Machine Learning Project Based Learning (PBL) Report on

SNAKE GAME IMPLEMENTATION

Submitted in partial fulfilment of the

Requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

Computer Science and Engineering (AI&ML)

Submitted by

K Rajkumar 21R11A6622

SK Sohib Arafath 21R11A6640

V Koushik Sharma 21R11A6646

Under the esteemed guidance of

Mr. Shaik Akbar

Associate Professor, CSE(AI&ML) Department

Geethanjali College of Engineering and Technology

Geethanjali College of Engineering and Technology

Department of Computer Science and Engineering (AI&ML)

(UGC AUTONOMOUS INSTITUTION)

Accredited by NAAC with 'A+' Grade & NBA, Approved by AICTE and Affiliated to JNTUH

Cheeryal (V), Keesara (M), Medchal (Dist), Telangana – 501 301.

JANUARY - 2024

Geethanjali College of Engineering and Technology

Department of Computer Science and Engineering (AI&ML)

(UGC AUTONOMOUS INSTITUTION)

Accredited by NAAC with 'A+' Grade & NBA, Approved by AICTE and Affiliated to JNTUH

Cheeryal (V), Keesara (M), Medchal (Dist), Telangana – 501 301.

January – 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

CERTIFIACTE

This is to certify that the Machine Learning Project Based Learning (PBL) Report entitled **"Snake Game Implementation"** is a bonafide work done and submitted by **K Rajkumar (21R11A6622), SK Sohib Arafath (21R11A6640), V Koushik Sharma (21R11A6646)** during the academic year 2022 – 2023, in partial fulfilment of requirement for the award of Bachelor of Technology degree in **"Computer Science and Engineering (AI&ML)"** from Geethanjali College of Engineering and Technology (Accredited by NAAC with 'A+' Grade & NBA, Approved by AICTE and Affiliated to JNTUH), is a bonafide record of work carried out by them under guidance and supervision.

Certified further that to my best of the knowledge, the work in this project has not been submitted to any other institution for the award of any degree or diploma.

FACULTY GUIDE
Mr. Shaik Akbar
Associate Professor
CSE(AI&ML) Department

HEAD OF THE DEPARTMENT
Dr. L. Venkateswarlu
Professor & HOD
CSE(AI&ML) Department

DECLARATION

We hereby declare that the project report entitled **"Snake Game Implementation"** is an original work done and submitted to CSE(AI&ML) Department, from Geethanjali College of Engineering and Technology (Accredited by NAAC with 'A+' Grade & NBA, Approved by AICTE and Affiliated to JNTUH), in partial fulfilment of the requirement for the award of Bachelor of Technology in **"Computer Science and Engineering (AI&ML)"** and it is a record of bonafide project work carried out by us under the guidance of **Mr. Shaik Akbar, Associate Professor, Department of CSE(AI&ML)**.

Signature of the Student
K Rajkumar
(21R11A6622)

Signature of the Student
SK Sohib Arafath
(21R11A6640)

Signature of the Student
V Koushik Sharma
(21R11A6646)

ACKNOWLEDGEMENT

This satisfaction of completing this project would be incomplete without mentioning our gratitude towards all the people who have supported us. Constant guidance and encouragement have been instrumental in the completion of this project.

First and foremost, we thank the chairman, Principal, Vice Principal for availing infrastructural facilities to complete the project in time.

We offer our sincere gratitude to our faculty guide **Mr. Shaik Akbar,** Associate Professor, CSE(AI&ML) Department, from Geethanjali College of Engineering and Technology (Accredited by NAAC with 'A+' Grade & NBA, Approved by AICTE and Affiliated to JNTUH), for his immense support, timely co-operation and valuable advice throughout the course of our project work.

We would like to thank the Head of Department, **Dr. L. Venkateswarlu,** for his meticulous care and co-operation throughout the project work.

We are thankful to all the project Coordinators for their supportive guidance and for having provided the necessary help for carrying forward this project without any obstacle and hindrances.

ABSTRACT

The Snake game implementation in Python using the Pygame library combines classic gaming elements with modern programming techniques. The game revolves around guiding a snake, controlled by the player through arrow keys, to consume randomly generated apples, leading to the snake's growth and an increment in the player's score. Key features include intuitive controls, responsive collision detection mechanisms to prevent wall and self-intersections, and a pause/resume functionality to enhance user experience. The Pygame library facilitates the creation of a visually engaging graphical interface, utilizing various colors and fonts. The project promotes fundamental game development concepts, such as event handling, game loops, and dynamic rendering. Players can initiate a new game session, pause or resume gameplay at will, and are presented with a game-over screen displaying their final score upon completion. The Snake game project offers a practical and enjoyable introduction to Python game development, illustrating the implementation of essential gaming mechanics within a well-designed and interactive gaming environment.

Keywords: pygame library , classic gaming , programming techniques , arrow keys , intuitive controls , wall and elf-intersections , resume functionality , graphical interface , dynamic rendering , gaming mechanics , interactive environments .

LIST OF FIGURES

TABLE OF CONTENTS

1. INTRODUCTION

The Snake Game project is a classic and engaging Python-based gaming application developed using the Pygame library. This timeless arcade-style game provides players with a nostalgic experience reminiscent of early digital entertainment. Through intuitive controls utilizing arrow keys, players navigate a snake across the screen, aiming to consume apples strategically placed within the gaming environment. As the snake devours apples, it dynamically grows, adding an exciting dimension to the challenge.

The project incorporates fundamental programming techniques to create an interactive and visually appealing gaming experience. The implementation involves meticulous collision detection algorithms to handle interactions between the snake, apples, and game boundaries. Users are presented with a clear graphical interface that includes vibrant colors, diverse fonts, and an informative display of their current score.

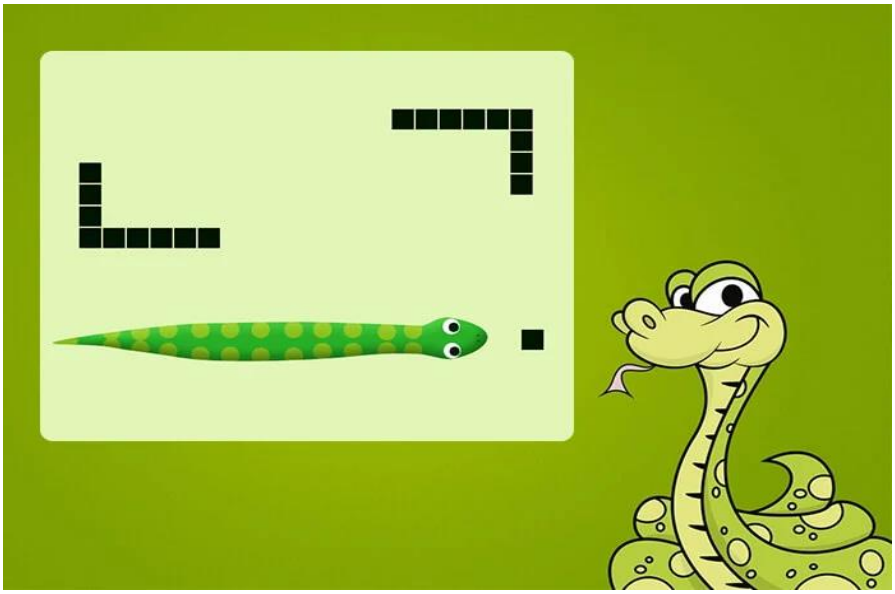


Fig. 1.1.1 Snake Game

1.1 Key Features

Key features of the Snake Game project include responsive event handling, allowing seamless player control, and a game loop that ensures continuous and fluid rendering. The application introduces dynamic elements such as pause/resume functionality, enhancing the user experience and providing flexibility during gameplay. Users are prompted with a game-over screen upon the snake's demise, displaying the final score and offering options to either exit the game or initiate a new session.

2. AIM and OBJECTIVES

2.1 Aim of Proposed Project

The aim of the Snake Game project is to develop a classic arcade-style game using Python and the Pygame library, providing an engaging and entertaining experience for players. This project serves as a practical application to reinforce programming concepts and enhance skills in game development.

2.2 Objectives of Proposed Project

Implement Classic Gameplay: Create the fundamental mechanics of the Snake Game, including snake movement, apple generation, and dynamic growth upon apple consumption.

Utilize Pygame Library: Leverage the Pygame library for graphical rendering, event handling, and overall game development, allowing for a visually appealing and interactive user interface.

Incorporate Responsive Controls: Implement responsive controls using arrow keys to allow players seamless navigation of the snake within the game environment.

Ensure Collision Detection: Develop accurate collision detection algorithms to handle interactions between the snake, apples, and game boundaries, preventing glitches and ensuring smooth gameplay.

Introduce Dynamic Elements: Enhance the gaming experience by introducing dynamic elements such as a pause/resume feature, providing flexibility and convenience for players during gameplay.

Create Informative Display: Implement a clear and informative display that includes the player's score, offering feedback on their performance throughout the game.

Handle Game Over Scenario: Develop a game-over screen that triggers upon the snake's demise, displaying the final score and providing options for users to either exit the game or start a new session.

Educational Value: Serve as an educational tool for developers, offering insights into game development concepts, graphical interfaces, event-driven programming, and the integration of external libraries.

Balance Challenge and Enjoyment: Strike a balance between challenging gameplay and entertainment, ensuring the project appeals to both novice and experienced gamers.

Encourage Further Exploration: Motivate developers and learners to explore and modify the project, fostering creativity and expanding their understanding of game development.

3.SOFTWARE REQUIRMENT SPECIFICATIONS

3.1. Software requirements

For developing the application the following are the Software Requirements

1. **Python 3**

Fig. 3.1.1 Python 3 Logo

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently whereas the other languages use punctuations. It has fewer syntactical constructions than other languages like it is Interpreted, Interactive, Object-Oriented, Beginners Language. In this project we use python language to develop the web application.

2. **Pygame Library:**

Fig 3.1.2 Pygame Library Logo

Pygame is used for graphical rendering and handling input in the Snake Game.Pygame is a free and open-source cross-platform library for the development of multimedia applications like video games using Python. It uses the Simple Direct Media Layer library and several other popular libraries to abstract the most common functions, making writing these programs a more intuitive task

3. PYCHARM

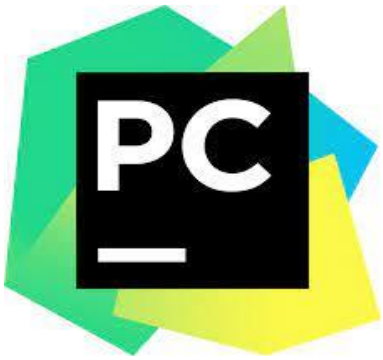


Fig. 3.1.3 Pycharm Logo

PyCharm is an Integrated Development Environment (IDE) for the Python programming language. It is developed by JetBrains and is available in both free and paid versions. PyCharm provides a range of features to support Python development, including code editing, debugging, testing, and code analysis.

4. VISUAL STUDIO CODE

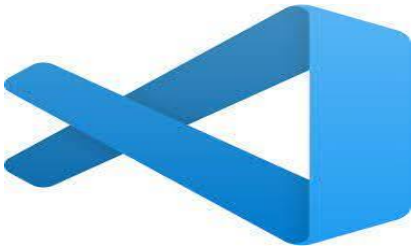


Fig. 3.1.4 Visual Studio Code Logo

Visual Studio Code (VS Code) is a free and open-source code editor developed by Microsoft. It has gained immense popularity among developers due to its lightweight nature, extensibility, and powerful features. Here are some key aspects of Visual Studio Code:
VS Code is available for Windows, macOS, and Linux, making it a cross-platform code editor.
Unlike traditional integrated development environments (IDEs), VS Code is designed to be lightweight and responsive. It starts quickly and consumes fewer system resources.
While it started as a code editor primarily for JavaScript, TypeScript, and web development, VS Code has evolved to support a wide range of programming languages. It has built-in support for languages like Python, Java, C++, and more.

4.IMPLEMENTATION

Initialization:
The script starts by importing necessary modules: random, math, pygame, and sys.
Constants and variables are defined, including window size, colors, block size, directions, and game state flags.

Game Loop:
The main function contains the main game loop, which runs continuously.
The loop handles user input, game logic, and rendering.

Start Screen:
At the beginning of the game, a start screen is displayed with instructions for the player.
The snake and an apple are drawn on the screen, and a welcome message is displayed.

Input Handling:
The script captures user input using the Pygame event system.
Arrow keys control the snake's direction, 'q' quits the game, 'p' pauses the game, and 'r' resumes the game.

Game Logic:
The snake's movement is regulated by a clock to control the speed of the game.
The snake cannot reverse its direction instantly to avoid colliding with itself.
If the snake hits the wall or collides with itself, the game ends.

Apple Generation:
An apple is randomly generated on the screen. The game ensures that the apple does not appear in the snake's position.

Scoring:
The player scores points by eating apples.
The snake grows longer each time it eats an apple.

Rendering:
The screen is cleared, and the game borders are drawn.
The snake and apple are drawn on the screen.
The current score is displayed at the top of the screen.

Game Over Screen:
When the snake dies, a game over screen is displayed with the player's score.
Options to quit or play again are provided.

Event Handling (Post-Game):
After the game over screen, the script waits for user input ('q' to quit or 'n' to play again).

Clock Regulation:
The clock regulates the speed of the game, preventing the snake from moving too quickly.

Pygame Initialization:

Pygame is initialized, and a window is created for the game.

Execution:

The script checks if it is the main module and calls the main function to start the game.

Note: The implementation assumes that the Pygame library is installed in the Python environment. If not, it can be installed using the command pip install pygame. The code structure and comments make the implementation relatively easy to understand and modify.

5. SAMPLE CODE

```
#####################################################################

######### IMPORTS ###################################################

import random, math, pygame,sys
from pygame.locals import *

counter = 0

######### MAIN ######################################################

def main():

    showstartscreen = 1

    while 1:
        ######## CONSTANTS

        WINSIZE = [800,600]
        WHITE = [255,255,255]
        BLACK = [0,0,0]
        RED = [255,0,0]
        GREEN = [0,255,0]
        BLUE = [0,0,255]
        BLOCKSIZE = [20,20]
        UP = 1
        DOWN = 3
        RIGHT = 2
        LEFT = 4
        MAXX = 760
        MINX = 20
        MAXY = 560
        MINY = 80
        SNAKESTEP = 20
```

```
TRUE = 1
FALSE = 0


######## VARIABLES

direction = RIGHT # 1=up,2=right,3=down,4=left
snakexy = [300,400]
snakelist = [[300,400],[280,400],[260,400]]
counter = 0
score = 0
appleonscreen = 0
#applexy = [0,0]
newdirection = RIGHT
snakedead = FALSE
gameregulator = 6
gamepaused = 0
growsnake = 0  # added to grow tail by two each time
snakegrowunit = 2 # added to grow tail by two each time


pygame.init()
clock = pygame.time.Clock()
screen = pygame.display.set_mode(WINSIZE)
pygame.display.set_caption('SNAKER')
screen.fill(BLACK)

#### show initial start screen

if showstartscreen == TRUE:
   showstartscreen = FALSE

   s                                                                                    =
[[180,120],[180,100],[160,100],[140,100],[120,100],[100,100],[100,120],[100,140],[100,160],[1
20,160],[140,160],[160,160],[180,160],[180,180],[180,200],[180,220],[160,220],[140,220],[120,
220],[100,220],[100,200]]
   apple = [100,200]

   pygame.draw.rect(screen,GREEN,Rect(apple,BLOCKSIZE))
   pygame.display.flip()
   clock.tick(8)
```

```
        for e in s:
            pygame.draw.rect(screen,BLUE,Rect(e,BLOCKSIZE))
            pygame.display.flip()
            clock.tick(8)

        font = pygame.font.SysFont("arial", 64)
        text_surface = font.render("NAKER", True, BLUE)
        screen.blit(text_surface, (220,180))
        font = pygame.font.SysFont("arial", 24)
        text_surface = font.render("Move the snake with the arrow keys to eat the apples", True,
GREEN)
        screen.blit(text_surface, (50,300))
        text_surface = font.render("Avoid the walls and yourself !", True, GREEN)
        screen.blit(text_surface, (50,350))
        text_surface = font.render("Press s to start a new game - Press q to quit at any time", True,
GREEN)
        screen.blit(text_surface, (50,400))
        text_surface = font.render("Press p to pause r to resume at any time", True, GREEN)
        screen.blit(text_surface, (50,450))

        pygame.display.flip()
        while 1:
            for event in pygame.event.get():
                if event.type == QUIT:
                    exit()

            pressed_keys = pygame.key.get_pressed()
            if pressed_keys[K_q]:
                pygame.quit()
                sys.exit()
            if pressed_keys[K_s]: break

            clock.tick(10)


    while not snakedead:

        ###### get input events  ####

        for event in pygame.event.get():
            if event.type == QUIT:
                exit()
```

```
pressed_keys = pygame.key.get_pressed()

if pressed_keys[K_LEFT]: newdirection = LEFT
if pressed_keys[K_RIGHT]: newdirection = RIGHT
if pressed_keys[K_UP]: newdirection = UP
if pressed_keys[K_DOWN]: newdirection = DOWN
if pressed_keys[K_q]: snakedead = TRUE
if pressed_keys[K_p]: gamepaused = 1

### wait here if p key is pressed until p key is pressed again

while gamepaused == 1:
    for event in pygame.event.get():
        if event.type == QUIT:
            exit()
    pressed_keys = pygame.key.get_pressed()
    if pressed_keys[K_r]:
        gamepaused = 0
    clock.tick(10)


### added gameregulator because setting a very low clock ticks
### caused the keyboard input to be hit and miss.  So I up the
### gameticks and the input and screen refresh is at this rate
### but the snake moving and all other logic is at the slower
### "regulated" speed


if gameregulator == 6:

    ####### lets make sure we can't go back the reverse direction

    if newdirection == LEFT and not direction == RIGHT:
        direction = newdirection

    elif newdirection == RIGHT and not direction == LEFT:
        direction = newdirection

    elif newdirection == UP and not direction == DOWN:
        direction = newdirection
```

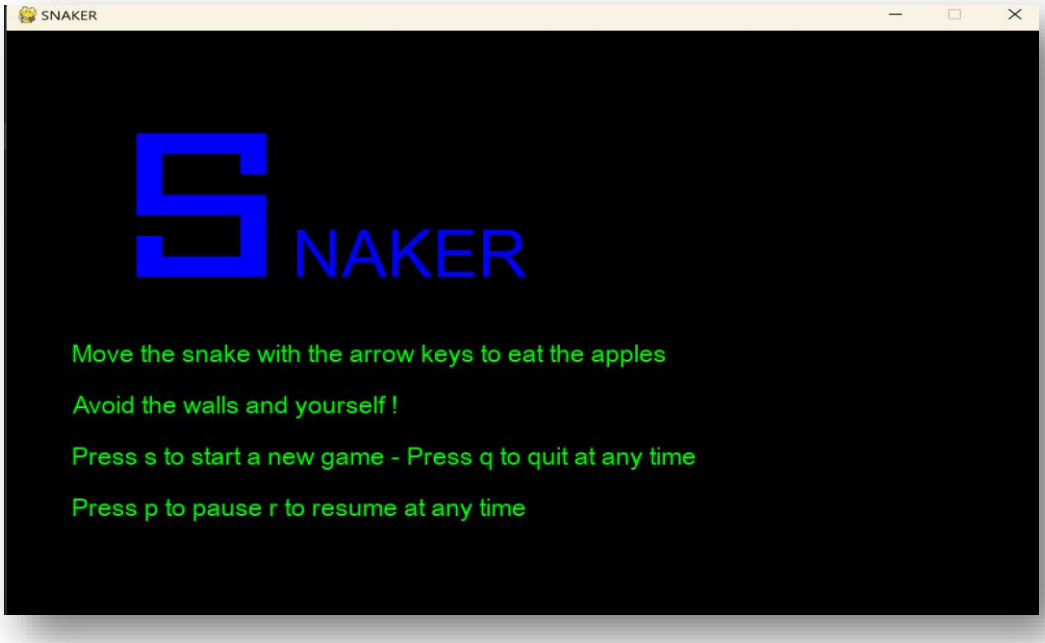
```
elif newdirection == DOWN and not direction == UP:
    direction = newdirection

##### now lets move the snake according to the direction
##### if we hit the wall the snake dies
##### need to make it less twitchy when you hit the walls


if direction == RIGHT:
    snakexy[0] = snakexy[0] + SNAKESTEP
    if snakexy[0] > MAXX:
        snakedead = TRUE

elif direction == LEFT:
    snakexy[0] = snakexy[0] - SNAKESTEP
    if snakexy[0] < MINX:
        snakedead = TRUE

elif direction == UP:
    snakexy[1] = snakexy[1] - SNAKESTEP
    if snakexy[1] < MINY:
        snakedead = TRUE

elif direction == DOWN:
    snakexy[1] = snakexy[1] + SNAKESTEP
    if snakexy[1] > MAXY:
        snakedead = TRUE

### is the snake crossing over itself
### had to put the > 1 test in there as I was
### initially matching on first pass otherwise - not sure why

if len(snakelist) > 3 and snakelist.count(snakexy) > 0:
    snakedead = TRUE



#### generate an apple at a random position if one is not on screen
#### make sure apple never appears in snake position

if appleonscreen == 0:
    good = FALSE
```

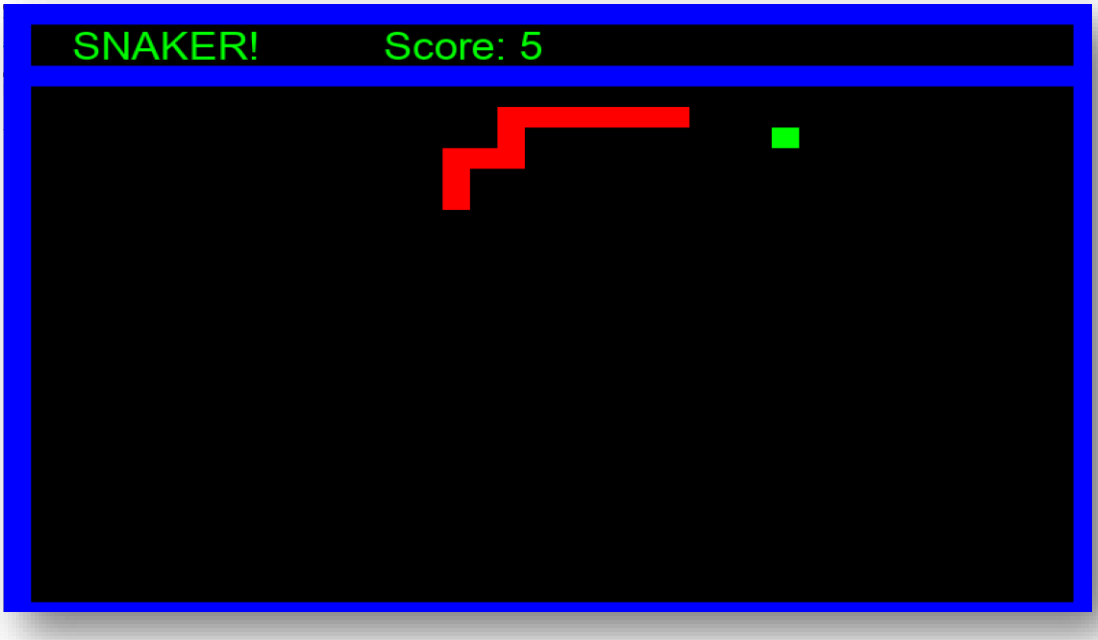
```
        while good == FALSE:
            x = random.randrange(1,39)
            y = random.randrange(5,29)
            applexy = [int(x*SNAKESTEP),int(y*SNAKESTEP)]
            if snakelist.count(applexy) == 0:
                good = TRUE
        appleonscreen = 1

    #### add new position of snake head
    #### if we have eaten the apple don't pop the tail ( grow the snake )
    #### if we have not eaten an apple then pop the tail ( snake same size )

    snakelist.insert(0,list(snakexy))
    if snakexy[0] == applexy[0] and snakexy[1] == applexy[1]:
        appleonscreen = 0
        score = score + 1
        growsnake = growsnake + 1
    elif growsnake > 0:
        growsnake = growsnake + 1
        if growsnake == snakegrowunit:
            growsnake = 0
    else:
        snakelist.pop()



    gameregulator = 0


###### RENDER THE SCREEN ###############

###### Clear the screen
screen.fill(BLACK)

###### Draw the screen borders
### horizontals
pygame.draw.line(screen,BLUE,(0,9),(799,9),20)
pygame.draw.line(screen,BLUE,(0,590),(799,590),20)
pygame.draw.line(screen,BLUE,(0,69),(799,69),20)
### verticals
pygame.draw.line(screen,BLUE,(9,0),(9,599),20)
pygame.draw.line(screen,BLUE,(789,0),(789,599),20)
```

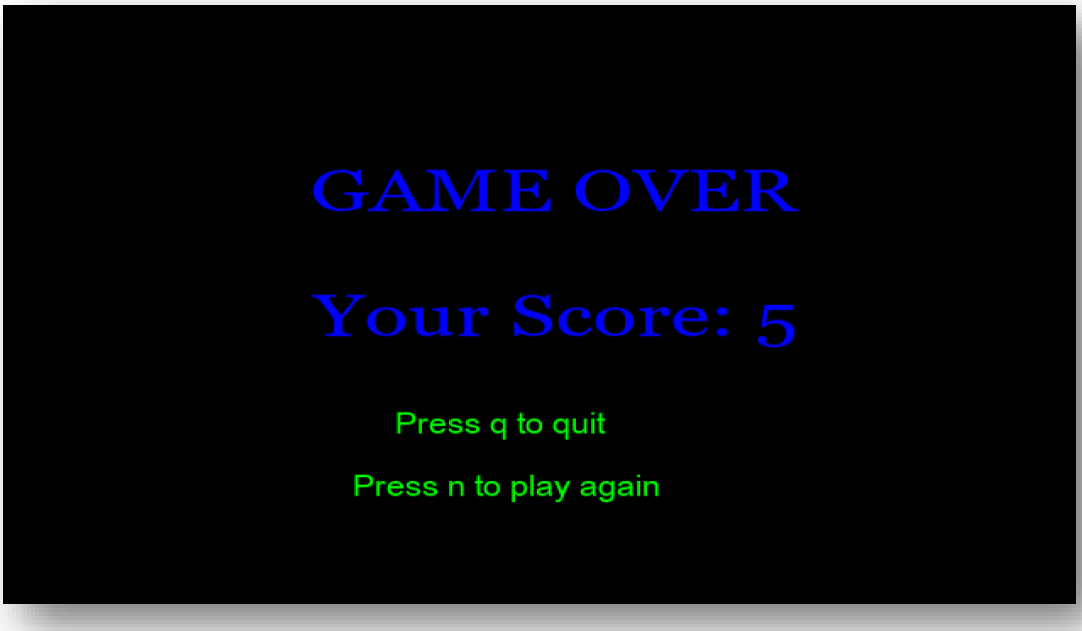
```
        ###### Print the score
        font = pygame.font.SysFont("arial", 38)
        text_surface = font.render("SNAKER!          Score: " + str(score), True, GREEN)
        screen.blit(text_surface, (50,18))

        ###### Output the array elements to the screen as rectangles ( the snake)
        for element in snakelist:
            pygame.draw.rect(screen,RED,Rect(element,BLOCKSIZE))

        ###### Draw the apple
        pygame.draw.rect(screen,GREEN,Rect(applexy,BLOCKSIZE))

        ###### Flip the screen to display everything we just changed
        pygame.display.flip()



        gameregulator = gameregulator + 1

        clock.tick(25)


    ##### if the snake is dead then it's game over

    if snakedead == TRUE:
        screen.fill(BLACK)
        font = pygame.font.SysFont("georgia", 48)
        text_surface = font.render("GAME OVER", True, BLUE)
        screen.blit(text_surface, (250,200))
        text_surface = font.render("Your Score: " + str (score), True, BLUE)
        screen.blit(text_surface, (250,300))
        font = pygame.font.SysFont("arial", 24)
                                    text_surface = font.render("Press q to q14
uit", True, GREEN)
        screen.blit(text_surface, (300,400))
        text_surface = font.render("Press n to play again", True, GREEN)
        screen.blit(text_surface, (275,450))

        pygame.display.flip()
        while 1:
            for event in pygame.event.get():
```

```
            if event.type == QUIT:
                exit()

        pressed_keys = pygame.key.get_pressed()
        if pressed_keys[K_q]: exit()
        if pressed_keys[K_n]: break

        clock.tick(10)

if __name__ == '__main__':
    main()
```

6. OUTPUT SCREENS




SNAKER!
Score: 5


GAME OVER
Your Score: 5
Press q to quit
Press n to play again

7. CONCLUSION

The Snake Game implemented using Pygame combines classic gameplay with modern Python programming, offering an engaging and nostalgic experience. The project successfully achieves its objectives of providing an interactive and entertaining game where players control a snake to consume apples while avoiding collisions with the walls and the snake's own body.

Throughout the implementation, effective use of Pygame's functionalities enables smooth graphics rendering, user input handling, and game logic execution. The modular design of the code promotes readability and maintainability, allowing for easy modifications or additions to enhance the game further.

The incorporation of a start screen with instructions adds a user-friendly element to the game, guiding players on how to navigate and play. The game's visual elements, such as the snake, apple, and borders, are well-designed and contribute to an immersive gaming experience.

The inclusion of features like pausing/resuming the game and displaying the player's score enhances the overall gaming experience. The project strikes a balance between simplicity and complexity, making it suitable for both beginners and experienced programmers to study, learn, and extend.

In conclusion, the Snake Game project demonstrates the practical application of Python programming, particularly in the context of game development. It provides an excellent foundation for further exploration of game design principles and serves as an enjoyable programming exercise for those looking to develop their skills in Python and Pygame.

8. FUTURE SCOPE

The Snake Game project exhibits significant potential for future enhancements, each contributing to an enriched gaming experience. Firstly, the introduction of advanced AI opponents can elevate the single-player mode, employing sophisticated algorithms or even exploring machine learning techniques for adaptive and challenging adversaries. Another avenue for development involves integrating multiplayer features, enabling real-time competition between players, and potentially introducing online multiplayer capabilities for a global gaming experience.

Customization options present an opportunity to engage players further. This includes expanding personalization features, allowing players to customize the snake's appearance and the game's visual elements. Additionally, a level editor could empower players to create and share their own levels, fostering community engagement and extending the game's longevity.

Enhancing gameplay dynamics, the inclusion of power-ups and new obstacles adds depth and variety. Dynamic environments, featuring moving obstacles or changing backgrounds, contribute to a more interactive and engaging game world. An achievement system with rewards, such as unlockable skins or themes, provides players with tangible goals and incentives.

Mobile optimization is a crucial aspect, ensuring the game's accessibility on diverse platforms. This may involve adapting controls for touch interfaces and publishing the game on app stores. Improving graphics and animations enhances the visual appeal, creating a more immersive gaming experience.

Leaderboards and tournament features contribute to the competitive aspect, allowing players to showcase their skills globally. The introduction of a story mode with quests provides narrative depth and additional challenges. Creating a vibrant community around the game, through forums or social media, encourages player interactions and feedback.

Cross-platform compatibility ensures seamless transitions between devices, providing flexibility to players. Implementing accessibility features, such as adjustable difficulty levels and colorblind-friendly options, ensures inclusivity. Lastly, continuous bug fixes, updates, and an open feedback loop maintain the game's quality and responsiveness to player preferences. This comprehensive future scope encompasses a wide array of possibilities for evolving the Snake Game project.

9. BIBILOGRAPHY

1. https://github.com/harsitbaral/Snake/blob/main/main.py
2. https://github.com/harsitbaral/Snake/blob/main/font.ttf
3. https://www.edureka.co/blog/snake-game-with-pygame/
4. https://gist.github.com/wynand1004/ec105fd2f457b10d971c09586ec44900