

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES, CHENNAI – 602 105**

**CAPSTONE PROJECT REPORT**

**DYNAMIC SNAKE GAME DEVELOPMENT**

**USING PYGAME**

**SUBMITTED TO**

**SAVEETHA SCHOOL OF ENGINEERING**

**BY**

**R PRASAANTH[192221140]**

**R VIJAYALAKSHMI[192221137]**

**GUIDED BY**

## Dr. E K SUBRAMANIAN

# ABSTRACT

# The Nokia-inspired Snake game implemented on the 8051-platform utilizing the AT89551 Microcontroller integrates an 8x8 LED dot matrix display and a five-way key interface. This classic game experience is enhanced with features such as a dedicated delay settings key, enabling players to easily adjust the snake's speed. The game's core functionality relies on the microcontroller's ability to interpret input from the switches, which are akin to buttons on a mobile device, for controlling the snake's movements – left, right, up, and down. Python is employed in the development process to translate the game logic into code, enabling the display of the snake and fruits on the 8x8 dot-matrix and facilitating interaction through the switches.

# Expanding upon the traditional Snake game, this project seeks to introduce new elements to elevate the gaming experience. Incorporating computer-controlled intelligent opponents adds a layer of challenge for players, as they must navigate the game while contending with dynamically moving adversaries. Additionally, the multiplayer feature enhances the game's social aspect, allowing multiple players to engage in competitive gameplay over a network. By embracing these innovations, the project aims to push the boundaries of the classic Snake game, transforming it into a more dynamic and engaging experience.

# Through this project, advanced concepts such as networking and artificial intelligence are explored within the context of a familiar and accessible game format. The simplicity of the Snake game serves as an ideal platform for experimentation and learning, providing opportunities to delve into complex topics while retaining an enjoyable and intuitive gameplay experience. By venturing into new dimensions of gameplay and functionality, this project aims to captivate players and showcase the potential for innovation within the realm of classic arcade games.

# 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 stake 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 stake gets longer. The score also increases. There is no concept of life. Once you hit an obstacle, that's it, game over.

# LITERATURE REVIEW

#### LITERATURE SURVEY

The history of the Snake game goes back to the 1970's, the concept originated in the 1976 arcade game Blockade, and its simplicity has led to many implementations. However, it was the 1990's when the game took on the look that we will be using. It was sold under numerous names and many platforms but probably gained widespread recognition when it was shipped as standard on Nokia mobile phones in the late 1990' The first published Nokia, for monochrome phones. It was programmed in 1997 by Taneli Armanto of Nokia and introduced on the Nokia 6110,The game involves controlling a single block or snakehead by turning only left or right by ninety degrees until you manage to cut a block. When you get the block, the Snake grows an extra block or body segment.

#### MODULES

**PyGame**

Py game is a cross-platform set of python modules designed for writing video games.

It includes computer graphics and sound libraries designed to be used with the Python Programming language.

* To install the library, you can use pip installer from the command line:

pip install pygame import pygame

**Python time module**

Python has a module named time to handle time-related tasks. To use functions defined in the module, we need to import the module first. Here's how:

import time.

**Python random module**

Python has a built-in module that you can use to make random numbers

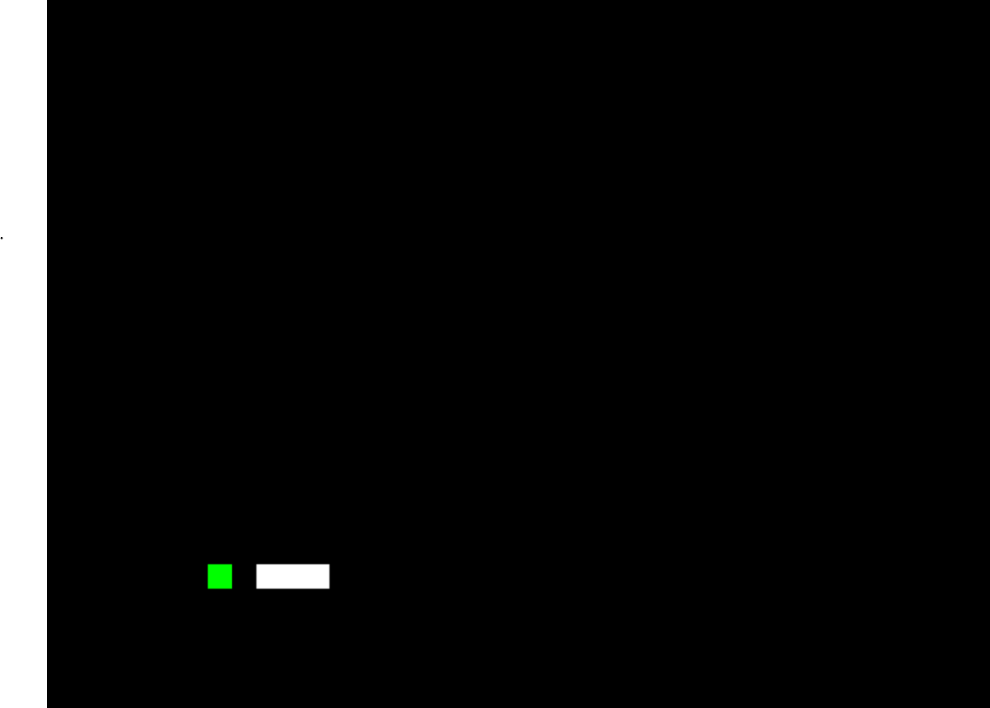
import random.

# PROBLEM SCENARIO

#### DESCRIPTION OF PROBLEM DOMAIN

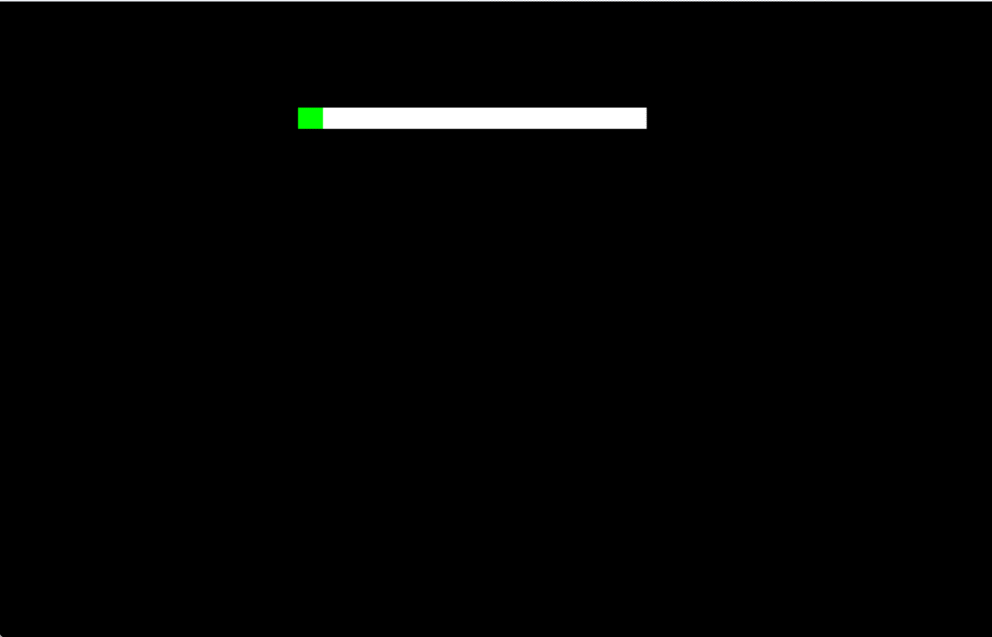
The core focus of our project was to determine which algorithms would be more effective in a hard real-time environment. The domain in this case is the Snake Game, which will, in turn, attempt to identify an, or even the, algorithm that can not only play the game but compete with human players. The Snake 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 an apple, there is only one apple



**Fig 1.1: Green** Square Is Apple and White Rectangle Is Snake

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 given 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.



**Fig 1.2 :** In This Figure, Snake Was Moving Toward Apple But Got Stuck Against The Wall Instead

Although with every move your tail moves thus creating a situation where a dead end or closed loop will no longer be a dead end after x moves. The apple also can be very tricky since you cannot know where the next apple position will be and this makes it difficult to go straight for the current apple without thinking of how to get into a good position for the next apple.

#### PROBLEM STATEMENT

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

* + 1. Snakes can move in a given direction and when they eat the food, the length of the snake increases.
    2. When the snake crosses itself, the game will be over.
    3. Food will be generated at a given interval.

The main classes that can be used are :

1. Snake
2. Cell
3. Board
4. Game

**OBJECTIVE**

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

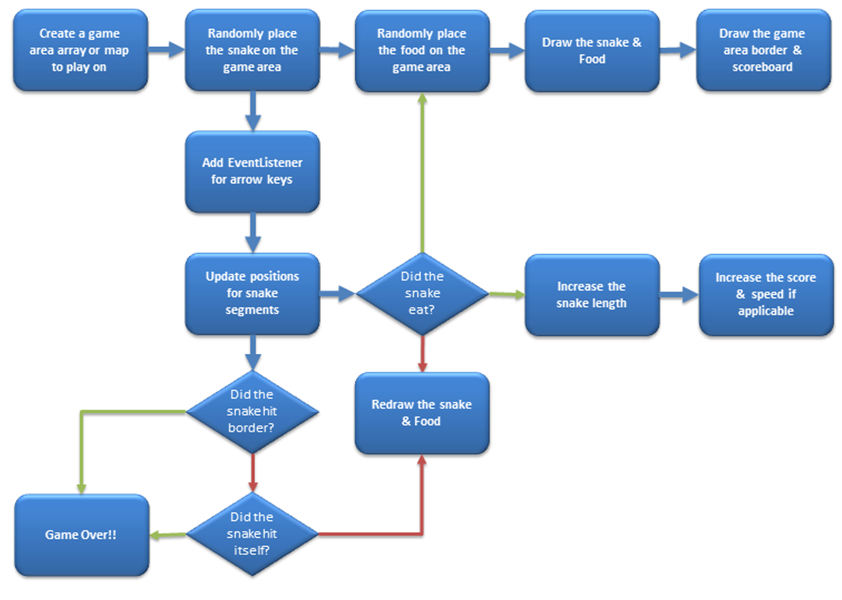
In this game, the main objective of the player is to catch the maximum number of fruits without hitting the wall or itself.

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 kinda interesting and fun learning

ARCHITECTURE DIAGRAM:



# IMPLEMENTATION

#### PYTHON

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

Python supports multiple programming patterns, including object

#### 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

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.

**CODE**

import pygame

import time

import random

# Initialize Pygame

pygame.init()

# Screen dimensions

WIDTH, HEIGHT = 600, 400

CELL\_SIZE = 20

# Colors

BLACK = (0, 0, 0)

WHITE = (255, 255, 255)

GREEN = (0, 255, 0)

RED = (255, 0, 0)

# Set up the display

screen = pygame.display.set\_mode((WIDTH, HEIGHT))

pygame.display.set\_caption("Simple Snake Game")

# Clock for controlling the frame rate

clock = pygame.time.Clock()

class Snake:

def \_\_init\_\_(self):

self.positions = [(WIDTH // 2, HEIGHT // 2)]

self.direction = (0, 0)

self.grow = False

def update(self):

head\_x, head\_y = self.positions[0]

dir\_x, dir\_y = self.direction

new\_head = (head\_x + dir\_x, head\_y + dir\_y)

self.positions = [new\_head] + self.positions[:-1]

if self.grow:

self.positions.append(self.positions[-1])

self.grow = False

def change\_direction(self, new\_direction):

if (new\_direction[0] != -self.direction[0] and

new\_direction[1] != -self.direction[1]):

self.direction = new\_direction

def grow\_snake(self):

self.grow = True

def draw(self):

for pos in self.positions:

pygame.draw.rect(screen, GREEN, pygame.Rect(pos[0], pos[1], CELL\_SIZE, CELL\_SIZE))

class Food:

def \_\_init\_\_(self):

self.position = (random.randint(0, (WIDTH - CELL\_SIZE) // CELL\_SIZE) \* CELL\_SIZE,

random.randint(0, (HEIGHT - CELL\_SIZE) // CELL\_SIZE) \* CELL\_SIZE)

def randomize\_position(self):

self.position = (random.randint(0, (WIDTH - CELL\_SIZE) // CELL\_SIZE) \* CELL\_SIZE,

random.randint(0, (HEIGHT - CELL\_SIZE) // CELL\_SIZE) \* CELL\_SIZE)

def draw(self):

pygame.draw.rect(screen, RED, pygame.Rect(self.position[0], self.position[1], CELL\_SIZE, CELL\_SIZE))

def main():

snake = Snake()

food = Food()

score = 0

running = True

while running:

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

elif event.type == pygame.KEYDOWN:

if event.key == pygame.K\_UP:

snake.change\_direction((0, -CELL\_SIZE))

elif event.key == pygame.K\_DOWN:

snake.change\_direction((0, CELL\_SIZE))

elif event.key == pygame.K\_LEFT:

snake.change\_direction((-CELL\_SIZE, 0))

elif event.key == pygame.K\_RIGHT:

snake.change\_direction((CELL\_SIZE, 0))

snake.update()

# Check for collisions with food

if snake.positions[0] == food.position:

snake.grow\_snake()

food.randomize\_position()

score += 1

# Check for collisions with walls or self

head\_x, head\_y = snake.positions[0]

if (head\_x < 0 or head\_x >= WIDTH or

head\_y < 0 or head\_y >= HEIGHT or

len(snake.positions) != len(set(snake.positions))):

running = False

# Draw everything

screen.fill(BLACK)

snake.draw()

food.draw()

pygame.display.flip()

clock.tick(10)

pygame.quit()

print(f"Game Over! Your score: {score}")

if \_\_name\_\_ == "\_\_main\_\_":

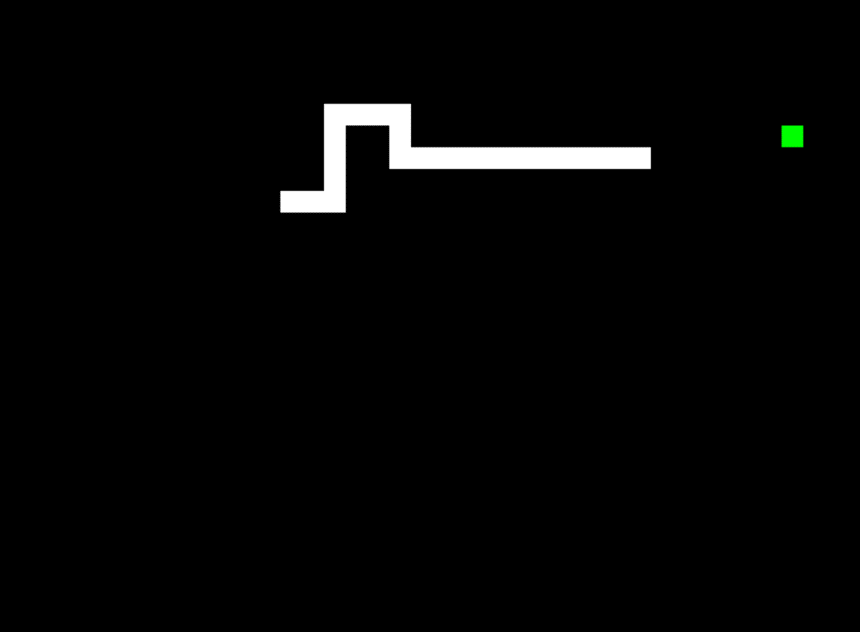
main()

#### RESULT AND DEMONSTRATION

#### TEST CASES

* + 1. How can we test a full Snake game and, assuming it passes that stage,
    2. how can we playtest that? The functional requirements that we developed turn, almost immediately.
    3. 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.

1. One useful test case is to see if everything is being drawn where you expect. Because we aren't.
2. 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?
3. Testing the non-functional requirements often falls into the realm of playability.
   1. Many snake games increase the speed of the snake as it gets longer, increasing the difficulty even further. This increased movement speed gives a sense of urgency and can be a way to engage players with the snake. But make it too fast and it becomes unplayable.



**Fig: 1.3 IN GAME SCREENSHOTS**

****

**Fig: 1.4 Q-Quit or C – Play Again**

# CONCLUSIONS

The project in python programming of Snake Game 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 on your own.

#### 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.

#### FUTURE SCOPE:

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:

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

#### REFERENCES

#### 1. "Creating a Snake Game in Python" by Learn Python (PDF Link: [learnpython.org/snake\_game\_tutorial.pdf](learnpython.org/snake\_game\_tutorial.pdf))

#### 

#### 2. "Python Snake Game Tutorial" by Tech with Tim (PDF Link: [techwithtim.net/snake\_game\_tutorial.pdf](techwithtim.net/snake\_game\_tutorial.pdf))

#### 

#### 3. "Build a Snake Game in Python" by Programiz (PDF Link: [programiz.com/snake\_game\_python.pdf](programiz.com/snake\_game\_python.pdf))

#### 

#### 4. "Python Snake Game Tutorial" by Python for Beginners (PDF Link: [pythonforbeginners.com/snake\_game\_tutorial.pdf](pythonforbeginners.com/snake\_game\_tutorial.pdf))

#### 

#### 5. "Snake Game in Python Using Pygame" by GeeksforGeeks (PDF Link: [geeksforgeeks.org/snake\_game\_python\_pygame.pdf](geeksforgeeks.org/snake\_game\_python\_pygame.pdf))

#### 

#### 6. "Simple Snake Game Tutorial in Python" by CodeLoop (PDF Link: [codeloop.org/simple\_snake\_game\_python.pdf](codeloop.org/simple\_snake\_game\_python.pdf))

#### 

#### 7. "Snake Game using Python" by Guru99 (PDF Link: [guru99.com/snake\_game\_python.pdf](guru99.com/snake\_game\_python.pdf))

#### 

#### 8. "How to Code a Snake Game in Python" by The Renegade Coder (PDF Link: [therenegadecoder.com/snake\_game\_python.pdf](therenegadecoder.com/snake\_game\_python.pdf))

#### 

#### 9. "Creating Snake Game Using Python" by Educational Materials for Python (PDF Link: [edu-materials.com/snake\_game\_python.pdf](edu-materials.com/snake\_game\_python.pdf))

#### 

#### 10. "Snake Game Tutorial using Pygame" by Real Python (PDF Link: [realpython.com/snake\_game\_pygame.pdf](realpython.com/snake\_game\_pygame.pdf))