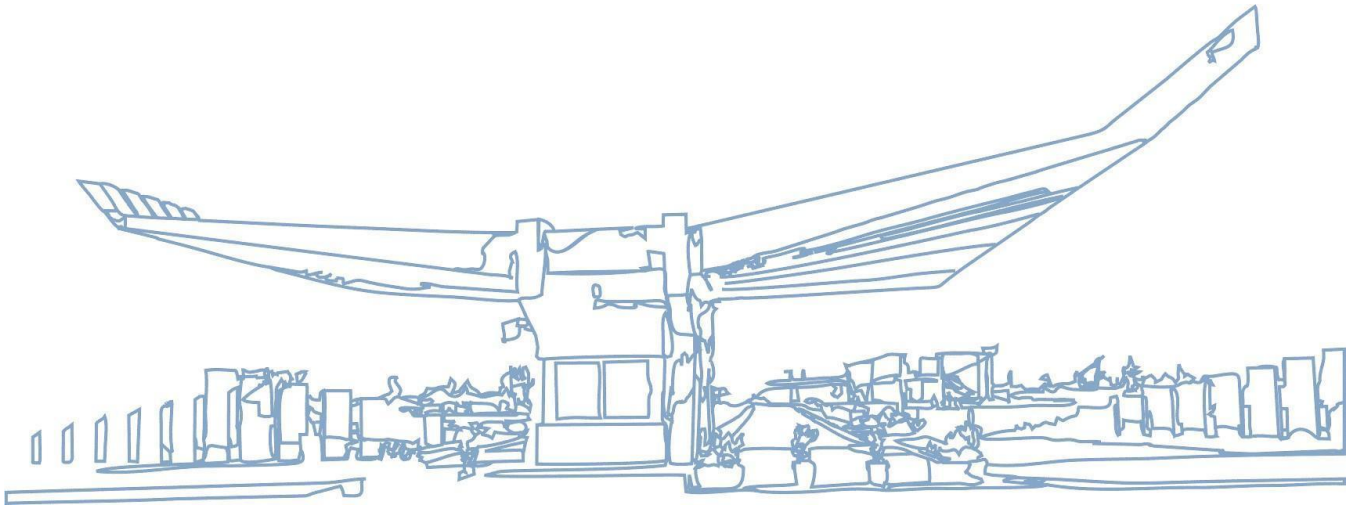
EPOKA UNIVERSITY

INTRODUCTION TO SOFTWARE ENGINEERING

Bob Ndertusat



Team Members:

Klaudia Tamburi

Nidia Fino

Juna Dako

Sara Onjea

Snake Game Requirements Specification

EPOKA UNIVERSITY

Table of Contents

1. Executive Summary

1.1 Project Overview

The Snake Game project aims to develop a classic arcade-style game where players control a snake to eat food and grow in length while avoiding collisions with obstacles or the snake's own body. The objective is to achieve the highest score possible.

1.2 Purpose and Scope of this Specification

This specification document outlines the requirements for the Snake Game, providing a comprehensive understanding of its features, functionalities, and design considerations.

This document addresses the requirements related with the first release of the Snake Game regarding its features, without including features like “Sound effect” or “Multiple player mode” that are planned to be included in the following releases.

2. Product/Service Description

2.1. Product Context

The Snake Game is typically an independent and self-contained product. It does not necessarily have direct dependencies on other products or systems-aside from needing an IDE supporting python to run the code for now. The initial plan is for the game to be developed as a standalone desktop application for Windows and Mac platforms. However in the future, it can be part of a larger gaming platform or integrated into a gaming console where users can access multiple games. The Snake Game will utilize modern graphics and audio libraries to provide an engaging gaming experience.

2.2. User Characteristics

The targeted audience for the first release is the younger generation of programmers that enjoy gaming and would like to have some fun with our snake code in between their coding breaks, since the users are expected to have basic computer literacy and familiarity with keyboard controls.

2.3. Assumptions

It is assumed that the user has some basic knowledge about computers and also some general knowledge about programming. And also the user has to have a laptop or PC with an IDE installed that supports Python 3 to be able to run the program and play our Snake Game.

2.4. Constraints

Throughout the development of the Snake Game, we encountered both time and software constraints. With a fixed deadline for project completion, we faced the challenge of efficiently managing our time to ensure effective planning and execution of the development process. This involved making strategic decisions to prioritize tasks and allocate resources appropriately.

In terms of software constraints, our team encountered limitations in relation to object-oriented programming, as this was a relatively new experience for all of us. We had to work within the confines of the Python programming language and the specific libraries we selected as beginners, which presented certain limitations that influenced the design and implementation of the game. Despite these

constraints, we adapted our approach and utilized the available resources to create an engaging and functional Snake Game.

Whereas regarding the final product, when the players get the game, there will be system constraint since the game will only run in an IDE supporting Python 3.

2.5. Dependencies

Firstly the Snake Game depends on an IDE that supports python. And the code itself depends on the turtle library for the visuals, the time library for the speed of animation and on the random library for the random generation of the food position.

3. Requirements

3.1. Functional Requirements

Req#	Requirement	Comments	Priority	Date Rvwd	SME Reviewed / Approved
R_01	The system should be able to display a game board for the snake with fixed size	Business Process = "Changing dues in the System" - Creating the window display -KT Business Process = "Maintenance" - 08/05/23 - Updating the window display, removing the resizable feature- KT	2	04/05/23 08/05/23	aprvd:Klaudia Tamburi, rvwd:Sara Onjea
R_02	The system should display an initial snake body.	Business Process = "Changing Dues in the System" - 05/05/23 - Creating snake head -KT Creating initial snake body connected with the head - JD	2	06/05/23	aprvd:Klaudia Tamburi, Juna Dako, rvwd:Nidia Fino

R_03	The system should use keyboard input to move the snake	Business Process = "Changing Dues in the System" - 05/05/23- Creating keybindings to move the snake head -KT 06/05/23 - Priority changed from 2 to 1 - KT Business Process = "Changing Dues in the System" - 07/05/23 - Creating the function for body to follow head - JD	~~2~~ 1	06/05/23 08/05/23	aprvd:Klaudia Tamburi, Juna Dako, rvw:Nidia Fino
R_04	The Snake Should eat food generated in random positions	Business Process = "Changing Dues in the System" - 08/05/23- Creating the food set up and function to generate food position randomly after snake eats the food- SO	2	09/05/23	aprvd:Sara Onjea, rvwd:Juna Dako
R_05	The game should be over and the snake stops moving and growing when it eats the tail or crashes with the borders.	Business Process = "Changing Dues in the System" - 09/05/23- Creating function to stop moving when snake eats tail - SO 12/05/23 - Creating function to stop moving when the snake touches the borders - NF	2	11/05/23 13/05/23	aprvd:Sara Onjea, Nidia Fino, rvwd:Klaudia Tamburi

R_06	The system should be able to keep track and display the current score and the highest score achieved in the game	Business Process = "Changing Dues in the System" - 10/05/23 - Creating scoring system to print and display the accumulated score whenever the snake grows and the highest achieved - NF	2	13/05/23	aprvd:Nidia Fino, rvw:Juna Dako
R_07	The system should be able to reset the game. The snake is back to its initial conditions and the current score goes back to zero to start from the beginning.	Business Process = "Changing Dues in the System" - 11/05/23 - Creating the function for restarting and resetting the score inside the function- NF 12/05/23 - Modifying the function to reset the snake body to its initial state - JD 12/05/23 - Modifying the function to set the head to its initial position and create key-binding for the function (player resets the game he presses "R")	3	14/05/23	aprvd:Juna Dako, Klaudia Tamburi, Nidia Fino rvwd:Sara Onjea
R_08	The system should be able to speed up the snake movement whenever the score reaches a certain value to make the game more challenging.	Business Process = "Changing Dues in the System" - 23/05/23 - Creating function to change the screen delay- NF	3	24/05/23	aprvd:Nidia Fino, rvw:Juna Dako

R_09	The system should be able to have sound effect	June 2023 - New requirement	3		
R_10	The system should be able to pause the game	June 2023 - New requirement	3		
R_11	The system should be able to switch between 1 player and 2 player mode	June 2021 - New requirement	2		

3.2. Non-Functional Requirements

- Product Requirements:
  - Functional Requirements
    Snake movement mechanics (e.g., turning, growing, collision detection)
    Food generation and consumption
    Game over conditions (e.g., hitting walls, colliding with the snake's body)
    Score tracking and display
    High scores management
  - User Interface Requirements
    Game settings
    Keyboard input
    Graphics and visual effects
    Sound effects and background music (not covered in this document)
  - Compatibility Requirements
    Cross-platform compatibility (e.g., Windows, macOS, Android, iOS)
    Screen resolution and aspect ratio adaptation
    Input device compatibility (The game requires a physical keyboard)
    Requires software libraries and programming language (Python)
- Organizational Requirements:
  - Development Methodology
    For the fulfillment of this project, we employed the test-driven development approach. Initially, we participated in discussions regarding the coherence development of the code., detailing the step-by-step procedures and expected functionality. Afterwards, individual programmers were assigned specific functions, and they began their respective work. Thorough tests were conducted on each function, followed by interchanging them amongst programmers to ensure the seamless integration of their efforts. Through the repetition of this

iterative process, we effectively achieved the delivery of an elementary yet fully operational snake game.

- Collaboration and communication
  We used different communication channels to keep in touch with each other during the development of the game. We kept track of the code and documented the changes and shared them during the meetings.

3.2.1. User Interface Requirements

- Screen Format/Organization:

- The game is displayed in a graphical window with a size of 700x650 pixels.
- The window has a green background color and a title "Snake Game by Bob Ndertusat".

- Scores Display:

- The current score and the high score are shown at the top of the window using the `turtle` module's pen.
- The scores are updated during gameplay.

- Keyboard Controls:

- The game is controlled using the arrow keys (up, down, left, right) and the letter keys (W, S, A, D).
- Pressing the arrow keys or their corresponding letter keys moves the snake's head in the corresponding direction.
- Pressing the letter "R" restarts the game.

- Game Over Message:

- When the game is over (collision with borders or body), a "GAME OVER! Press 'R' to restart." message is displayed in the center of the window.
- Pressing the letter "R" after the game over message restarts the game.

- Graphics and Animation:

- The snake's head is represented by a circular shape with a dark blue color.
- The snake's body is made up of multiple circular segments with a light blue color.
- The food (apple) is represented by a triangular shape with a red color.
- The snake's head and body segments move smoothly through the graphical window.
- The snake speeds up a little each time it eats 5 apples in a row.

3.2.2. Usability

- Learnability:
  The game has a simple and intuitive interface that is easy to understand and navigate and the game mechanics are designed to be easy to grasp and allow the player to learn and improve their skills quickly.
  There are clear instructions provided about the game in our user guide.

- User Documentation and Help:
  To access the comprehensive user guide document please click the following link:
  https://drive.google.com/file/d/1-RJEHfi9yltVFnNZc__ipGoD_S41r3jU/view?usp=sharing

3.2.3. Performance

- Static Numerical Requirements:
- Amount and Type of Information: The game should handle and display relevant information, such as the player's score, high scores, and game settings, in a timely manner.

- Dynamic Numerical Requirements:
- Number of Transactions and Tasks: The system should be able to handle a certain number of game-related transactions and tasks, such as snake movements, food generation, and collision detection, within a specific time period.
- Amount of Data to be Processed: The game should efficiently process and update data related to the game state, such as the snake's position, food positions, and score, during normal gameplay and peak workload conditions.
- Response Time: A measurable response time should be specified for critical operations or interactions, such as moving the snake, updating the game state, or displaying changes on the screen. For example, in our code, the time delay that the code introduces to us is a delay of 0.14 seconds from the start and then will be decremented until it reaches 0.05 for every 5 apples.

3.2.4. Manageability/Maintainability

3.2.4.1. Monitoring

- Health Monitoring Requirements:
- The game should continuously monitor the health of critical components such as the game loop, keyboard input, and collision detection.
- It should detect any anomalies or failures in these components and take appropriate actions.

- Failure Conditions:
- The game handles and recovers from failure conditions such as collisions with borders or the snake's body.
- Upon encountering a failure condition, the game provides feedback to the user, pauses the game, and provides an option to restart.

- Error Detection:
- The game includes error detection mechanisms to identify issues with inputs, such as detecting invalid keystrokes or unexpected behavior.

- Error Correction:
- It should offer the ability to restart the game to rectify any errors or failures encountered during gameplay.

3.2.4.2. Maintenance

The snake game system is designed with a simple structure where every feature is divided and created on a single function creating consistency within the code and making it easier to do periodic maintenance and modifications. The code has low complexity and is neatly written so there is no difficulty in reading and understanding it during regular checks.

3.2.4.3. Operations

- Periods of interactive operations and periods of unattended operations
  The user can control the snake's movements using the keyboard keys (arrow keys or letter keys), restart the game by pressing the "R" key, and observe the score updates and game over messages displayed on the screen. During these periods of interactive operations, the user actively engages with the game, providing input and receiving visual feedback.The periods of unattended operations refer to the gameplay itself, where the snake moves automatically based on the predefined logic. The snake's movement and collision detection are handled by the game's internal mechanisms, without requiring direct user input.
- Data processing support functions
  Data processing support functions in the context of the snake game include functions for updating scores, detecting collisions, generating random coordinates for the food (apple), managing the snake's body segments, and handling graphical elements like the turtle objects representing the snake, food, and score display.
- Safety considerations and requirements
  Regarding safety considerations, the code does not have explicit safety measures, as it is a simple snake game. However, it is important to reassure that the game does not cause harm or damage to the user's system or data.

3.3. Domain Requirements

- Window Configuration
- The game window should have a width of 700 pixels and a height of 650 pixels.
- The window should not be resizable.
- The window should have a title "Snake Game by Bob Ndertusat".
- The background color of the window should be set to '#60D164'.
- The animation/screen updates should be turned off initially.

- Score Display:
- The game should display the current score and the high score.
- The score display should be shown using a turtle pen.
- The initial score and high score should be set to 0.
- The score display should be centered and written using the Courier font with a font size of 12.

- Snake Head:
- The snake head should be displayed using a turtle.
- The snake head should have a circular shape.
- The snake head should have an animation speed of 0 (no delay).

- The snake head should have a size 1.1 times bigger than the basic turtle size.
- The snake head's color should be set based on the selected snake color.

- Snake Food:
- The game should display snake food using a turtle.
- The snake food should have a triangular shape.
- The snake food should have an animation speed of 0 (no delay).
- The snake food's color should be set to "red".
- The initial position of the snake food should be at coordinates (80, 0).

- Keyboard Bindings:
- The game should listen for keyboard input.
- The following key bindings should be implemented:
- "W" or "w" or "Up" - Moves the snake upwards.
- "S" or "s" or "Down" - Moves the snake downwards.
- "A" or "a" or "Left" - Moves the snake to the left.
- "D" or "d" or "Right" - Moves the snake to the right.
- "R" or "r" - Restarts the game.

- Game Loop:
- The game should continuously run in a loop.
- The screen should be updated in each iteration of the loop.
- The snake's movement, collision detection, and scoring should be updated within the loop.
- The loop should introduce a delay which is calculated with a function that speeds up the movement based on the snake's growth.

4. Design thinking methodologies

4.1. Empathy:

During the empathy phase, we adopted a user-centric perspective by placing ourselves in the shoes of our target audience rather than solely relying on our roles as code developers. This approach allowed us to gain a deeper understanding of the problem at hand. Considering our target audience consists of beginner programmers, we recognized the importance of implementing all the fundamental rules of a traditional snake game while providing clear and easily comprehensible code to our audience. We aimed to apply simple and basic, yet visually appealing graphics for a good user experience.

4.2. Define:

In the Define stage, we gathered as a team to analyze the information we had gathered during the Empathy stage. We realized that the core problem was to create a snake game that would provide a balance between challenge and fun. To form a clear problem statement or design challenge, we discussed with our teammates and potential players (beta testers). We discussed and refined the problem statement together, ensuring that everyone's perspectives were considered.

4.3. Ideate:

During our ideation sessions, we gathered around and finally expressed all the ideas which each programmer had envisioned for this project. Throughout the ideation phase, we encouraged a

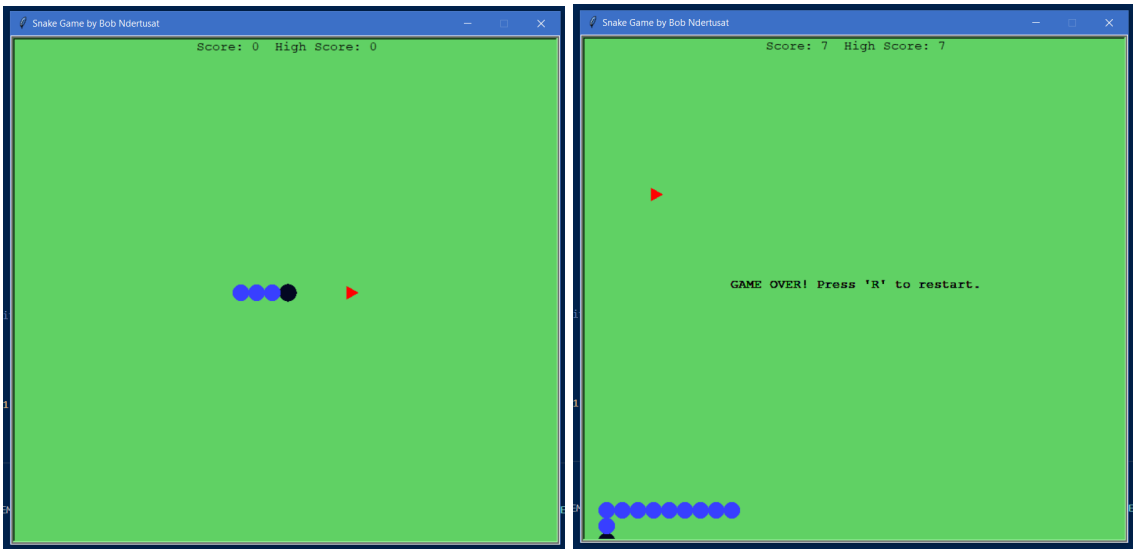
collaborative and open-minded approach, valuing each idea and building upon them collectively. We kept an open communication, allowing for discussions and debates to refine and prioritize the most promising concepts. We further discussed the design of the game as well as the additional idea to make the game more interesting by every 5 apples, incrementing the speed. This was the most engaging and entertaining part for each programmer.

4.4. Test:

Throughout the code development process, we implemented a rigorous approach to ensure code quality. This involved conducting unit tests for each function within the team. One team member would create a function while another would thoroughly test it, verifying its correctness and alignment with product requirements, as well as identifying and addressing any potential vulnerabilities or issues.

Once the code was completed, we proceeded with further comprehensive functional and performance tests. To gather valuable feedback, we provided the source code to a select group of friends who belonged to our target audience. Their continuous gameplay and periodic feedback enabled us to identify any issues or bugs that may have arisen. Any reported problems were promptly addressed and resolved by our team.

4.5. GUI



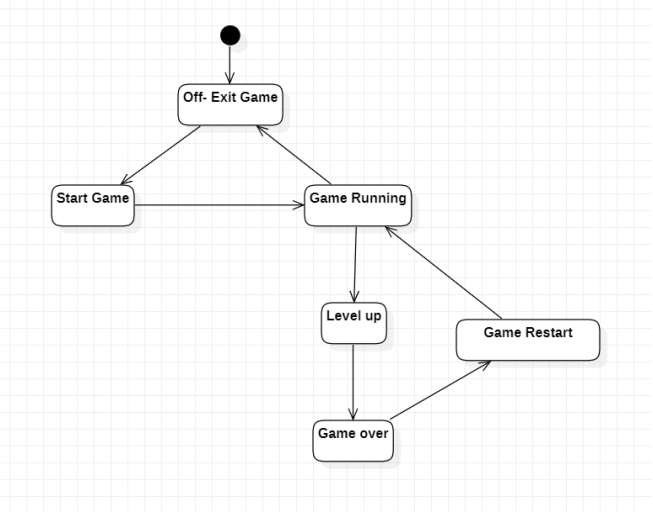
5. Software Design

5.1. Use Case

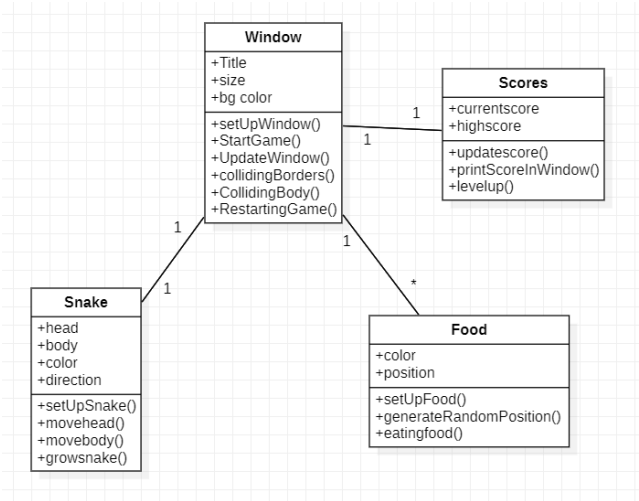
The product, Snake Game, will be able to perform these main functions:
1- Start Game: Open the window where the game board to play the snake game is.
2- Move Snake: The player can play the game while moving the snake with keyboard input (like the arrow keys).
3- Grow Snake: The player can make the snake to eat food and to grow each time,game

difficulty increases.
4- Game Over: The game is over when the snake eats itself or crashes with the borders. Teh snake stops moving and growing.
5- Reset: The player can choose to reset the game and start from the beginning by using keyboard input (pressing ‘r’).
6- Exit Game: When the player doesn’t want to play anymore they can close the game board (game window) with the closure button.
Check also use case diagram here:
https://docs.google.com/document/d/1N128lOXkGvgxkXCgYlf7zodHrqzK0aIM/edit?usp=sharing&ouid=104408659093492863891&rtpof=true&sd=true

5.2. State Diagram



5.3. Class Diagram



APPENDIX

Appendix A. Definitions, Acronyms, and Abbreviations

- **GUI - Graphical User Interface:** It refers to the visual interface that allows users to interact with software or applications using graphical elements such as windows, buttons, menus, icons, and other visual components. The GUI provides an intuitive and user-friendly way for users to interact with and navigate through the software, making it easier to perform tasks and access various features and functionalities.
- **IDE - Integrated Development Environment:** It is a software application that provides comprehensive tools and features to facilitate software development. An IDE typically includes a source code editor, compiler or interpreter, debugging tools, and other utilities that streamline the development process.
- **PYTHON -** Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance.

Appendix B. References

- https://drive.google.com/file/d/1-RJEHfi9yltVFnNZc__ipGoD_S41r3jU/view?usp=sharing
- https://docs.google.com/document/d/1N128lOXkGvgxkXCgYlf7zodHrqzK0aIM/edit?usp=sharing&ouid=104408659093492863891&rtpof=true&sd=true