resources (images, sounds).
2. Player Character:
   * Implement movement controls (left, right, jump) with smooth transitions.
   * Add collision detection to prevent falling off platforms or passing through walls.
3. Game Loop:
   * The main loop runs the game, checking for player input, updating game objects, and rendering the game state to the screen.
4. Score and Level Progression:
   * Implement a score system that tracks items collected or enemies defeated.
   * Include level progression that allows players to move to the next stage once certain conditions are met.
5. Visual Effects:
   * Background for actions like jumping, and visual feedback for collisions and item pickups.

**V. FLOWCHART**

**A flowchart would include:**

1. Game Start: Initializes the game environment and resources.
2. Player Input: Detects keyboard input for controlling the player character.
3. Game Loop: Runs continuously, handling player movements, level transitions, and collisions.
4. Level Completion: If the player reaches the goal or completes the level, they are moved to the next stage.
5. Game Over/Win: Displays the final score and either restarts the game or shows a win screen based on the player's progress.

This flowchart visually demonstrates how the game progresses from start to finish, with each component interacting and updating based on player actions.

  
  
  
A diagram of a graph

Description automatically generated with medium confidence

**VI. PROJECT IMPLEMENTATION**

The Platformer Game is a 2D game in which players control a character to navigate through various levels, jumping between platforms, avoiding obstacles, and overcoming challenges. This project is built using Python and utilizes the Pygame library for rendering graphics, handling user inputs, and managing game mechanics.

**1. Project Setup and Environment**

The game is developed using Python, a versatile and user-friendly programming language. The Pygame library is employed to manage the graphical user interface and game-related functionalities such as input handling, rendering graphics, and game logic.

* Python 3.x: The game requires Python 3 for compatibility with modern libraries and features.
* Pygame: Pygame is utilized for creating the visual aspects of the game, managing game states, and handling event-driven programming.

**2. Game Structure Overview**

The structure of the game is divided into several key components that interact with each other to create the overall gameplay experience:

1. Game Initialization: This step involves setting up the game environment, initializing game assets (such as images and sounds), and configuring game settings like screen dimensions and background.
2. Player Controls: Players control the main character to move left, right, and jump to navigate through platforms while avoiding obstacles.
3. Game Mechanics: Key mechanics include player movement, gravity, platform interaction, and basic game physics.
4. Rendering: All graphical elements, including the player character, background, platforms, and obstacles, are drawn on the screen.
5. Game State Management: This involves tracking the player's health, score, and transitioning between levels or handling game over scenarios.

**3. Game Initialization**

At the start of the game, the Pygame library is initialized, and the game window is created. The game assets (images, sounds, etc.) are loaded into memory to ensure a smooth gameplay experience.

* Game Window: The game window is initialized with specific dimensions, and the window title is set.
* Assets: Key game assets like the player character sprite, platforms, background images, and sound effects are preloaded.

**4. Player Controls**

The player character is controlled using the keyboard. The main controls are:

* Horizontal Movement: The player can move the character left or right using the arrow keys.
* Jumping: The player can make the character jump using the 'Spacebar'. The character can only jump when they are on a platform.
* Gravity: The character is affected by gravity, which pulls them downward unless they are standing on a platform.

The player’s movement and actions are continuously updated each frame, ensuring responsive gameplay.

**5. Game Mechanics**

The game mechanics are designed to make the gameplay both challenging and enjoyable:

* Gravity: The player character experiences gravity, causing them to fall when not standing on a platform.
* Jumping: The character can jump, but the jump height is controlled, and the character will fall back down due to gravity.
* Platform Collision: The game detects when the player lands on a platform and ensures the character stays on the platform while allowing vertical movement when jumping.

**6. Level Design**

The game consists of multiple levels, each with its own platform and obstacle layout. Each level increases in difficulty, offering more challenging obstacles, higher platforms, and tricky jumps.

* Platforms: Static platforms are placed at varying heights and positions in the game world. The player must jump between them to progress.
* Obstacles: Some levels include hazards like spikes or moving platforms that the player must avoid to stay alive.

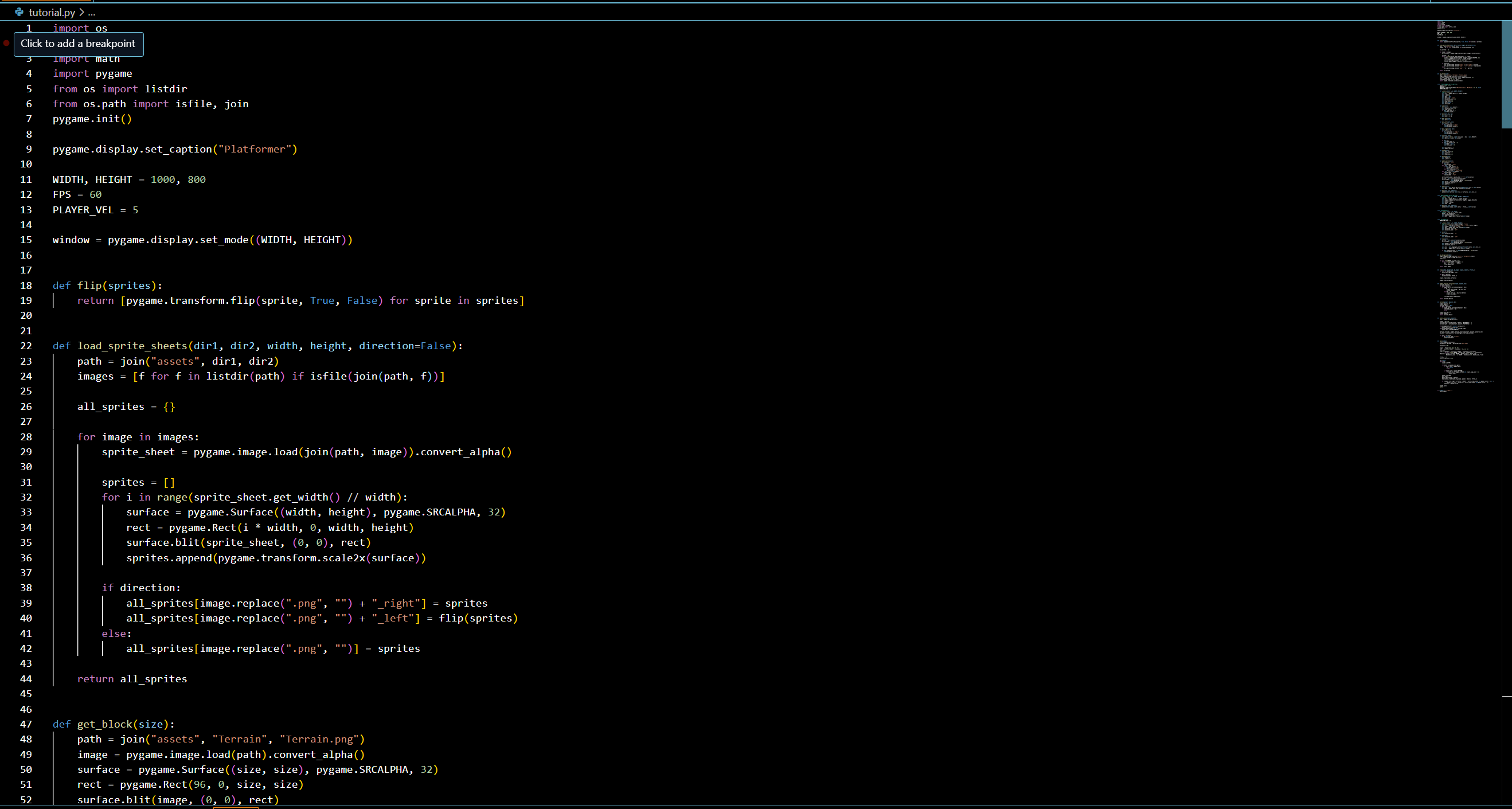
The game transitions between levels once the player successfully completes a level by reaching the end or achieving a specific objective.

**7. Rendering and Animation**

The game uses several visual elements to create an engaging experience:

* Sprites: The player, platforms, and obstacles are represented as images, called sprites, that are drawn on the screen during each frame.
* Background: The game features static or scrolling backgrounds that provide context for each level.
* Animations: The player character may have animations for different actions, like running or jumping, which enhances the game's visual appeal.

**Screenshots of code:**

****

**A black screen with white text

Description automatically generated**

**A black screen with white text

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

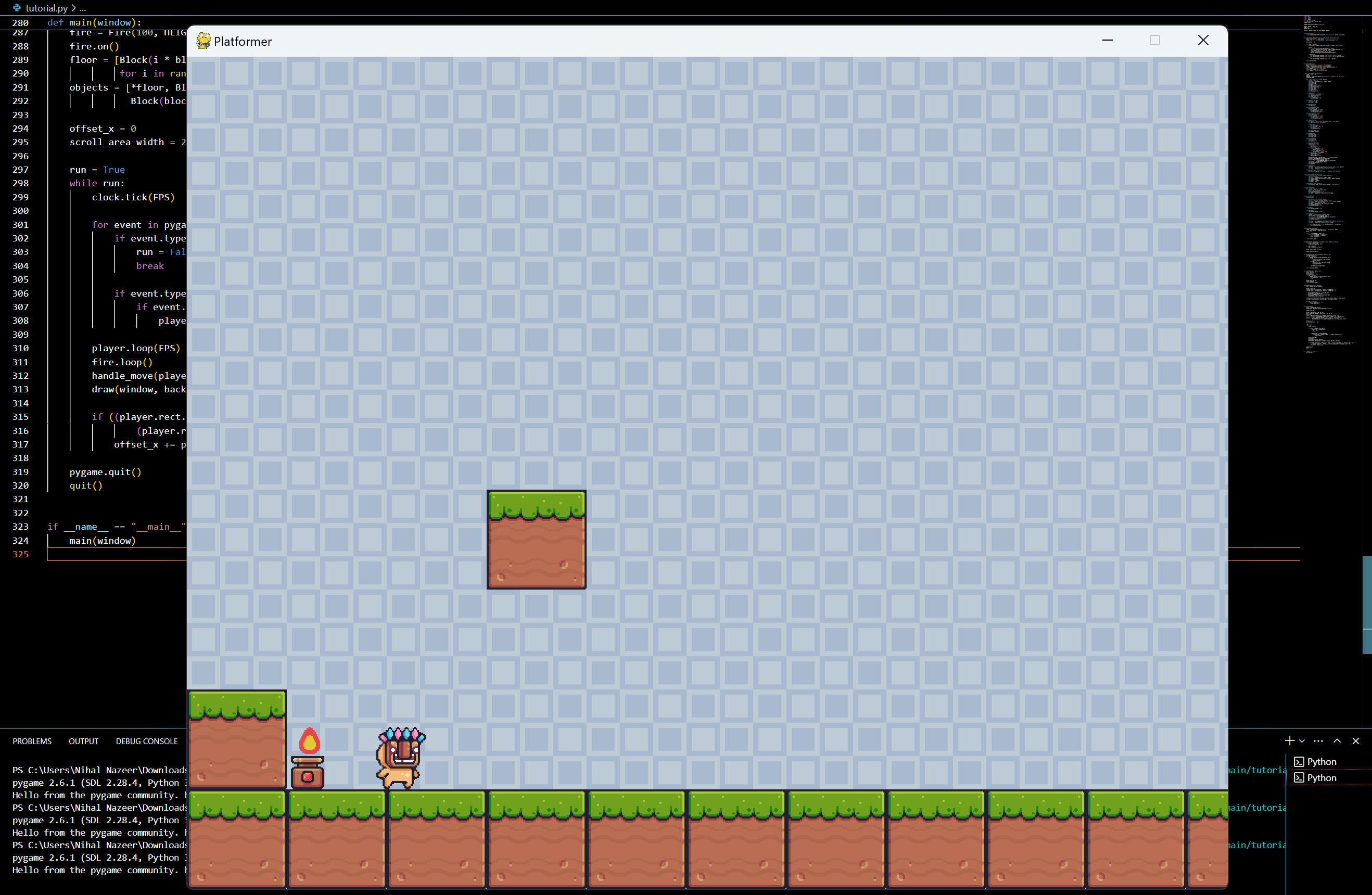
**A black background with colorful dots

Description automatically generated**

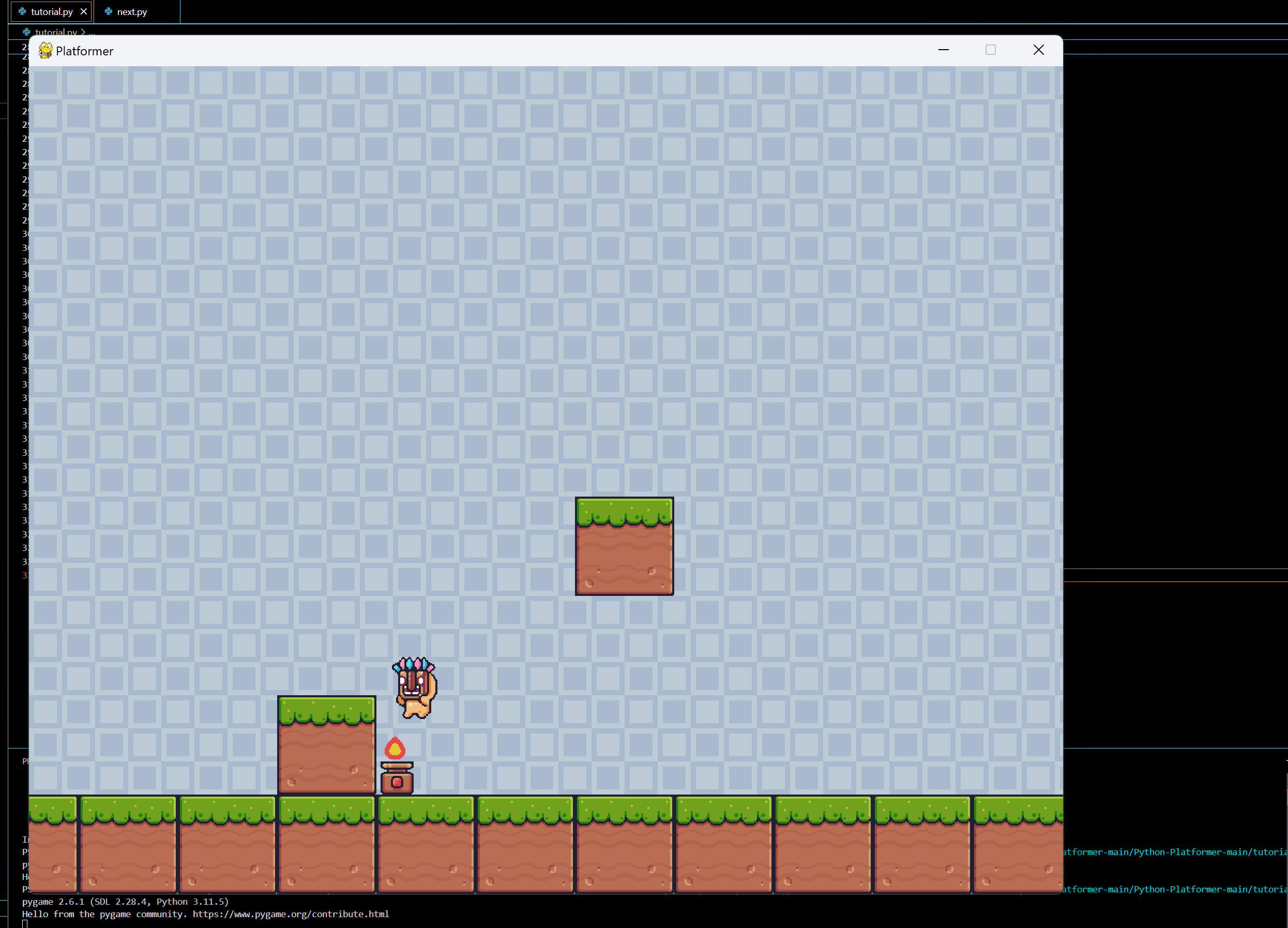
**A screenshot of a computer

Description automatically generated**

**Output:**

****





These are all the screenshots of execution and code. Important module screenshots are only added.

**VII. TESTING AND VALIDATION**

The Platformer Game project underwent thorough testing to ensure its functionality, performance, and user experience. Various testing methods were used to validate the game’s mechanics, interactions, and overall performance.

Below is a detailed account of the testing process:

| **Test Case ID** | **Test Description** | **Input** | **Expected Output** | **Actual Output** | **Status** |
| --- | --- | --- | --- | --- | --- |
| T01 | Validate player movement | Press left, right, or spacebar for jump | Character moves left/right or jumps correctly | Character moved as expected | Passed |
| T02 | Validate collision with platforms | Move player to platform and jump | Player lands or interacts with platform | Platform collision detected | Passed |
| T03 | Test gravity behavior | Let the player fall from a height | Character falls according to gravity | Gravity applied correctly | Passed |
| T04 | Test jumping height | Press spacebar to jump | Character jumps to a fixed height | Jump height correct | Passed |
| T05 | Validate collision with obstacles | Move player into spikes or obstacles | Player loses health or dies | Collision detection with obstacles | Passed |
| T06 | Test score increment | Complete a level or collect items | Score increases | Score updated correctly | Passed |
| T07 | Check level progression | Complete a level | Transition to next level | Level change executed correctly | Passed |
| T08 | Test player death | Player falls off screen or hits an obstacle | Game over screen displayed | Game over condition triggered | Passed |
| T09 | Test game over screen functionality | Trigger game over condition | Game over screen appears with restart option | Game over screen displayed successfully | Passed |
| T10 | Validate level design and layout | Play through the level | Platforms and obstacles appear correctly | Level layout and design accurate | Passed |
| T11 | Check movement smoothness | Move player continuously | Smooth transition without stuttering | Movement smooth, no lags | Passed |
| T12 | Test jumping and landing accuracy | Jump and land on various platforms | Player lands without clipping through platforms | Landing behavior accurate | Passed |
| T13 | Test performance under stress (e.g., large levels) | Play on larger levels or complex maps | Game runs without performance issues | Game performs without lag | Passed |
| T14 | Test for potential bugs or glitches | Play through the game without interruptions | No unexpected crashes or bugs | Game stable, no crashes | Passed |

All the above test cases were executed during the development of the Platformer Game. The tests confirm that the game functions as expected, with accurate player movement, collision detection, level progression, and more. Each functionality has been validated to ensure a smooth and enjoyable gaming experience for the player.

**VIII. Summary**

The Platformer Game has been designed to provide an engaging and enjoyable experience for players, focusing on smooth gameplay mechanics and level progression. By integrating simple yet effective movement controls, players can easily navigate through levels, jump over obstacles, and interact with the game environment. These gameplay mechanics contribute to a seamless experience that encourages players to immerse themselves in the game.

A key achievement of the project is the balanced difficulty and progression system. As players advance through the levels, they encounter different obstacles that increase in complexity, providing a sense of challenge and accomplishment. This balance between accessibility and challenge keeps players engaged and motivates them to continue progressing through the game.

The game also features an intuitive interface and responsive controls, which help ensure that players can focus on the gameplay itself without distractions. The controls are easy to understand, making the game suitable for players of various skill levels.

In summary, the Platformer Game delivers a fun and interactive gaming experience with well-designed levels, smooth mechanics, and simple controls. The project successfully achieves its goal of creating an accessible and enjoyable game, and it serves as a solid foundation for potential future enhancements, such as more levels, features, and challenges.

**IX. BIBLIOGRAPHY**

* + 1. Geeks for Geeks (https://www.geeksforgeeks.org /)
    2. Javatpoint (https://www.javatpoint.com/)
    3. W3Schools (https://www.w3schools.com/)
    4. Draw.io (for flowchart)

These websites were used as references for the project. These helped in the syntaxes, effectiveness and better functioning of the application.

*THANK YOU*