A Project Report on

CLASSIC SNAKE GAME IMPLEMENTATION USING PYTHON

At

SREENIVASA INSTITUTE OF TECHNOLOGY AND MANAGEMENT STUDIES CHITTOOR

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of technology

By

R. Aswin - 20751A05E4

T. R. Guna Sekhar - 20751A05H8

V. Pavan Kumar - 20751A05I9

V. Sharath Kiran - 20751A05J0

Under the esteemed guidance of

Mr. A. SRINIVASAN, M. Tech

Associate Professor

Department of Computer Science and Engineering

SREENIVASA INSTITUTE OF TECHNOLOGY AND MANAGEMENT STUDIES

(Affiliated to JNTU Anantapur, Anantapur) Murukambattu, Chittoor – 517 127

JULY 2023

SREENIVASA INSTITUTE OF TECHNOILOGY AND MANAGEMENT STUDIES(AUTONOMOUS)

Affiliated To J.N.T.U.A, Anantapur & Accredited By NAAC

CHITTOOR-517127

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that the project work entitled "Classic Snake Game Implementation Using Python" is carried out by R. Aswin(20751A05E4), T. R. Guna Sekhar(20751A05H8), V. Pavan Kumar (20751A05I9), V. Sharath Kiran(20751A05J0) under my supervision and guidance Mr. A. Srinivasan during the academic year 2020-2024, in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology.

Mr. A. SRINIVASAN, M. Tech
Internal Guide

Mr. P. SUDHEER, Ph. D
Head of CSE Department

Submitted for Viva-Voce examination held on

Internal Examiner

External Examiner

DECLARATION

We affirm that the project work titled "**Classic Snake Game Implementation Using Python**" being submitted in partial fulfillment for the award of **Bachelor of technology** is the original work carried out by me. It has not formed the part of any other project work submitted for award of any degree, either in this or any other university.

Name of the candidate	**Roll No**	**Signature of the candidate**
R Aswin	20751A05E4	
T R Guna Sekhar	20751A05H8	
V Pavan Kumar	20751A05I9	
V Sharath Kiran	20751A05J0	

I certify that the declaration made above by the candidates are true.

(Signature of the Guide)

Mr. A. SRINIVASAN, M. Tech

ACKNOWLEDGEMENT

Predominantly our thanks goes to late **Sri. D. K. AUDIKESAVULU Garu** Founder, Mrs. Sri **K. RANGANATHAM Garu** Chairperson, SITAMS for the extensive lab facilities provided in the college.

We would like to express our profound gratitude to our principal **Dr. N. VENKATACHALAPATHI Garu**, and **Mr. P. SUDHEER, Ph. D, HOD of CSE Department**. For offering us a chance to surve in our reputed institution and providing all possible facilities throughout the completion of my project.

We express our sincere thanks to our project guide **Mr. A. SRINIVASAN**, **M. Tech, Associate Professor, and Mr. A. VENKATESAN, M.E, Assistant Professor in CSE Department**, offering us the opportunity to do this work, for their benevolent advice and guidance at each step of our project.

Our sincere thanks to all teaching and non-teaching staff members of CSE Department for their cooperation and guidance.

We are deprived of words to account to the cooperation, motivation and support extended by our parents and friends at all the moments and hours of this academic venture.

Finally, we extend our thanks to one and all, whoever helped us directly and indirectly for this presentation in most appropriate and attractive form.

SREENIVASA INSTITUTE OF TECHNOLOGY AND MANAGEMENT STUDIES (AUTONOMOUS)

(Affiliated To J.N.T.U.A, Anantapur & Accredited by NAAC & NBA)

CHITTOOR - 517 127, Andhra Pradesh.

(2020-2024)

Institute Vision

To emerge as a Centre of Excellence for Learning and Research in the domains of engineering, computing and management.

Institute Mission

- Provide congenial academic ambience with state-art of resources for learning and research.
- Ignite the students to acquire self-reliance in the latest technologies.
- Unleash and encourage the innate potential and creativity of students.
- Inculcate confidence to face and experience new challenges.
- Foster enterprising spirit among students.
- Work collaboratively with technical Institutes / Universities / Industries of National and International repute

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION

To contribute for the society through excellence in Computer Science and Engineering with a deep passion for wisdom, culture and values.

MISSION

M 1: Provide congenial academic ambience with necessary infrastructure and learning resources.

M 2: Inculcate confidence to face and experience new challenges from industry and society.

M 3: Ignite the students to acquire self-reliance in State-of-the-Art Technologies

M 4: Foster Enterprising spirit among students

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs):

Graduates of Computer Science and Engineering shall

PEO1: Excel in Computer Science and Engineering program through quality studies, enabling success in computing industry. **(Professional Competency)**

PEO2: Surpass in one's career by critical thinking towards successful services and growth of the organization, or as an entrepreneur or in higher studies. **(Successful Career Goals)**

PEO3: Enhance knowledge by updating advanced technological concepts for facing the rapidly changing world and contribute to society through innovation and creativity. **(Continuing Education and Contribution to Society)**

PROGRAM SPECIFIC OUTCOMES (PSO's):

Students Shall

PSO1: Have Ability to understand, analyze and develop computer programs in the areas like algorithms, system software, web design, big data analytics, and networking.

PSO2: Deploy modern computer languages, environment, and platforms in creating innovative products and solutions.

PROGRAMME OUTCOMES (PO's)

Computer Science and Engineering Graduates will be able to: **PO1 - Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization for the solution of complex engineering problems.

PO2 - Problem analysis: Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3 - Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for public health and safety, and cultural, societal, and environmental considerations.

PO4 - Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5 - Modern tool usage: Ability to design and develop hardware and software in emerging technology environments like cloud computing embedded products, real-time systems, Internet of Things, Big Data, etc.

PO6- Engineering and Society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice

PO7- Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8- Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9 - Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10 - Communication: Communicate effectively on complex engineering activities with the engineering community and with the society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11 - Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12 - Life-long learning: Basic knowledge in hardware/software methods and tools for solving real-life and R&D problems with an orientation to lifelong learning.

COURSE OUTCOMES FOR PROJECT WORK

On completion of project work we will be able to,

CO1. Demonstrate in-depth knowledge on the project topic.

CO2. Identify, analyze and formulate complex problem chosen for project work to attain substantiated conclusions.

CO3. Design solutions to the chosen project problem.

CO4. Undertake investigation of project problem to provide valid conclusions.

CO5. Use the appropriate techniques, resources and modern engineering tools necessary for project work.

CO6. Apply project results for sustainable development of the society.

CO7. Understand the impact of project results in the context of environmental sustainability.

CO8. Understand professional and ethical responsibilities while executing the project work.

CO9. Function effectively as individual and a member in the project team.

CO10. Develop communication skills, both oral and written for preparing and presenting project report.

CO11. Demonstrate knowledge and understanding of cost and time analysis required for carrying out the project.

<u>CO-PO MAPPING</u>

COs \POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	√										
CO2		√									
CO3			√								
CO4				√							
CO5					√						
CO6						√					
CO7							√				
CO8								√			
CO9									√		
CO10										√	
CO11											√

EVALUATION RUBRICS FOR PROJECT WORK:

Rubric (CO)	**Excellent (Wt. = 3)**	**Good (Wt. = 2)**	**Fair (Wt. = 1)**
Selection of Topic (CO1)	Select a latest topic through complete knowledge of facts and concepts.	Select a topic through partial knowledge of facts and concepts.	Select a topic through improper knowledge of facts and concepts.
Analysis and Synthesis (CO2)	Thorough comprehension through analysis/ synthesis.	Reasonable comprehension through analysis/ synthesis.	Improper comprehension through analysis/synthesis.
Problem Solving (CO3)	Thorough comprehension about what is proposed in the literature papers.	Reasonable comprehension about what is proposed in the literature papers.	Improper comprehension about what is proposed in the literature.
Literature Survey (CO4)	Extensive literature survey with standard references.	Considerable literature survey with standard references.	Incomplete literature survey with substandard references.
Usage of Techniques &Tools (CO5)	Clearly identified and has complete knowledge of techniques & tools used in the project work.	Identified and has sufficient knowledge of techniques & tools used in the project work.	Identified and has inadequate knowledge of techniques & tools used in project work.
Project work impact on Society (CO6)	Conclusion of project work has strong impact on society.	Conclusion of project work has considerable impact on society.	Conclusion of project work has feeble impact on society.
Project work impact on Environment (CO7)	Conclusion of project work has strong impact on Environment.	Conclusion of project work has considerable impact on environment.	Conclusion of project work has feeble impact on environment.
Ethical Attitude (CO8)	Clearly understands ethical and social practices.	Moderate understanding of ethical and social practices.	Insufficient understanding of ethical and social practices.

Independent Learning (CO9)	Did literature survey and selected topic with a little guidance	Did literature survey and selected topic with considerable guidance	Selected a topic as suggested by the supervisor
Oral Presentation (CO10)	Presentation in logical sequence with key points, clear conclusion and excellent language	Presentation with key points, conclusion and good language	Presentation with insufficient key points and improper conclusion
Time and Cost Analysis (CO11)	Comprehensive time and cost analysis	Moderate time and cost analysis	Reasonable time and cost analysis
Continuous learning (CO12)	Highly enthusiastic towards continuous learning	Interested in continuous learning	Inadequate interest in continuous learning

TABLE OF CONTENTS

ABSTRACT

The main objective of this project is to implement the classic snake game using the python programming language. This project utilizes the Pygame library to create an interactive gaming experience for the user. This snake game involves controlling a snake to navigate through a grid, consuming food to grow longer while avoiding collisions with the snake's own body or the boundaries. In this project, initially we create a grid system where the snake and food item will be positioned. Enable user input to control the snake's movement (up, down, left, right) and update its position accordingly. Randomly generate food items within the grid for the snake to consume. Implement collision detection mechanisms to check if the snake has collied with the boundaries or its own body. Track and display the player's score on the number of food items consumed. Last but definitely not the least, we need to display the score and provide the options to restart or quit the game. This project aims to show the fundamental concepts of game development and enhance the participants' programming skills. Finally, we implement the snake game in the IDLE shell.

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Playing games is fun and exciting. It gives us relief from stress and unwinds from our stressful work. Many of us spend our free time or others that use most of their time in playing and exploring new games. Today, with the raped development of technology we have, games that are rising up together with it.

Nowadays with technology we have many games that are developed for computers specifically for windows. With the high technology equipped with these computer games become robust and attract many people to buy or have this gadget for them to experience what's inside it which makes it a trend for the new generation of gadget.

Snake game is a computer action game, whose goal is to control a snake to move and collect food in a map. It has been around since the earliest days of home computing and has re-emerged in recent years on mobile phones.

It isn't the world's greatest game, but it does give you an idea of what you can achieve with a simple python program, and perhaps the basis by which to extend the principles and create more interesting games on your own. To move the snake, use up arrow for up, down arrow for down, left arrow for left and right arrow for right. Press "Q” to exit the game at any time, press 'C" to continue the game.

The aim of the game is to collect the dots (food) and avoid the obstacles (walls, boundaries). As you collect the food, the snake gets longer. The score also increases. There is no concept of life. Once you hit an obstacle, that's it, game over.

1.1 OBJECTIVE

Snake game is one of the most popular arcade games of all time.

In this game, the main objective of the player is to catch the maximum number of food items without hitting the wall or itself.

Creating a snake game can be taken as a challenge while learning Python or Pygame. It is one of the best beginner – friendly projects that every novice programmer should take as a challenge.

Learning to build a video game is kind of interesting and fun learning.

CHAPTER 2

SYSTEM ANALYSIS

System analysis is a process of gathering and interpreting facts, diagnosing problems and the information about the snake game implementation using python. It is a problem solving activity that requires intensive communication between the system users and system developers. System analysis is an important phase of any system development process. The system is studied to the minutest detail and analyzed. The system analyst plays the role of the interrogator and dwells deep into the working of the present system. The system is viewed as a whole and the input to the system are identified.

System analysis is concerned with becoming aware of the problem, identifying the relevant and decisional variables, analyzing and synthesizing the various factors and determining an optimal or at least a satisfactory solution or program of action: A detailed study of the process must be made by various techniques like interviews, questionnaires etc. The data collected by these sources must be scrutinized to arrive to a conclusion. The conclusion is an understanding of how the system functions. This system is called the existing system. Now the existing system is subjected to close study and problem areas are identified. The designer now functions as a problem solver and tries to sort out the difficulties that the user faces. The solutions are given as proposals.

The proposal is then weighed with the existing system analytically and the best one is selected. The proposal is presented to the user for an endorsement by the user. The proposal is reviewed on user request and suitable changes are made. This is loop that ends as soon as the user is satisfied with proposal. Preliminary study is the process of gathering and interpreting facts, using the information for further studies on the system. Preliminary study is problem solving activity that requires intensive communication between the system users and system developers.

2.1 EXISTING SYSTEM

In the existing snake game implementation, we created a simple layout screen where the snake moves continuously. Also, the existing snake game contains only default colors such as white and black. In the existing game, the snake navigates through the boundaries and the snake's speed is high. We can utilise the power of the Pygame library to build the snake game better than how it is now.

However, developing the game from scratch allows for a better understanding of game development concepts**.**

The traditional snake game implemented in python usually involves a simple grid-based layout where the snake moves continuously and the player's objective is to eat food and grow in length without colliding with the boundaries or its own body.

2.1.1 DRAWBACKS

There are few drawbacks of the existing system. Some of them are as follows:

- Lack of Graphics
- Lack of User interface
- Limited Gameplay Features

2.2 PROPOSED SYSTEM

The proposed system involves designing and implementing the snake game using python and the Pygame library. The key components of the project include:

1. **Grid Creation:** Create a grid system where the snake and food item will be positioned.
2. **Snake Movement:** Enable user input to control the snake's movement (up, down, left, right) and update its position accordingly.
3. **Food Generation:** Randomly generate food items within the grid for the snake to consume.
4. **Collision Detection:** Implement collision detection mechanisms to check if the snake has collied with the boundaries or its own body.
5. **Scoring System:** Track and display the player's score on the number of food items consumed.
6. **Game Over and Restart:** Handle game over scenario's and provide the options to restart or quit

the game.

2.2.1 ADVANTAGES

The advantages of the proposed system are as follows:

- Customization Options
- Introduction to power-ups
- Display high score
- Better game play

CHAPTER 3

PROBLEM FORMULATION

3.1 DESCRIPTION OF PROBLEM DOMAIN

The core focus of our project was to determine which algorithms would be more effective in a hard real-time environment. The domain in this case is the Snake Game, which will, in turn, attempt to identify an, or even the, algorithm that can not only play the game but compete with human players. The Snake Game is a classic arcade style game where it is a single-player game but the focus is to achieve the highest score possible thus competing with yourself and others.

To play the game one controls a snake by selecting one of the cardinal directions that the snake will move in. In order to score points, you must direct the snake to the food item, there is only one food item in the game at time. The snake then eats the food and increases in length by one square or unit. The game ends when the snake runs into either the boundaries of the play area or itself, the trailing snake body. The domain provides a very interesting problem given that the snake always moves after a green timing delay and the snake continually grows in size. The delay is the feature that really makes the game difficult because if you do not react fast enough the snake will continue moving in the last direction given. This causes the player to try to act as quickly as possible before the snake runs into an obstacle. Also because the snake is constantly trailed by its tail (being the main obstacle) any move taken cannot be undone or immediately back tracked. So if you were to make a wrong turn into a dead end there is no way to reverse that move to back out of the loop.

3.2 PROBLEM STATEMENT

The problem is to design a Snake Game which provides the following functionalities:

- Snakes can move in a given direction and when they eat the food, the length of the snake increases.
- When the snake crosses itself, the game will be over.

- Food will be generated at a given interval.

  The main classes that can be used are:

- Snake
- Cell
- Board
- Game

3.3 PROBLEM DEPICTION

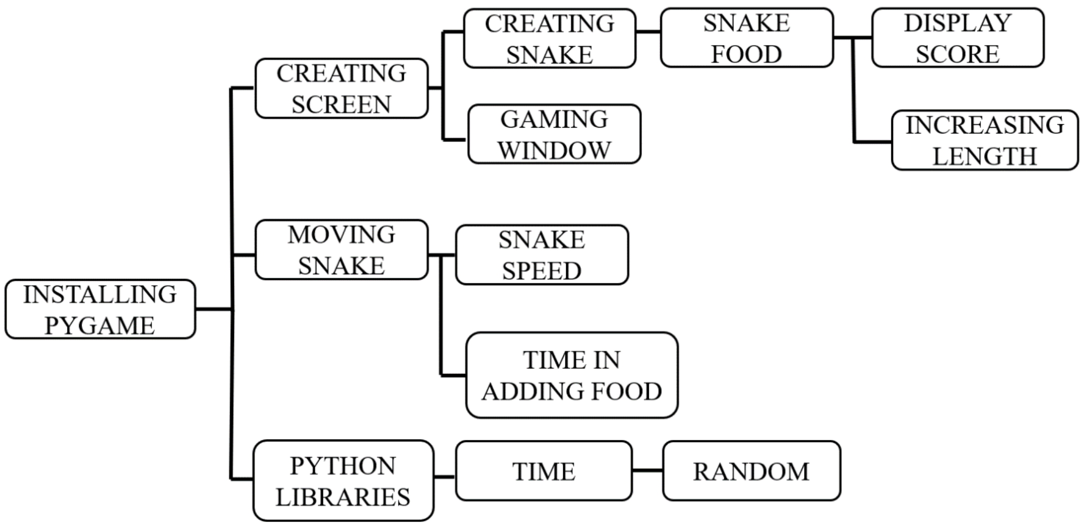


Fig: 3.3 Problem depiction diagram

CHAPTER 4

SOFTWARE REQUIREMENT SPECIFICATION

4.1 SOFTWARE REQUIREMENT SPECIFICATION

Software Requirement Specification (SRS) is the starting point of the software developing activity. As the system grew more complex it became evident that the goal of the entire system cannot be easily comprehended. Hence the need far the requirement phase arose. '[lie software project is initiated by the client. The SRS is the means of translating the ideas of the minds of clients (the input) into a formal document (the output of the requirement phase). The SRS phase consists of the activities.

4.1.1 PROBLEM/REQUIREMENT ANALYSIS

The process is order and more nebulous of the two, deals with understanding the problem, the constraints.

4.1.2 REQUIREMENT SPECIFICATION

Here, the focus is on specifying what has been found giving analysis such as representation specification languages and tools, and checking the specifications are addressed during this. The Requirement phase terminates with the production of the validate SS document. Producing the SRS document is the basic goal of this phase.

4.1.3 ROLE OF SRS

The purpose of the Software Requirement Specification is to reduce the communication gap between the clients and developers. Software Requirement Specification is the medium which makes sure which the client and user needs are accurately specified. It forms the basis of software development. A good SRS should satisfy all the parties involved in the system. The SRS document is the basic goal of this phase. The purpose of the Software Requirement Specification is to reduce the communication gap between the clients and the developers. The developer is responsible for asking for clarifications, where necessary, and will not make alterations without the permission of the client.

4.2 HARDWARE AND SOFTWARE REQUIREMENTS

4.2.1 SOFTWARE REQUIREMENTS

The development and deployment of the application requires the following general and specific minimum requirements for software:

• Programming Language Translaties Python 18

• IDLE - Operating System used Windows 10.

4.2.2 HARDWARE REQUIREMENTS

The development and deployment of the application requires the following general and specific minimum requirements for hardware:

• Processor-(32-bit or 64-bit) RAM (4 GB).

CHAPTER 5

PROJECT DESCRIPTION

5.1 SYSTEM DESIGN

To create a Snake game that allows users to control the movement of a snake on a screen, to get points for eating food and avoiding running into the walls or the growing tail of the snake itself. In this problem, we want to write a game where a graphical representation of a snake moves across the screen. When it encounters a piece of food, the snake grows longer and we gain a point. If it hits the wall we die.

To write this program we are going to need:

1. A way of representing the snake.
2. A way of representing the food.
3. A way to display the score.
4. A way for our instructions to reach the snake.
5. A way to know when we've run into something and died.

Our system is going to involve working with both hardware and software, and so we will need to understand what we have available in hardware that can assist us.

If we build our software so that the snake is controlled by directional arrows on the keyboard. Now that understand how our hardware will work in the design of our system, let’s move on to starting the design of our software system.

5.1.1 SOFTWARE DESIGN

We are going to use an object-oriented approach and provide some detail here. We have to think about the Classes that we want to build, with the associated variables and functions that will make sense for the development.

Let's start by looking at the snake itself, the hero of the game. The snake has a location on the screen, and contains multiple visual elements, as it can grow, and the snake's head is connected to the rest of the snake and the snake's body follows it around the screen. If the snake "eats" food, it grows. The snake moves in a very precise way. Based on what the user types, the snake will move in a given direction. Every time the snake moves, the head will go in the new direction, and every piece of the snake will move up, by occupying the space that was formerly occupied by the piece in front of it.

To grow in size, the snake has to eat food. How can we show the snake eating? The simplest answer is that if the head of the snake and the food are in the same place, we consider that the snake eats the food. This means that we have to know where the food is. When it's eaten, it disappears, the snake grows and the food shows up somewhere else. But we also wanted to show a score, so we need a variable to keep track of that as well. In this case, we'll create a Scoreboard class where we can increase the counter value and display it.

5.2 MODULE DESCRIPTION

Snake game is a classic arcade game where the player controls a snake that moves around a grid or a game board. The objective is to eat food items placed randomly on the board, which causes the snake's length to increase. As the snake grows longer, the game becomes more challenging because the player must navigate the snake without running into its own body or the boundaries.

5.2.1 MODULES

Here is a description of the different modules typically involved in the snake game:

Game Board/Grid: This module represents the game board or grid on which the snake moves. It is typically a two-dimensional array or matrix with rows and columns.

Snake: The snake module handles the movement and behavior of the snake. It maintains the snake's current position, direction, and length.

Food: The food module generates and manages the food items on the game board. It randomly places food items on empty cells and ensures they do not overlap with the snake's body or existing food items.

User Input: This module handles user input, allowing the player to control the snake's direction. It detects keyboard or other input events and updates the snake's direction accordingly.

Score: The score module keeps track of the player's score, which typically corresponds to the number of food items the snake has eaten. It provides functions to increment the score and display it to the player.

Game Loop: The game loop module controls the flow of the game. It repeatedly updates the game state, handles user input, and restart the game.

These modules work together to create an interactive snake game experience.

5.3 DATA FLOW DIAGRAM

DFD Diagram, the data flow diagram represents the relationship between the components.

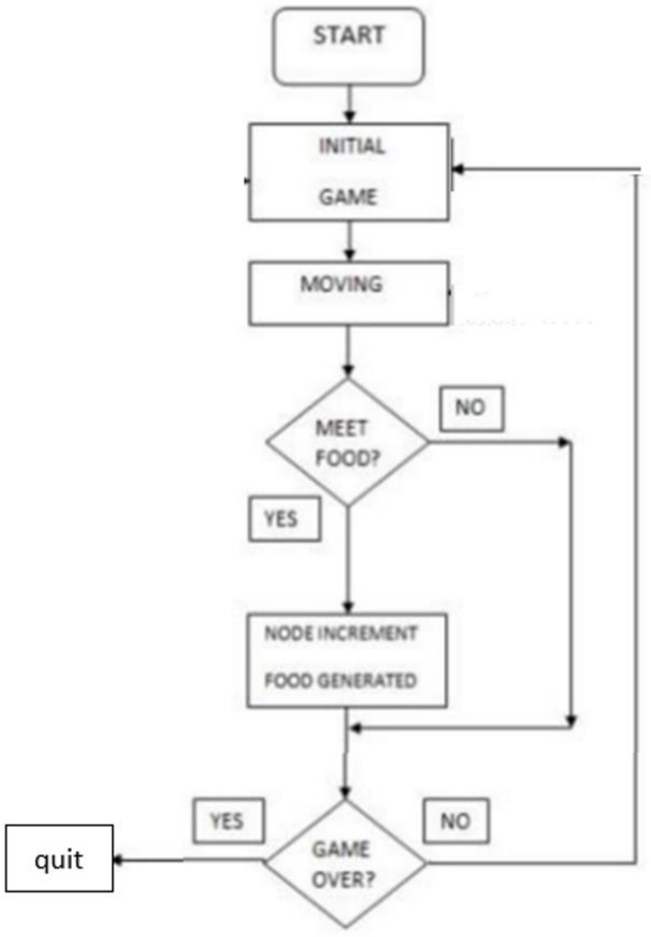


Fig: 5.3 Data flow diagram

5.4 UML DIAGRAMS

USE CASE DIAGRAM

A use case diagram is a diagram that shows a set of use cases and actors and relationships.

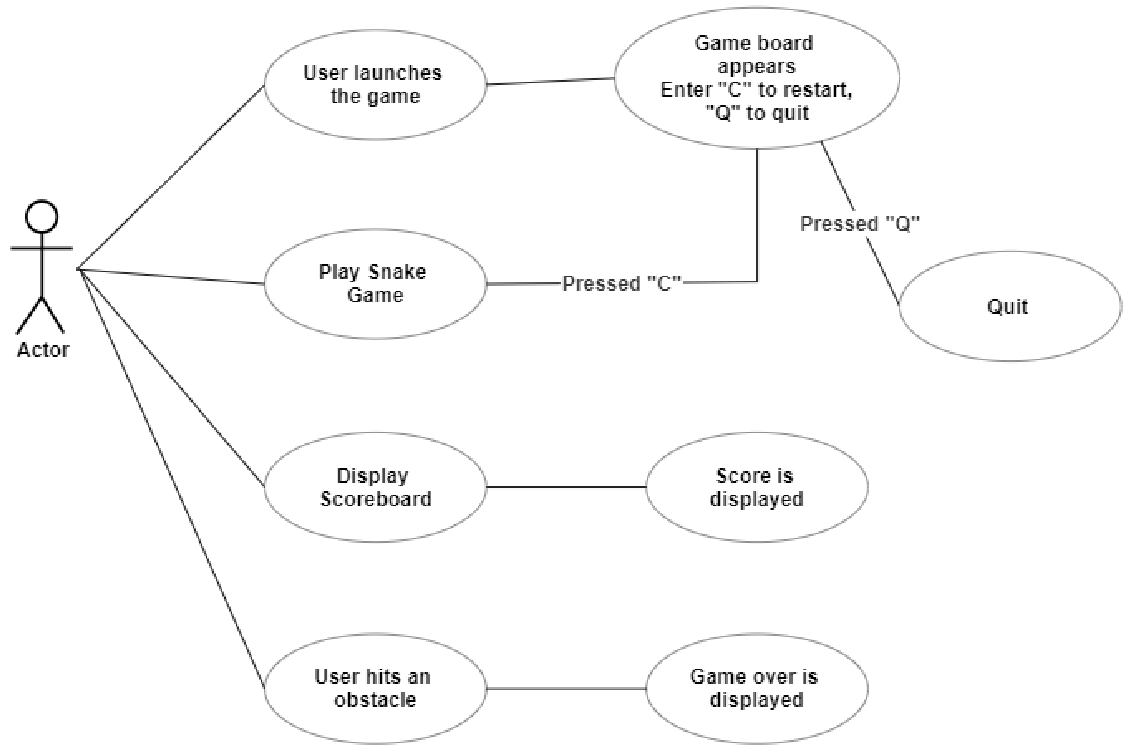


Fig: 5.4.1 Use case diagram

CLASS DIAGRAM

A class diagram represents the structure of the system. It shows set of classes, interfaces, and relationships between them.

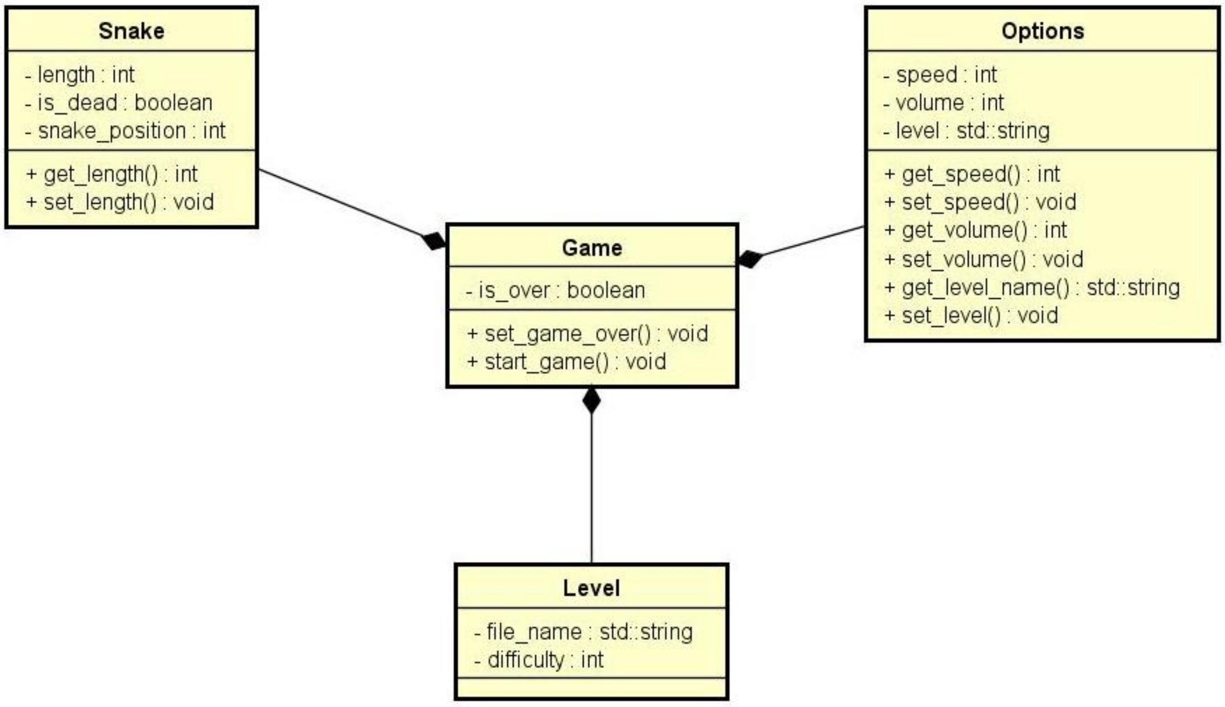


Fig: 5.4.2 Class diagram

ACTIVITY DIAGRAM

An activity diagram shows the flow from activity to activity. An activity is an ongoing non-atomic execution within a state machine.

Activities ultimately result in some action, which is made up of executable atomic computations that result in a change in state of the system or the return of a value.

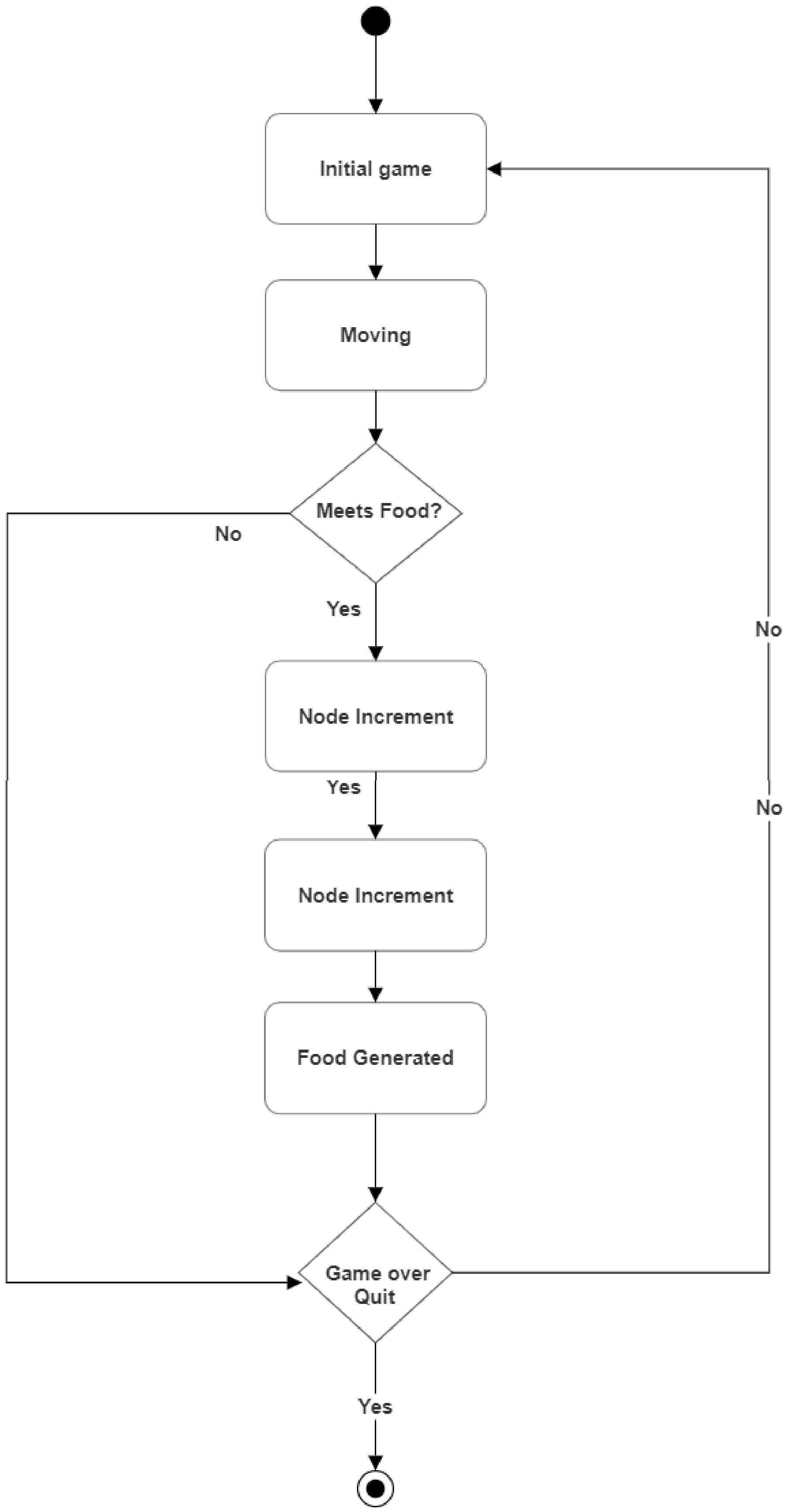


Fig: 5.4.3 Activity diagram

SEQUENCE DIAGRAM

- An interaction diagram shows an interaction, consisting of a set of objects and their relationships, including the messages that may be dispatched among them.
- A sequence diagram is an interaction diagram that emphasizes the time ordering of messages.
- Graphically, a sequence diagram is a table that shows objects arranged along x- axis and messages, ordered in increasing time, along the y-axis.

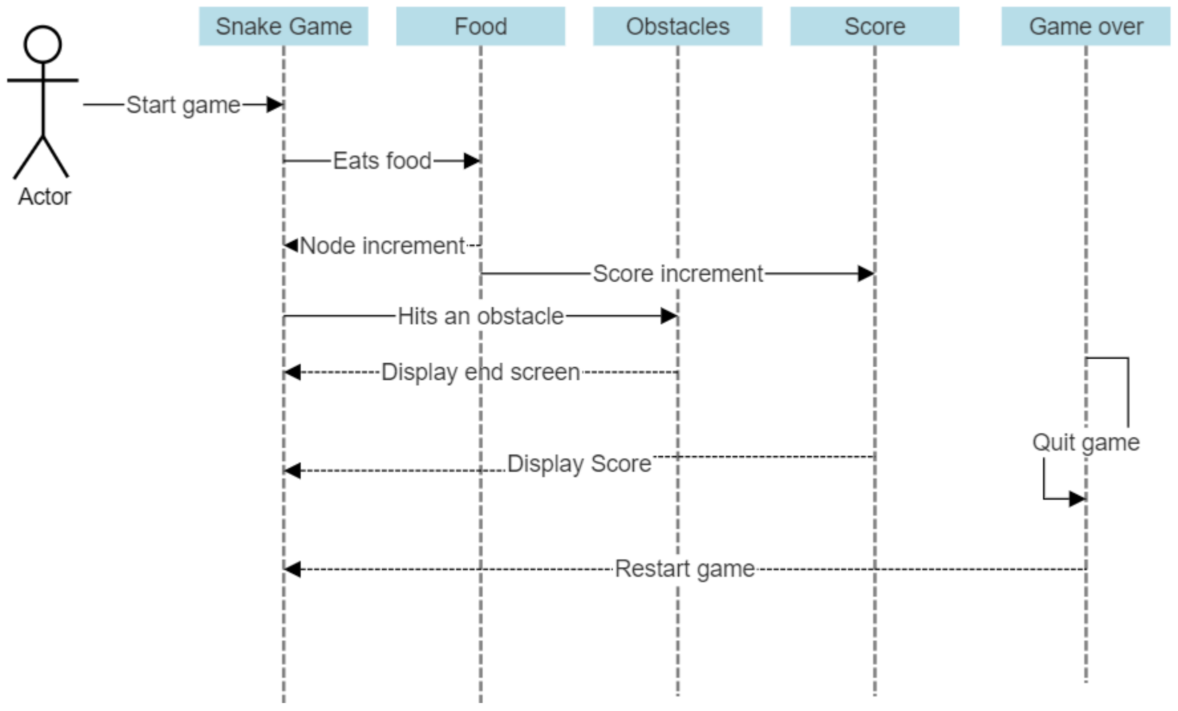


Fig: 5.4.4 Sequence diagram

COMPONENT DIAGRAM

Component is a physical Part of a system that conforms to and provides realization of set of interfaces. A component is a self-contained unit that encapsulates state and behaviour of various set of classifiers. A component provides set of classes and interfaces with some functionality and GUI interfaces which may be required to in several services.

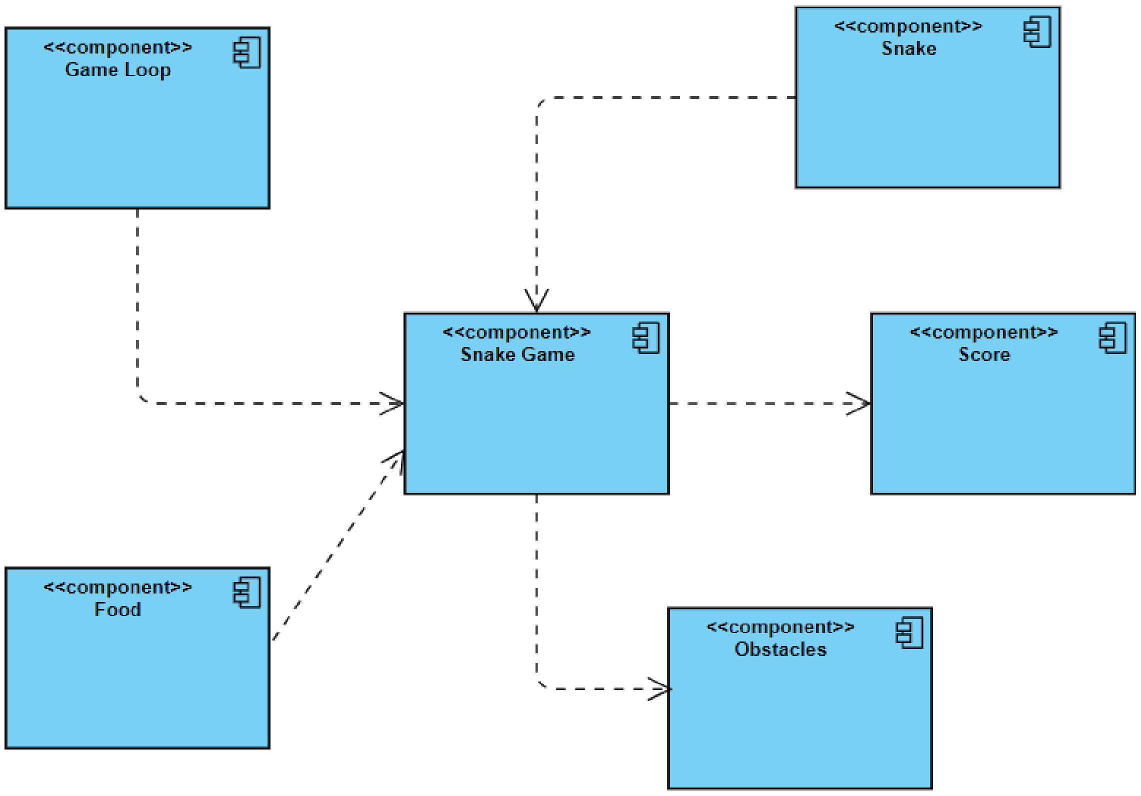


Fig: 5.4.5 Component diagram

DEPLOYMENT DIAGRAM

- A deployment diagram is a diagram that shows the configuration of run time processing nodes and the components that live on them.
- Graphically, a deployment diagram is collection of vertices and arcs.

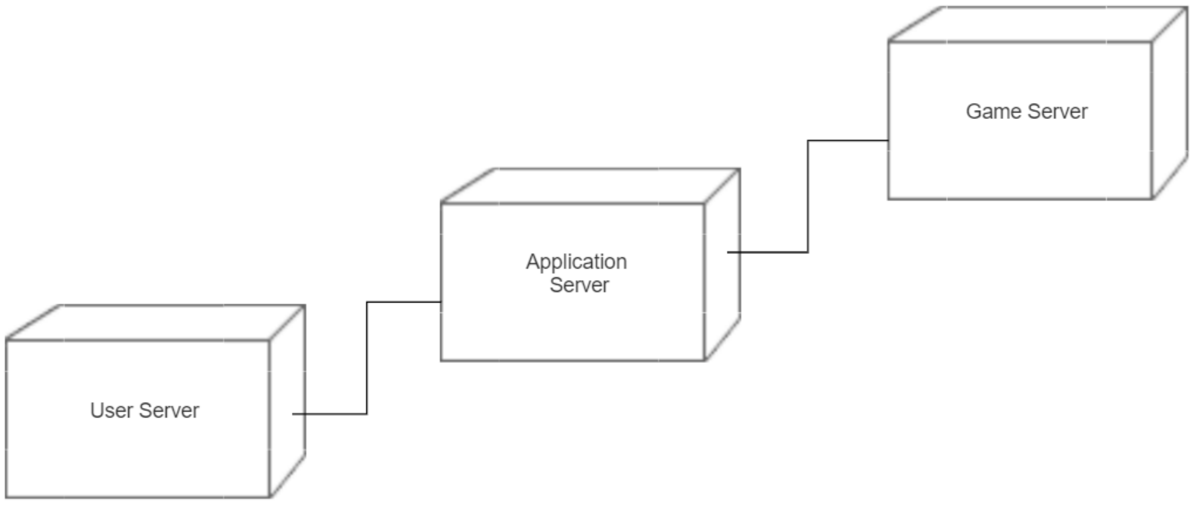


Fig: 5.4.6 Deployment diagram

CHAPTER 6

SYSTEM TESTING

6.1 SOFTWARE TESTING

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and code generation

Testing Objectives

- To ensure that during operation the system will perform as per specification. To make sure that system meets the user requirements during operation
- To make sure that during the operation, incorrect input, processing and output will be detected
- To see that when correct inputs are fed to the system the outputs are correct
- To verily that the controls incorporated in the same system as intended
- Testing is a process of executing a program with the intent of finding an error
- A good test case is one that has a high probability of finding an as yet undiscovered

The software developed has been tested successfully using the following testing strategies and any errors that are encountered are corrected and again the part of the program or the procedure of function is put to testing until all the errors are removed. A successful test is one that uncovers an as yet undiscovered error.

Note that the result of the system testing will prove that the system is working correctly, it will give confidence to system designer, users of the system.

6.2 TEST CASE DESIGN

White Box Testing:

White box testing is a testing ease design method that uses the control structure of the procedure design to derive test eases. All independent paths in a module are exercised at least once. I logical decisions are exercised at once, execute all loops at boundaries and within [heir operational bounds exercise internal data structure to ensure their validity. Here the customer is given three chances to enter a valid choice out of the given menu. After which the control exits the current menu.

Black Box Testing:

Black Box Testing attempts to find errors in following areas or categories, incorrect or missing. functions Interface error, errors In data structures, performance error and initialization and termination error. Here all the input data must match the data type to become a valid entry.

Unit Testing:

Unit testing is essentially for the verification of the code produced during the coding phase and the goal is to test the internal logic of the module program. In the Genetic code project, the unit testing is done during the coding phase of data entry forms whether the functions are working properly or not. In this phase all the drivers are tested whether they are rightly connected or not.

Integration Testing:

All the tested modules are combined into sub systems, which are then tested. The goal is to see if the modules are properly integrated, and the emphasis being on the testing interfaces between the modules. In the genetic code integration testing is done mainly on table creation modules and insertion modules.

Validation Testing:

This testing concentrates on confirming that the software is error-free in all respects. All the specified validations are verified and the software is subjected to hard cone testing. It also aims at determining the degree of deviation that exists in the software designed from the specification, they are listed out and are corrected.

System Testing:

This testing is a series of different tests whose primary is to fully exercise the computer-based system. This involves: Implementing the system in a simulated production environment and testing it. Introducing errors and testing for error handling.

6.3 TEST CASES

How can we test a full Snake game and, assuming it passes that stage, how can we playtest that? The functional requirements that we developed turn, almost immediately.

To display the snake, the first thing we want to do is to make sure that we can draw the snake and move it around on the screen. So our testing for correct function will be:

1. Can I display the snake's head on the screen?
2. Will it move around as I want it to using keyboard control?
3. Is it displaying correctly?
4. Is the body moving correctly?
5. If we identify an error in the snake, because it’s a class, we will go into the snake class and fix it there. However, because we've written the Food and Scoreboard as separate classes, whatever we do in the Snake class shouldn't break anything in there, unless we accidentally change the code without noticing. The next step for the snake will be checking what happens when the head is detected as colliding with something. Does it grow when it eats cat food? Does it die when it hits a wall or itself! We'd then continue to test the program until we've tested all of the individual elements and their interactions together.
6. One useful test case is to see if everything is being drawn where you expect. Because we aren't.
7. Using all the screen, it's possible to draw the food or the snake so that it overlaps the black rectangle that's the boundary. Has the programmer put the correct limits on the ranges where the snake and the food can appear?

CHAPTER 7

SYSTEM IMPLEMENTATION

A development environment refers to the mix of software tools, methods, and physical resources that an IT (Information Technology) team uses to create an information system. Implementation is the stage where the theoretical design is turned into a working system. The most crucial stage in achieving a new successful system and in giving confidence on the new system for the users that it will work efficiently and effectively.

The system can be implemented only after thorough testing is done and if it is found to work according to the specification. It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the change over and an evaluation of change over methods a part from planning. Two major tasks of preparing the implementation are education and training of the users and testing of the system.

The implementation phase comprises of several activities. The required hardware and software acquisition is carried out. The system may require some software to be developed. For this, programs are written and tested. The user then changes over to his new fully tested system and the old system is discontinued.

PYTHON

Python is an object-oriented, high level language, interpreted, dynamic and multipurpose programming language.

Python is easy to learn yet powerful and versatile scripting language which makes it attractive for Application Development.

Python supports multiple programming patterns, including object oriented programming, imperative and functional programming or procedural styles Python is not intended to work on special areas such as web programming. That is why it is known as multipurpose because it can be used with web, enterprise, 3D CAD ere We don't need to use data types to declare variables because it is dynamically typed so we can write a=l0 to declare an integer value in a variable. Python makes the development and debugging fast because there is no compilation step included in python development and the edit-test-debug cycle is very fast.

Python History

Python laid its foundation in the late 1980s. The implementation of Python was started in December 1989 by Guido Van Rossum CWI in the Netherlands.

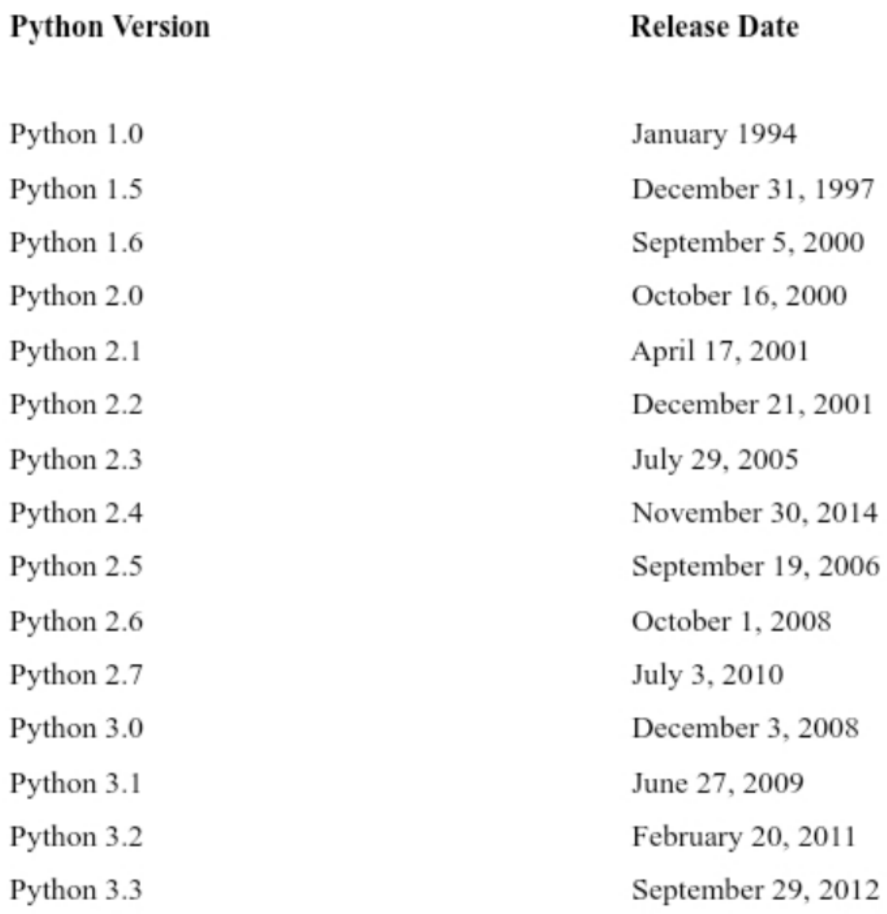
ABC programming language is said to be the predecessor of Python language which was capable of Exception Handling and interfacing with Amoeba Operating System.

Python is influenced by programming languages like

ABC language Modula-3

Python Versions

Python programming language is being updated regularly with new features and support. There are a lot of updates in python versions, from 1994 to the current date. A list of python versions with its date is given below:



Python Version	Release Date
Python 1.0	January 1994
Python 1.5	December 31, 1997
Python 1.6	September 5, 2000
Python 2.0	October 16, 2000
Python 2.1	April 17, 2001
Python 2.2	December 21, 2001
Python 2.3	July 29, 2005
Python 2.4	November 30, 2014
Python 2.5	September 19, 2006
Python 2.6	October 1, 2008
Python 2.7	July 3, 2010
Python 3.0	December 3, 2008
Python 3.1	June 27, 2009
Python 3.2	February 20, 2011
Python 3.3	September 29, 2012

PY-GAME

Python is the most popular programming language or nothing wrong to say that it is the next generation programming language.

In every emerging field in computer science, Python makes its presence actively Python has vast libraries for various fields such as Machine Learning (Numpy, Pandas, Matplotih), Artificial intelligence (Pytorch, TensorFlow), and Game development.

Python's syntax and dynamic typing with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas.

Game programming is very rewarding nowadays and it can also be used in advertising and as a teaching tool too. Game development includes mathematics, logic, physics, AL, and much more and it can be amazingly fine.

In python, game programming is done in pygame and it is one of the best modules for doing so Pygame is a cross-platform set of Python modules which is used to create video games. It consists of computer graphics and sound libraries designed to be used with the Python programming language.

Pygame was officially written by Pete Shinners to replace PySDL. Pygame is suitable to create client-side applications that can be potentially wrapped in a standalone executable.

Installing pygame:

The first thing you will need to do in order to create games using Pygame is to install it on your systems. To do that, you can simply use the following command:

```
pip install pygame
```

7.1 FUNCTION DESCRIPTION

Once that is done, just import Pygame and start our game development. Before moving on, take a look at the Pygame functions that have been used in this Snake Game along with their descriptions.

Function	Description
init()	Initializes all of the imported Pygame modules (returns a tuple indicating success and failure of initializations)
display.set_model()	Takes a tuple or a list us as parameter to create a surface (tuple preferred)
update()	Updates the screen
quit()	Used to un-initialize everything
set caption()	Will set the caption test on the top of the display screen.
event.get()	Returns list of all events
Surface fill()	Will fill the surface with a solid color
time Clock()	Helps truck time
font SysFont()	Will create a Pygame font from the System font resources.

Table 7.1: Function description table

7.2 IMPLEMENTATION STEPS

l. Installing pygame

2. Create the screen

3. Create the Snake

4. Moving the Snake

5. Game over when snake hits the boundaries

6. Adding the Food

7. Increasing the length of the snake & Displaying the score

7.3 WORKING ALGORITHM

Let's look at how a program to run the whole game might look.

l. Draw the playing area with bounding rectangle, set the counter to zero and display 1.

2. Draw the snake in a random position.

3. Draw the food in a random location.

4. On user input, change snake direction.

5. Move the snake one move.

6. If the snake is over food, eat it, increase the score, grow, move the food.

7. else if the snake is over in a wall, die.

8. Go back to step 4.

9. Until the snake die.

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

8.1 CONCLUSION

We are hereby conclude that the project “**Classic Snake Game Implementation Using Python**” is a simple console application with very simple graphics. In this project, you can play the popular "Snake Game" just like you played it elsewhere. You have to use the up, down, right, or left arrows to move the snake.

Foods are provided at the several coordinates of the screen for the snake to eat. Every time the snake eats the food. its length will be increased by one element along with the score. It isn't the world's greatest game, but it does give you an idea of what you can achieve with relatively simple python programming, and perhaps the basis by which to extend the principles and create more interesting games an your own.

8.1.1 LIMITATIONS:

- The existing system only provides a text-based interface, which is not as user-friendly as Graphical user Interface.
- Since the system is implemented in Manual, the response is very slow.
- The transactions are executed in off-line mode, hence on-line data capture and modification is not possible.

8.2 FUTURE ENHANCEMENT:

In this project, I have used a simple application. This project will be able to be implemented in future after making some changes and modifications as I made this project at a low level. The modifications that can be done in this project are:

- It can be made with good graphics.
- We can add more options like Top scores and Player Profile.
- We can add multiplayer option.

CHAPTER 9

APPENDIX

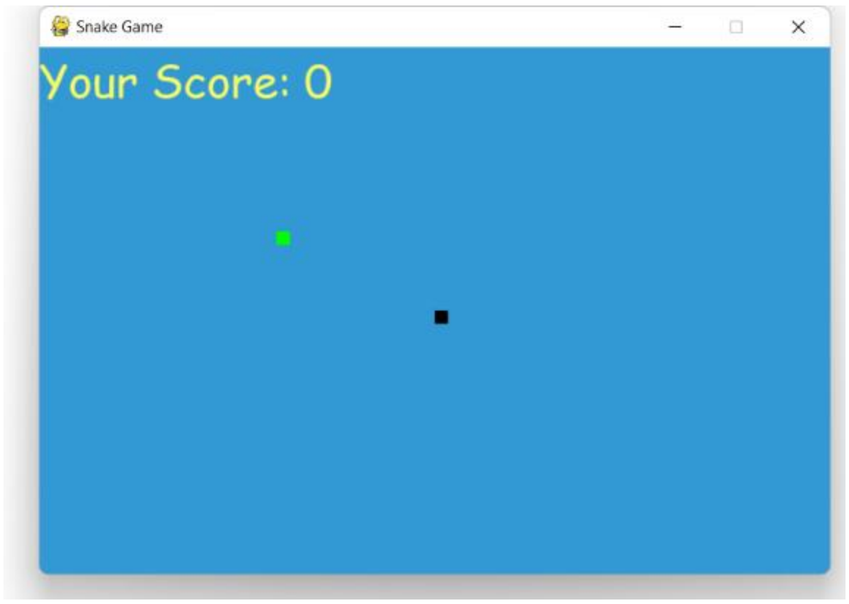
9.1 SOURCE CODE:

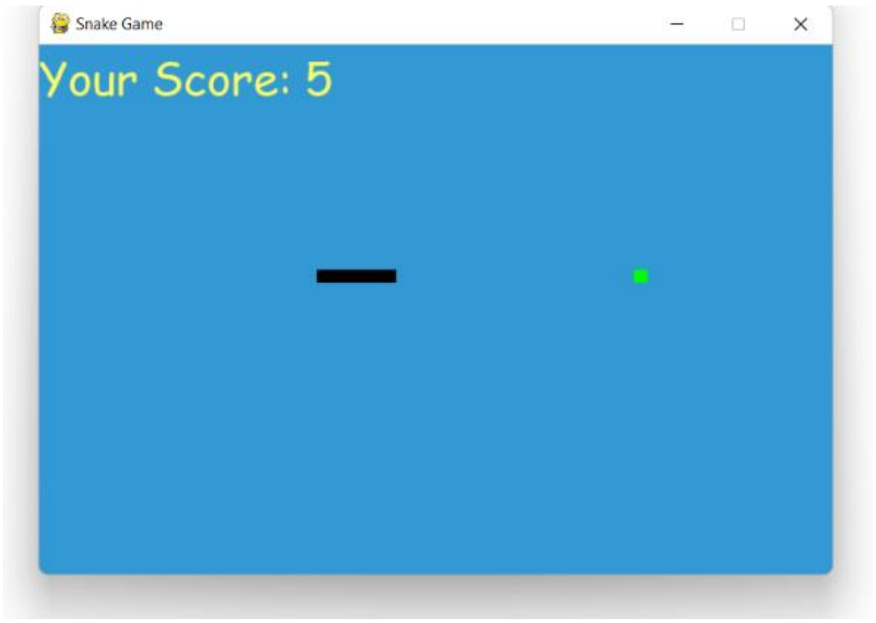
```
import pygame
import time
import random

pygame.init()

white = (255, 255, 255)
yellow = (255, 255, 102)
black = (0, 0, 0)
red = (213, 50, 80)
green = (0, 255, 0)
blue = (50, 153, 213)

dis_width = 600
dis_height = 400

dis = pygame.display.set_mode((dis_width, dis_height))
pygame.display.set_caption('Snake Game')

clock = pygame.time.Clock()

snake_block = 10
snake_speed = 10

font_style = pygame.font.SysFont("bahnschrift", 25)
score_font = pygame.font.SysFont("comicsansms", 35)


def Your_score(score):
    value = score_font.render("Your Score: " + str(score), True, yellow)
    dis.blit(value, [0, 0])



def our_snake(snake_block, snake_list):
    for x in snake_list:
        pygame.draw.rect(dis, black, [x[0], x[1], snake_block, snake_block])


def message(msg, color):
    mesg = font_style.render(msg, True, color)
    dis.blit(mesg, [dis_width / 6, dis_height / 3])


def gameLoop():
    game_over = False
    game_close = False

    x1 = dis_width / 2
    y1 = dis_height / 2

    x1_change = 0
    y1_change = 0

    snake_List = []
    Length_of_snake = 1

    foodx = round(random.randrange(0, dis_width - snake_block) / 10.0) * 10.0
    foody = round(random.randrange(0, dis_height - snake_block) / 10.0) * 10.0

    while not game_over:

        while game_close == True:
```

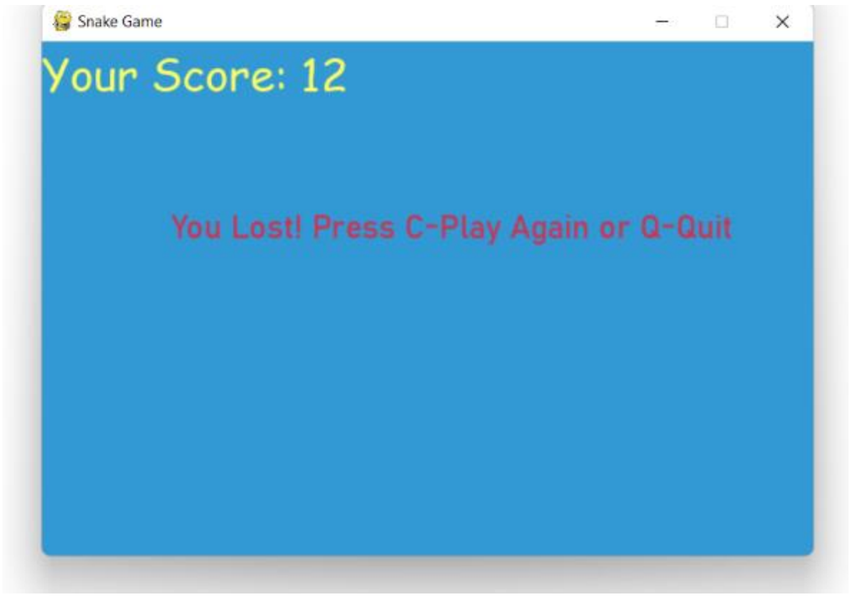
```
    dis.fill(blue)
    message("You Lost! Press C-Play Again or Q-Quit", red)
    Your_score(Length_of_snake - 1)
    pygame.display.update()

    for event in pygame.event.get():
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_q:
                game_over = True
                game_close = False
            if event.key == pygame.K_c:
                gameLoop()

for event in pygame.event.get():
    if event.type == pygame.QUIT:
        game_over = True
    if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_LEFT:
            x1_change = -snake_block
            y1_change = 0
        elif event.key == pygame.K_RIGHT:
            x1_change = snake_block
            y1_change = 0
        elif event.key == pygame.K_UP:
            y1_change = -snake_block
            x1_change = 0
        elif event.key == pygame.K_DOWN:
            y1_change = snake_block
            x1_change = 0

if x1 >= dis_width or x1 < 0 or y1 >= dis_height or y1 < 0:
    game_close = True
x1 += x1_change
y1 += y1_change
```

```
        dis.fill(blue)
        pygame.draw.rect(dis, green, [foodx, foody, snake_block, snake_block])
        snake_Head = []
        snake_Head.append(x1)
        snake_Head.append(y1)
        snake_List.append(snake_Head)
        if len(snake_List) > Length_of_snake:
            del snake_List[0]

        for x in snake_List[:-1]:
            if x == snake_Head:
                game_close = True

        our_snake(snake_block, snake_List)
        Your_score(Length_of_snake - 1)

        pygame.display.update()

        if x1 == foodx and y1 == foody:
            foodx = round(random.randrange(0, dis_width - snake_block) / 10.0) * 10.0
            foody = round(random.randrange(0, dis_height - snake_block) / 10.0) * 10.0
            Length_of_snake += 1

        clock.tick(snake_speed)

    pygame.quit()
    quit()


gameLoop()
```

9.1 SCREEN SHOTS:

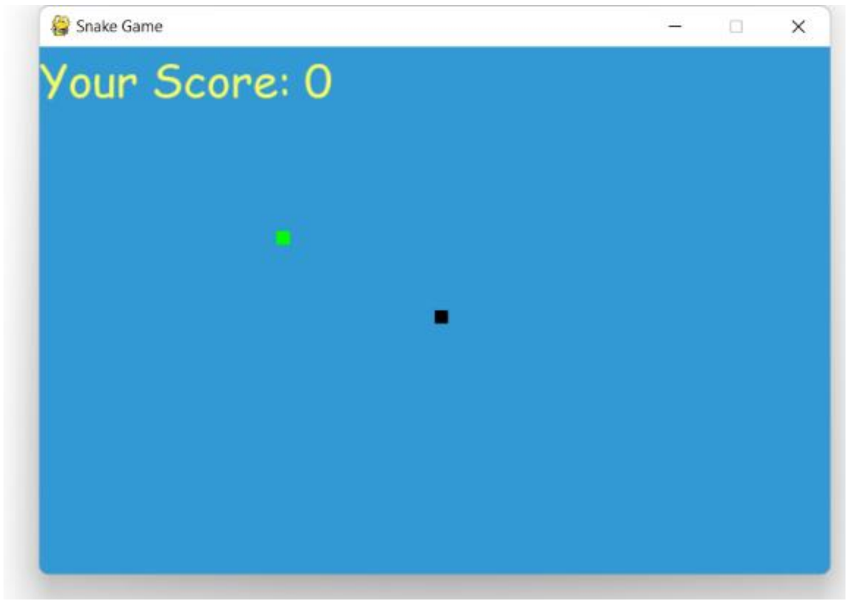





Snake Game

Your Score: 12

You Lost! Press C-Play Again or Q-Quit


Snake Game

Your Score: 0

CHAPTER 10

REFERENCES

https://www.edureka.co/blog/snake-game-with-pygame/

https://pypi.org/project/pygame/

https://yvww.edueba.com/python-pygame/

https://www.w3schools.com/python/module_random.asp

https://www.programmiz.com/python-programming/time