D.K.T.E. Society's Textile and Engineering Institute, Ichalkaranji.

(An Autonomous Institute, Affiliated to Shivaji University, Kolhapur) Accredited with 'A+' Grade by NAAC

Department of Computer Science & Engineering (AI) 2023-2024

SNAKE GAME APPLICATION
ON

Development of "Snake Game Using Python"

Under the guidance of

Miss. Vanchala Sutar **Submitted By:**

Madiha Mujawar	23UAI313
Tejaswini Khot	23UAI311
Vrushali Nikam	23UAI314

D.K.T.E. Society's Textile and Engineering Institute,
Ichalkaranji.

(An Autonomous Institute, Affiliated to Shivaji University, Kolhapur)
Accredited with 'A+' Grade by NAAC

Department of Computer Science & Engineering (AI) 2023-2024

Certificate of Completion

This is to certify that

Madiha Mujawar	**23UAI313**
Tejaswini Khot	**23UAI311**
Vrushali Nikam	**23UAI314**

Have successfully completed the Snake Game Using Python, of the mini project part-II entitled,

”Development of Snake Game Application”

In partial fulfillment for S.Y.B.Tech. CSE(AI) academics. This is the record of their work carried out during academic year 2023-2024.

Date: **Place:**ICHALKARANJI

Miss. Vanchala Sutar
[PROJECT GUIDE] **Prof(Dr).S.K.Shirgave** [EXTERNAL EXAMINER]

[HOD] **Prof(Dr).Mrs.L.S.Admuthe**
[DIRECTOR]

Contents

1 Introduction 5

1.1 Project Overview 5

1.2 Objectives 5

1.3 Scope 5

1.4 Organization of the Document....................5

2 Problem Statement 7

2.1 Background 7

2.2 Identified Problem 7

2.3 Scope of the Problem 7

2.4 Significance of the Problem 8

3 Problem Description 8

3.1 Collision Detection : 8

3.2 Snake Movement : 8

3.3 Game Over Condition : 9

3.4 Scoring System : 9

3.5 Game Speed: 9

3.6 User Interface : 9

3.7 Optimization : 9

3.8 Cross-platform Compatibility : 9

4 Requirement Specification 10

4.1 Functional Requirements 10

4.2 Non-functional Requirements 10

5 Requirement Analysis 11

5.1 User Requirements 11

5.2 System Requirements 11

6 Stakeholders..........12

6.1 Primary Stakeholders 12

6.2 Secondary Stakeholders..........12

7 System Design..........13

7.1 Frontend Design 13

8 DIAGRAMS 14

9 Test Plan 18

9.1 Test Objectives..........18

9.2 Testing Approach..........18

9.3 Test Deliverables..........19

10 References 20

1 Introduction

1.1 Project Overview

Welcome to the project report on the development of the Snake game using Python. In this report, we'll explore the journey of creating a classic arcadestyle game from scratch, leveraging the power of Python programming language and the Pygame library. The Snake game holds a special place in the hearts of gamers worldwide. Its simple yet addictive gameplay has entertained millions for decades. By undertaking the development of the Snake game, we aimed to pay homage to this timeless classic while also delving into the world of game development using Python.

1.2 Objectives

The primary objectives of this project include:

Learn Python Programming Explore

Game Development

Enhance Problem-Solving Skills Receive

Immediate Feedback

Engage with the Community

1.3 Scope

The scope of the project encompasses the following:

Our project scope encompassed the development of a classic Snake game with core functionalities such as controlling the movement of the snake, generating food items, detecting collisions, and tracking the player's score. Additionally, we aimed to implement features such as a graphical user interface (GUI) for an immersive gaming experience and sound effects to enhance gameplay engagement.

1.4 Organization of the Document

This document is structured as follows:

Section 2 discusses the problem statement addressed by the project.

Section 3 provides a detailed description of the problem and its significance.

Section 4 outlines the requirements for the Snake Game.

Section 5 presents the analysis of requirements to guide the design and development process.

Section 6 identifies the stakeholders involved in the project.

Section 7 describes the system design and architecture of the website.

Section 9 outlines the test plan for evaluating the functionality and performance of the Application.

Section 10 lists the references cited throughout the document.

2 Problem Statement

2.1 Background

The Snake game is a classic arcade game that gained popularity in the late 1970s and early 1980s. Originally introduced on arcade machines and later popularized on early mobile phones, the game involves controlling a snake that moves around a bounded area, consuming food items to grow longer. The player's objective is to navigate the snake without colliding into walls or itself, while strategically positioning it to collect as much food as possible.

In recent years, the Snake game has seen various adaptations and implementations across different platforms, with Python emerging as a popular choice for developing simple games due to its ease of use and versatility. The Pygame library, in particular, provides a comprehensive set of tools for developing 2D games and multimedia applications in Python, making it well-suited for creating a Snake game with enhanced graphics and interactivity.

2.2 Identified Problem

Nostalgic Appeal: People love the classic Snake game, but there's a need for a modern Python version that retains the charm while offering new features.

Educational Opportunity: Developing a Python-based Snake game provides a fun and practical way to learn programming and game development concepts

Technical Challenges: Building a fully functional Snake game involves solving various technical problems, like implementing game mechanics and optimizing performance.

Community Engagement: Joining the Python and game development communities through this project offers opportunities for collaboration, learning, and support.

2.3 Scope of the Problem

Platform Compatibility:

Ensure compatibility with various platforms and operating systems, including desktop computers, laptops, and mobile devices. Optimize performance and responsiveness to provide a smooth gaming experience across different devices and screen sizes. Documentation and Support: Create comprehensive documentation, including installation instructions, gameplay guides, and troubleshooting tips. Offer ongoing support and updates to address any issues, bugs, or feature requests reported by players or the community.

Features and Enhancements: Consider additional features to enhance gameplay, such as multiple levels, power-ups, or special game modes. Explore options for customization, allowing players to adjust settings like snake speed or game difficulty. Implement sound effects and background music to complement the visual experience and immerse players in the game world.

2.4 Significance of the Problem

The development of a Snake game using Python holds significant educational value as it provides a practical and engaging platform for learning and practicing programming concepts. By undertaking this project, developers, especially beginners, can apply fundamental programming principles such as loops, conditionals, and functions in a real-world context, while also exploring more advanced topics like object-oriented programming and game development techniques. Additionally, building a Python-based Snake game fosters the development of technical skills such as problem-solving, algorithmic thinking, and debugging, challenging developers to think critically and creatively to overcome various challenges throughout the development process. Moreover, the creative freedom inherent in designing and implementing the game mechanics and visual elements encourages innovation and experimentation, allowing developers to put their own unique spin on the classic game concept and explore new ideas for enhancing the player experience. Furthermore, participation in the Python and game development communities through this project promotes community engagement and collaboration, providing opportunities for networking, mentorship, and collective problem-solving. Ultimately, the development of a Pythonbased Snake game not only offers entertainment and enjoyment for players of all ages but also serves as a valuable addition to a developer's portfolio, showcasing their programming skills, creativity, and ability to deliver a complete project from concept to implementation.

3 Problem Description

3.1 Collision Detection :

Implementing collision detection to detect when the snake collides with itself or with the boundaries of the game window.

3.2 Snake Movement :

Managing the movement of the snake as it grows longer, ensuring that it moves smoothly and responsively in response to user input.

3.3 Game Over Condition :

Determining the conditions under which the game should end, such as when the snake collides with itself or with the boundaries of the game window.

3.4 Scoring System :

Designing a scoring system to keep track of the player's score as they eat food and the snake grows longer.

3.5 Game Speed:

Adjusting the speed of the game to provide an appropriate level of challenge for the player, while ensuring that it remains playable and enjoyable.

3.6 User Interface :

Creating a user-friendly interface for the game, including displaying the score, game over messages, and any other relevant information to the player.

3.7 Optimization :

Optimizing the code to ensure that the game runs smoothly and efficiently, especially as the snake grows longer and the complexity of the game increases.

3.8 Cross-platform Compatibility :

Ensuring that the game can run on different operating systems and environments without encountering compatibility issues.

4 Requirement Specification

4.1 Functional Requirements

1. **Gameplay:** The game should start with a snake of a certain length (e.g., 3 blocks) on a grid. The snake should move continuously in one direction until the player changes its direction or the game ends. The snake should be able to change direction based on user input (e.g., arrow keys). The snake should grow longer when it eats food. The game should end if the snake collides with itself or with the boundaries of the grid.

2. **Scoring:** The player should earn points for each piece of food eaten. The score should be displayed on the screen during gameplay.

3. **User Interface :** The game should have a graphical user interface (GUI) displaying the grid and the snake. The game should have clear and intuitive controls for moving the snake.

4. **Food Generation :** Food should appear randomly on the grid after the snake eats it or at regular intervals. Food should not appear on the snake's body.

5. **Game Over :**When the game ends, the player should be informed and given the option to restart or quit.

4.2 Non-functional Requirements

Performance: The game should run smoothly without lags or delays, even on low-end hardware. Response to user input should be immediate.

Usability: The game should be easy to understand and play, even for users unfamiliar with Snake games. The GUI should have clear visuals and intuitive controls.

Reliability: The game should handle errors gracefully and not crash unexpectedly. It should save the player's high score between sessions.

Scalability: The game code should be modular and easy to extend or modify for future updates or additional features. It should be possible to adjust parameters such as grid size or snake speed without significant code changes.

Compatibility: The game should be compatible with various operating systems that support Python and the required libraries . It should run consistently across different screen sizes and resolutions.

5 Requirement Analysis

5.1 User Requirements

Ease of Use: Users expect the game to be easy to understand and play. They want intuitive controls to navigate the snake. Users may desire clear visual feedback on their actions and the state of the game.

Entertainment Value : Users expect the game to be entertaining and engaging. They may seek challenges and enjoy increasing difficulty levels as the game progresses. Users might appreciate features such as sound effects or background music to enhance the gaming experience.

Scoring and Feedback : Users want to track their progress and compete with themselves or others. They desire clear feedback on their performance, such as the current score and high score. Users may expect indicators of success or failure, such as visual cues when they eat food or collide with obstaclesUsers may desire

Accessibility: Users with disabilities may require accessibility features such as customizable controls or visual aids. The game should be usable for individuals with color blindness or other visual impairments.

Customization Options: Users may desire customization options to personalize their gaming experience. They might appreciate features like adjustable difficulty levels, different grid sizes, or alternative visual themes.

5.2 System Requirements

Hardware Requirements: The game should run on standard hardware configurations commonly found in personal computers and laptops. It should not require excessive processing power or memory, ensuring compatibility with a wide range of devices.

Software Dependencies: The game requires Python installed on the user's system. It may depend on additional libraries or modules, such as Pygame for graphical rendering and input handling.

Performance Considerations : The game should perform efficiently to provide a smooth gaming experience. It should maintain a consistent frame rate and responsiveness, even under heavy load or when rendering complex scenes.

Storage Requirements: The game's storage footprint should be minimal, comprising the size of the game code, assets (such as images or sounds), and any saved game data. It should not consume excessive disk space on the user's system

6 Stakeholders

6.1 Primary Stakeholders

1. **Game Developers/Administration:** As the primary stakeholders, the developers/administration oversee the development and maintenance of the snake game. They are concerned with ensuring that the game accurately represents the desired gaming experience, including mechanics, aesthetics, and overall enjoyment. Additionally, they may set strategic goals and objectives for the game, such as increasing player engagement or improving user experience.

2. **Players:** Players are key stakeholders who interact with the game regularly. They require an intuitive and entertaining gaming experience, including responsive controls, challenging levels, and engaging visuals. Moreover, they may provide feedback on gameplay mechanics and suggest improvements to enhance their gaming experience.

6.2 Secondary Stakeholders

1. **IT Department:** If the game is part of a larger system or platform, the IT department may be responsible for managing technical aspects, such as server infrastructure or deployment processes. They play a crucial role in ensuring the game's availability, performance, and security.

2. **Regulatory Agencies:** While regulatory concerns may not be as prominent in a game development context compared to healthcare, there may still be legal and ethical considerations, especially regarding user privacy (e.g., GDPR compliance) or content ratings (e.g., ESRB ratings). Compliance with relevant regulations ensures that the game reaches its target audience without legal hurdles.

3. **Third-party Service Providers:** Third-party service providers, such as game engine providers, cloud hosting services, or analytics platforms, may be involved in supporting various aspects of the game's development, deployment, or maintenance. Coordination with these stakeholders is essential to ensure smooth project execution and ongoing support.

7 System Design

7.1 Frontend Design

Designing the frontend for a Snake game involves creating a visually appealing and intuitive user interface that displays game elements such as the snake, food, score, and game board. Here's a frontend design overview:

Game Window : The main window where the game is displayed. Provides a canvas for rendering game elements and receiving user input. Typically includes a fixed size or resizable window depending on the game's requirements.

Game Board : Represents the playing area where the snake moves and interacts with other game objects. Consists of a grid of cells or a continuous space depending on the game's design. Rendered using graphics primitives or images to create a visually appealing background.

Snake: The main player-controlled character in the game. Rendered as a series of connected segments or a continuous line representing the snake's body. Each segment may have a different color or texture to distinguish it from the rest of the snake.

Food : Objects that the snake eats to grow and increase the player's score. Rendered as small, colorful objects placed randomly on the game board. Disappear and respawn in a new location when eaten by the snake.

Score Display : Shows the player's current score and possibly other relevant information such as the high score or remaining lives. Positioned at a convenient location on the screen, such as the top or bottom corner.

Game Over Screen : Displays a message or animation when the game ends, indicating whether the player won or lost. Provides options to restart the game, go back to the main menu, or exit the game entirely.

8 DIAGRAMS

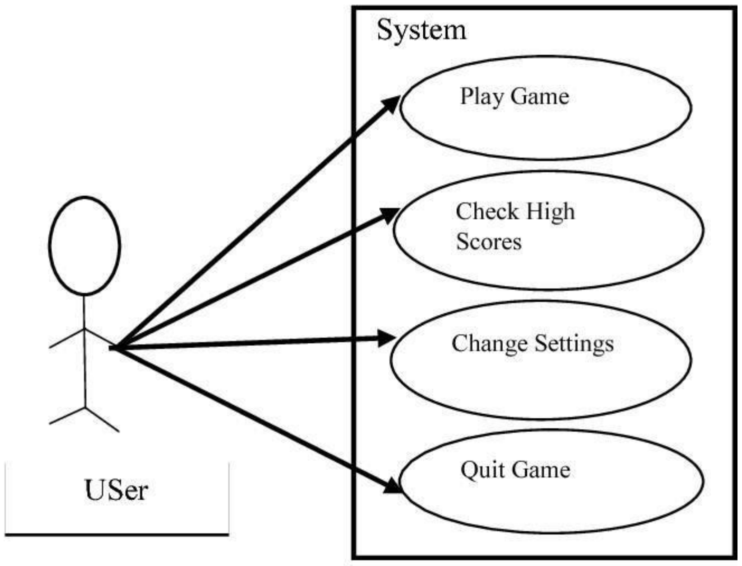


Figure 1: Use Case Dig

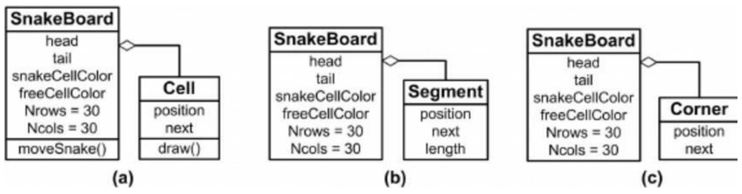


Figure 2: Sequence Dig

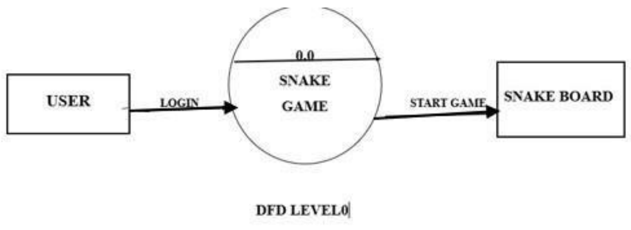


Figure: DFD LEVEL 0

Figure 3: DFD LEVEL 0

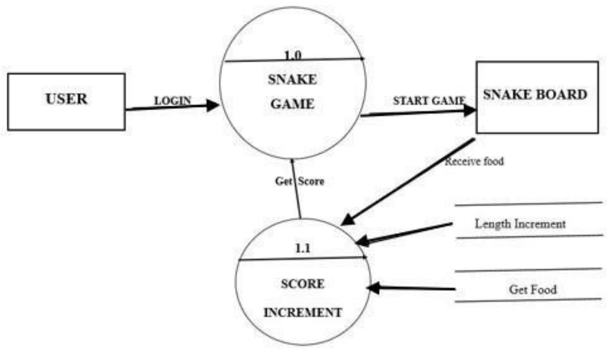


Figure: DFD LEVEL 1

Figure 4: DFD LEVEL 1

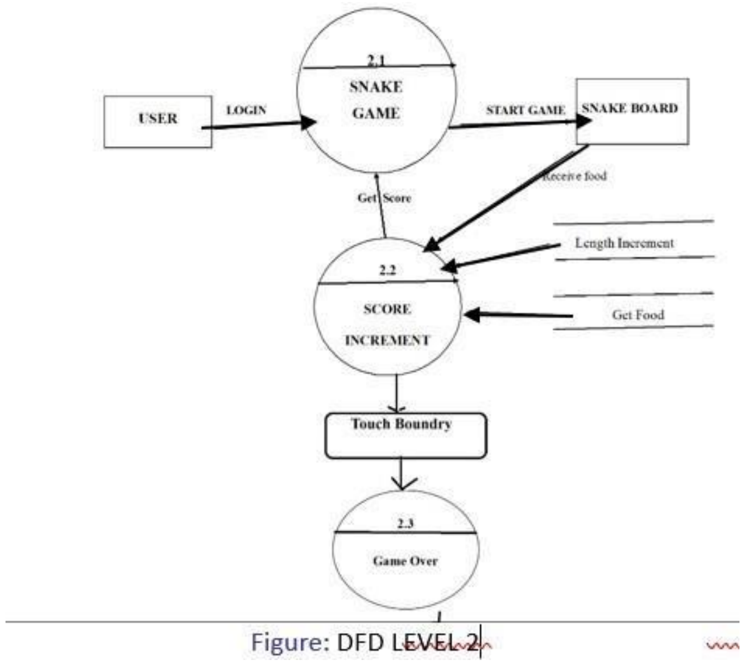
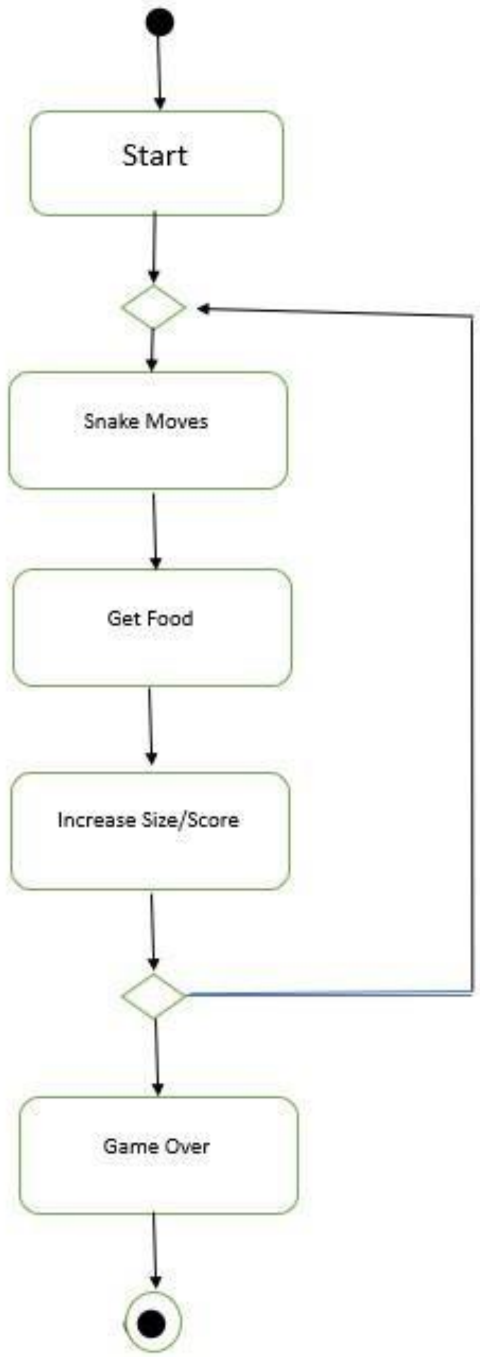


Figure 5: DFD LEVEL 2


Start
Snake Moves
Get Food
Increase Size/Score
Game Over

9 Test Plan

9.1 Test Objectives

The primary objectives of the testing phase are to ensure the functionality, usability, security, and performance of the hospital website. Specific test objectives include:

User Input Handling: Objective: Ensure that the game correctly captures and responds to user input for controlling the snake's movement. Test Cases: Verify that pressing the arrow keys moves the snake in the corresponding direction. Test that the snake does not move if no input is provided. Validate that invalid input (e.g., pressing opposite direction keys simultaneously) is ignored.

Snake Movement : Objective: Verify that the snake moves smoothly and continuously across the game board. Test Cases: Confirm that the snake moves one grid cell at a time in the specified direction. Ensure that the snake's movement speed remains consistent throughout the game. Validate that the snake's movement does not result in glitches or unexpected behavior.

Food Consumption : Objective: Validate that the snake correctly consumes food items and grows in length. Test Cases: Verify that the snake's length increases by one segment after eating a food item. Test that the consumed food item respawns in a random location on the game board. Ensure that the snake's movement and behavior are unaffected by food consumption.

Game Over Conditions : Objective: Ensure that the game correctly handles various scenarios leading to a game over condition. Test Cases: Verify that the game ends when the snake collides with a wall or its own body. Test game over conditions for other scenarios, such as reaching a maximum snake length or time limit. Confirm that the game over screen is displayed with the appropriate message and options.

Graphics and Visuals : Objective: Validate the quality and clarity of game graphics and visual effects. Test Cases: Verify that game elements (snake, food, game board) are rendered clearly and distinguishable. Test the visibility and readability of text elements (score, game over message) against different backgrounds. Ensure that animations and visual effects (e.g., snake movement, food consumption) are smooth and appealing.

9.2 Testing Approach

Functional Testing: Test all game features and scenarios, including positive, negative, and boundary cases, ensuring the game functions correctly.

Usability Testing: Evaluate the user interface for intuitiveness and accessibility, gathering feedback from users to improve the overall usability.

Security Testing: Automated vulnerability scanning using security testing tools (e.g., OWASP ZAP, Nessus) and manual penetration testing to identify and remediate security vulnerabilities.

Performance Testing: Assess the game's performance under different load conditions, monitoring system resources and scalability to ensure optimal gameplay experience.

9.3 Test Deliverables

The following deliverables will be produced as part of the testing phase:

Functional Testing Deliverables: Test cases document outlining all functional test scenarios. Test execution reports detailing the results of each test case (pass/fail). Defect reports documenting any issues found during testing, including steps to reproduce and severity.

Usability Testing Deliverables: Usability test plan outlining objectives, methodology, and test scenarios. Usability test scripts for conducting user testing sessions. Usability test report summarizing findings, feedback, and recommendations for improving usability.

Security Testing Deliverables: Security test plan detailing the approach, methodology, and test scenarios. Security test reports documenting vulnerabilities found, their severity, and recommendations for mitigation.

Performance Testing Deliverables: Performance test plan outlining performance goals, scenarios, and workload models. Performance test scripts for executing load, stress, and scalability tests. Performance test reports containing metrics, analysis, and recommendations for optimizing performance.

10 References

1. https://www.edureka.co/blog/snake-game-with-pygame/
2. https://pypi.org/project/pygame/
3. https://www.edueba.com/python-pygame/
4. https://www.programmiz.com/python-programming/time