College Of Basic Science & Humanities, OUAT

Hungry Snake: RE-Live the Childhood Using Python

Project submitted in fulfillment of the requirement for the Degree of Under Graduate in Computer Science

By

PRANGYA PARIMEETA DEHURY
BS-20-253

Under the guidance of:
Mrs. Binita Dash
Department of Computer Science

DECLARATION

I, **PRANGYA PARIMEETA DEHURY**, hereby declare that the project entitled "HUNGRY SNAKE" submitted by me in partial fulfillment for the project work is a bona fide work carried out by me under the guidance of **Mrs. BINITA DASH** and the result embodied in this project has not been submitted to any other university for the award of any other degree.

Date:

Place: Bhubaneswar Signature

ACKNOWLEDGEMENT

It is a great pleasure to be able to express my appreciation towards all the people who have helped me in various ways for the successful completion of the project entitled "HUNGRY SNAKE". We acknowledge the support given to us by a number of individuals in the preparation of this project. We would like to express a special debt of gratitude to our guide Mrs. Binita Dash for her scholarly guidance, encouragement and inspiration which added a lot to the successful completion of the project. We have learnt a lot from her in so many ways, both personally and technically. We are also acknowledging our heartiest thanks to the director and staff of "COLLEGE OF BASIC SCIENCE AND HUMANITIES, OUAT " for their kind help, motivation and co-operation. Last but not the least, we acknowledge our sincere gratitude to our parents and family members for their constant encouragement and love who has directly or indirectly helped me in completing this project successfully.

Name: Prangya Parimeeta Dehury
Roll no: BS-20-253

CERTIFICATE

This is to certify that the project entitled "HUNGRY SNAKE" is a bona fide work done by PRANGYA PARIMEETA DEHURY, Roll No: BS-20-253 in partial fulfillment of academic project work and has been carried out under your direct supervision and guidance. This report or a similar report on the topic has not been submitted for any other examination and does not form a part of any other course undergone by the candidate.

--	--
Signature of External	**Signature of Internal**

CONTENTS

Sl. no.	Topic	Page No.
1	INTRODUCTION	6
2	HISTORY	7
3	PROJECT INFLUENCE	8
4	TECHNOLOGY AND SOFTWARE USED	9
5	METHODOLOGY	11
6	GAME LOGIC	12
8	SOURCE CODE	15
9	GAME PLAY	24
12	CONCLUSION	27
13	REFERENCES	28

INTRODUCTION

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. It’s a simple game of infinite level type. It’s a challenging game for all. Every one of every age group can play this game. It is addictive as well.

In the action video game genre known as "Snake," the player controls the tip of a snake-themed expanding line. It becomes more difficult as the snake grows longer to prevent it from hitting itself as well as other obstacles. To survive longer than your opponent is the object of the 1976 Gremlin Industries two-player arcade video game Blockade. With each bite of food—often apples or eggs—a snake grows longer in a single-player variation of the idea. Games like snake have hundreds of variations—some of which have the words snake or worm in the title—for numerous platforms due to their simplicity and minimal technical requirements.

HISTORY

The Snake began with Gremlin's arcade video game Blockade, which was released in 1976.The same year, it was cloned as Bigfoot Bonkers. Atari, Inc. released two Blockade-inspired titles in 1977: Dominos, an arcade game, and Surround, an Atari VCS game. Surround was one of nine Atari VCS launch titles in the United States, and it was sold by Sears under the brand name Chase. Checkmate, a similar game for the Bally Astrocade, was released the same year.

Worm, the first known home computer version, was written in 1978 by Peter Trefonas for the TRS-80[2] and published the same year by CLOAD magazine. This was quickly followed by versions from the same author for the Commodore PET and Apple II. Peter Trefonas wrote and published CLOAD in 1979 a clone of the Hustle arcade game, which was itself a clone of Blockade.[7] Milton Bradley released an authorized version of Hustle for the TI-99/4A in 1980.[8] Snake Byte, a single-player game for Atari 8-bit computers, Apple II, and VIC-20, was released in 1982; a snake eats apples to complete a level, growing longer in the process. The snake in Dave Bresnen's Snake for the BBC Micro (1982) is controlled by the left and right arrow keys relative to the direction it is moving. The snake's speed increases as it grows longer, and there is only one life.

PROJECT INFLUENCE

We chose this game because first we are new to this field and we wanted to develop a project which is easy to make and easily accessible to any type of user. This is the modern world and esports is growing in India. We can see many esports players growing and getting a steady income source. Not only esports streaming careers as well. People are choosing games for streaming or esports competition as their jobs. There are many organizations who are giving chances to these people to make name, fame, money like velocity gaming, 8bit, S8UL etc.

Our main purpose in choosing this game is that from the early age group to the very old age people can access this game and get an idea that you can choose gaming as your career as well. We wanted to establish that everyone from an early age group can be a live streamer or esports player.

TECHNOLOGY AND SOFTWARE USED

We used **PYTHON** programming language to develop our game and **VSCode** Application is used to write the code. We also used **PYGAME** library to make our game. We also used a converter to convert our **python** file to **exe** file so that anyone can access the even those who does not have python installed. We used all these in our pc or desktop to make our game. The Hungry Snake Game using PyGame in Python is one of the most popular high-level programming languages. Python offers huge libraries for different fields like Artificial Intelligence (TensorFlow, PyTorch), Machine Learning (Pandas, NumPy, Matplotlib), and Game Development (Pyglet, PyGame). We can also consider Python as the next-generation programming language as it shows its presence actively in every emerging field of Computer Science.

Let us briefly understand the Python PyGame library. Understanding the PyGame library The PyGame library is a cross-platform set of Python modules utilized to develop video games. PyGame mainly comprises computer graphics and sound libraries designed to be utilized with the Python programming language. It is suitable to develop client-side applications that can be potentially wrapped in a standalone executable.

How to install the PyGame library? The PyGame library can be installed using the PIP installer by typing the following command in a command prompt or terminal.

Syntax:

```
# installing the PyGame library
$ pip install pygame
```

Once the installation is complete, we can verify whether the pygame library is installed properly or not by creating a new python program file and importing the pygame module. The following is the snippet of code illustrating the same.

File: verify.py

```
import pygame
```

Now, let us save the file and run the following command in a command prompt or terminal.

Syntax:

```
$ python verify.py
```

The library has been installed successfully if the program does not return any importing error. In case any exception is raised, try reinstalling the library and consider checking their official documentation.

METHODOLOGY

- Firstly, we will import the required modules.
- Then the classes and objects are added for background ,bird, pipes, main menu and game over screen.
- We add relevant game logic for smooth play and in accordance with screen and system settings.
- We add the Sprite Animation to the Game.
- We will then add some physics to the Game.
- We will then add a score counter.
- We will last add functions for game over and reset the game.
- Once everything is in check the user can execute the code and run the game.
- We also made an exe file so that everyone can play the game without any restrictions.

GAME LOGIC

It is true that Hungry Snake is one of the legendary games we used to play, 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.

1. **Create the Screen**: To create the screen using Pygame, you will need to make use of the display.set_mode() function. Also, you will have to make use of the init () and the quit () methods to initialize and initialize everything at the start and the end of the code. The update () method is used to update any changes made to the screen. There is another method i.e., flip () that works similarly to the update () function. The difference is that the update () method updates only the changes that are made (however, if no parameters are passed, updates the complete screen) but the flip () method redoes the complete screen again.

2. **Create the Snake**: To create the snake, I will first initialize a few color variables in order to color the snake, food, screen, etc. The color scheme used in Pygame is RGB i.e., “Red Green Blue ". In case you set all these to 0’s, the color will be black and all 255’s

will be white. So, our snake will actually be a rectangle. To draw rectangles in Pygame, you can make use of a function called draw. rect () which will help you draw the rectangle with the desired color and size.

3. **Moving the Snake**: To move the snake, you will need to use the key events present in the KEYDOWN class of Pygame. The events that are used over here are, K_UP, K_DOWN, K_LEFT, and K_RIGHT to make the snake move up, down, left and right respectively. Also, the display screen is changed from the default black to white using the fill () method. I have created new variables x1_change and y1_change in order to hold the updating values of the x and y coordinates.

4. **Game Over when Snake hits the boundaries**: In this snake game, if the player hits the boundaries of the screen, then he loses. To specify that, I have made use of an 'if' statement that defines the limits for the x and y coordinates of the snake to be less than or equal to that of the screen. Also, make a note over here that I have removed the hardcodes and used variables instead so that it becomes easy in case you want to make any changes to the game later on.

5. **Adding the Food**: Here, I will be adding some food for the snake and when the snake crosses over that food, I will have a message saying “Yummy!!”. Also, I will be making a small change wherein I will include the options to quit the game or to play again when the player loses.

6. **Increasing the Length of the Snake**: The following code will increase the size of our sake when it eats the food. Also, if the snake collides with his own body, the game is over and you will see a message as “You Lost! Press Q-Quit or C-Play Again “. The length of the snake is basically contained in a list and the initial size that is specified in the following code is one block. Displaying the Score: Last but definitely not the least, you will need to display the score of the player. To do this, I have created a new function called “Your_score ". This function will display the length of the snake subtracted by 1 because that is the initial size of the snake.

7. **Displaying the Score**: Last but definitely not the least, you will need to display the score of the player. To do this, I have created a new function called “Your_score”. This function will display the length of the snake subtracted by 1 because that is the initial size of the snake.

SOURCE CODE

```
import pygame
import time
import random
from pygame.locals import*
from pygame import mixer

mixer.init()
mixer.music.load("D:\music\Lukrembo - Onion.wav")
mixer.music.play()

snake_speed = 10

# Window size
window_x = 720
window_y = 480

# defining colors
black = pygame.Color(0, 0, 0)
white = pygame.Color(255, 255, 255)
red = pygame.Color(255, 0, 0)
green = pygame.Color(0, 255, 0)
```

```
blue = pygame.Color(0, 0, 255)

                                                  # Initializing pygame
pygame.init()

                                                  # Initialise game window
pygame.display.set_caption('hungry Snakes')
game_window = pygame.display.set_mode((window_x,
window_y))

                                                  # FPS (frames per second) controller
fps = pygame.time.Clock()

                                                  # defining snake default position
snake_position = [100, 50]

                                                  # defining first 4 blocks of snake body
snake_body = [[100, 50],
              [90, 50],
              [80, 50],
              [70, 50]]
```

```
                                                   # fruit position
fruit_position = [random.randrange(1, (window_x//10)) *
                  10, random.randrange(1, (window_y//10)) * 10]
fruit_spawn = True

                        # setting default snake direction towards
                        # right
direction = 'RIGHT'
change_to = direction

                                                    # initial score
score = 0

                                       # displaying Score function
def show_score(choice, color, font, size):

                                            # creating font object
score_font
score_font = pygame.font.SysFont(font, size)

                                # create the display surface object
                                # score_surface
```

```
31.      score_surface = score_font.render('Score : ' + str(score),
                                            True, color)

                                   # create a rectangular object for the text
                                   # surface object
32.      score_rect = score_surface.get_rect()

                                                          # displaying text
33.      game_window.blit(score_surface, score_rect)

                                                       # game over function
34.      def game_over():

                                               # creating font object my_font
35.      my_font = pygame.font.SysFont('times new roman', 50)

                         # creating a text surface on which text will be drawn
36.      game_over_surface = my_font.render( 'Your Score is : ' +
                                               str(score), True, red)

                                   # create a rectangular object for the text
                                   # surface object
```

```
game_over_rect = game_over_surface.get_rect()


                                        # setting position of the text
game_over_rect.midtop = (window_x/2, window_y/4)

                                  # blit will draw the text on screen
game_window.blit(game_over_surface, game_over_rect)
               pygame.display.flip()


                            # after 2 seconds we will quit the program
time.sleep(2)

                                       # deactivating pygame library


pygame.quit()

                                                  # quit the program
quit()

                                                     # Main Function
while True:
```

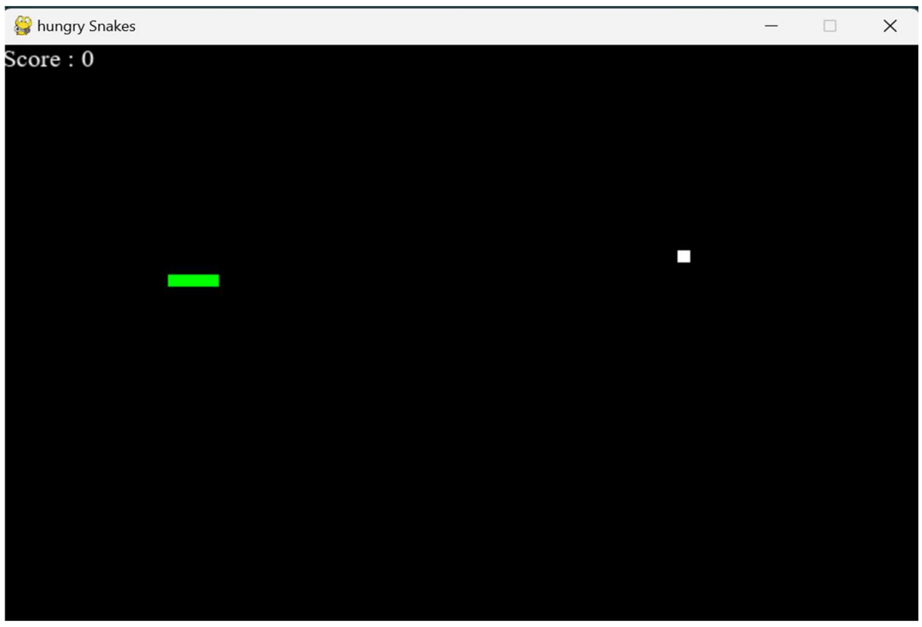
```
                                             # handling key events
    for event in pygame.event.get():
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_UP:
  change_to = 'UP'
            if event.key == pygame.K_DOWN:
  change_to = 'DOWN'
            if event.key == pygame.K_LEFT:
  change_to = 'LEFT'
            if event.key == pygame.K_RIGHT:
  change_to = 'RIGHT'

                                   # If two keys pressed simultaneously
                                 # we don't want snake to move into two
                                   # directions simultaneously
            if change_to == 'UP' and direction != 'DOWN':
 direction = 'UP'
            if change_to == 'DOWN' and direction != 'UP':
 direction = 'DOWN'
            if change_to == 'LEFT' and direction != 'RIGHT':
 direction = 'LEFT'
            if change_to == 'RIGHT' and direction != 'LEFT':
```

```
   direction = 'RIGHT'

                                                  # Moving the snake
54.          if direction == 'UP':
    snake_position[1] -= 10
55.          if direction == 'DOWN':
   snake_position[1] += 10
56.          if direction == 'LEFT':
   snake_position[0] -= 10
57.          if direction == 'RIGHT':
   snake_position[0] += 10

                                       # Snake body growing mechanism
                                   # if fruits and snakes collide then scores
                                       # will be incremented by 10
58.     snake_body.insert(0, list(snake_position))
59.     if snake_position[0] == fruit_position[0] and
      snake_position[1] == fruit_position[1]:
                score += 5
                fruit_spawn = False
60.       else:
                snake_body.pop()
```

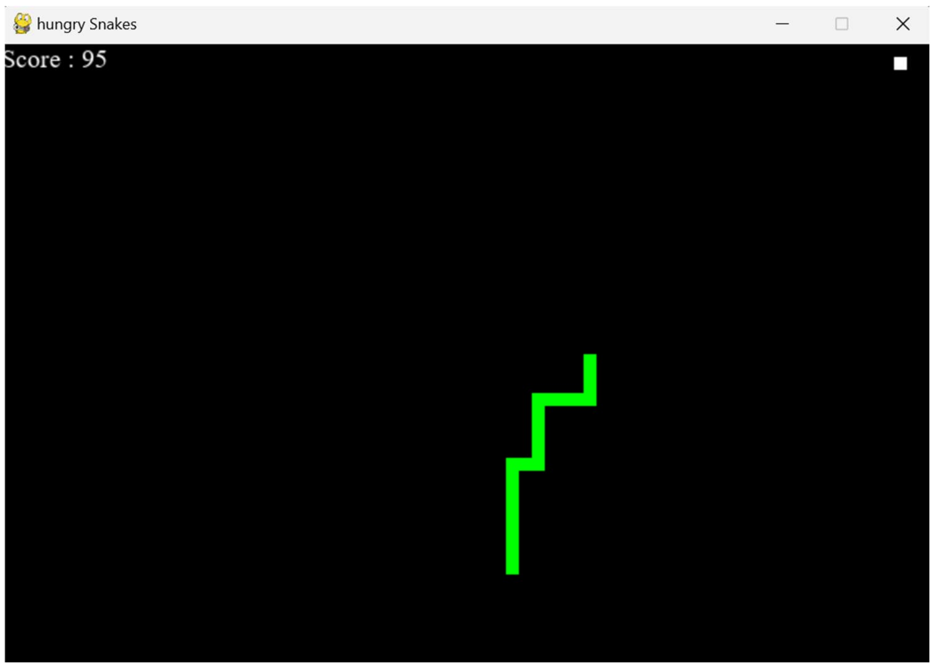
```
61.        if not fruit_spawn:
 fruit_position = [random.randrange(1, (window_x//10)) * 10,
          random.randrange(1, (window_y//10)) * 10]
   fruit_spawn = True
   game_window.fill(black)

62.        for pos in snake_body:
          pygame.draw.rect(game_window, green,
          pygame.Rect(pos[0], pos[1], 10, 10))
    pygame.draw.rect(game_window, white, pygame.Rect(
   fruit_position[0], fruit_position[1], 10, 10))

                                                   # Game Over conditions

63.        if snake_position[0] < 0 or snake_position[0] >
                   window_x-10:
                        game_over()
64.        if snake_position[1] < 0 or snake_position[1] >
      window_y-10:
             game_over()

                                                # Touching the snake body
65.        for block in snake_body[1:]:
    if snake_position[0] == block[0] and snake_position[1]
== block[1]:
            game_over()



                                        # displaying score continuously
show_score(1, white, 'times new roman', 20)

                                        # Refresh game screen
pygame.display.update()

                                        # Frame Per Second /Refresh Rate
fps.tick(snake_speed)
```

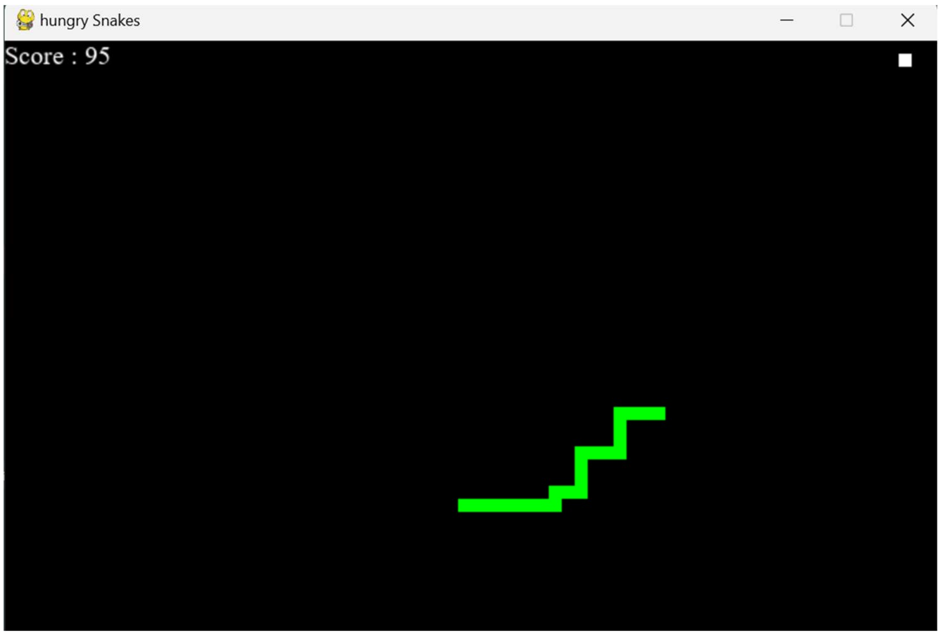
GAME PLAY

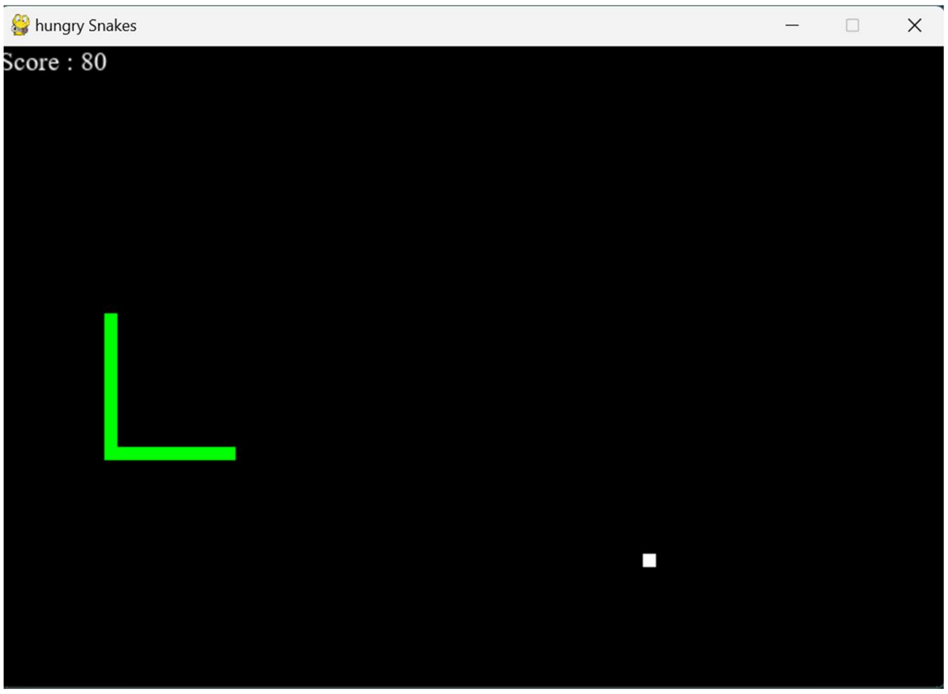
1. START OF GAME:

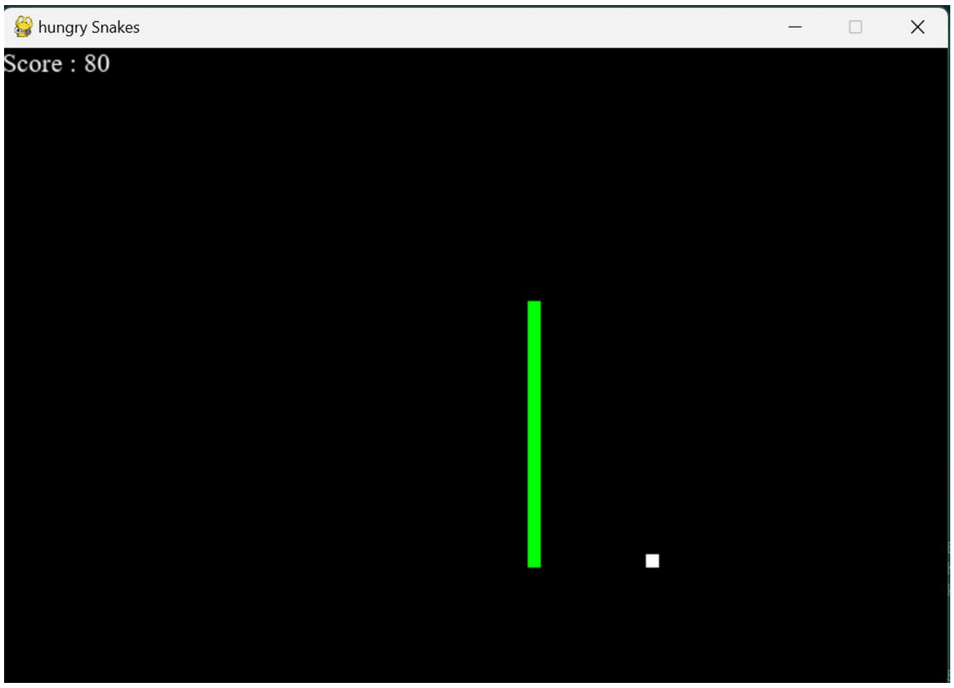


2. DIFFERENT DIRECTIONAL ORIENTATION

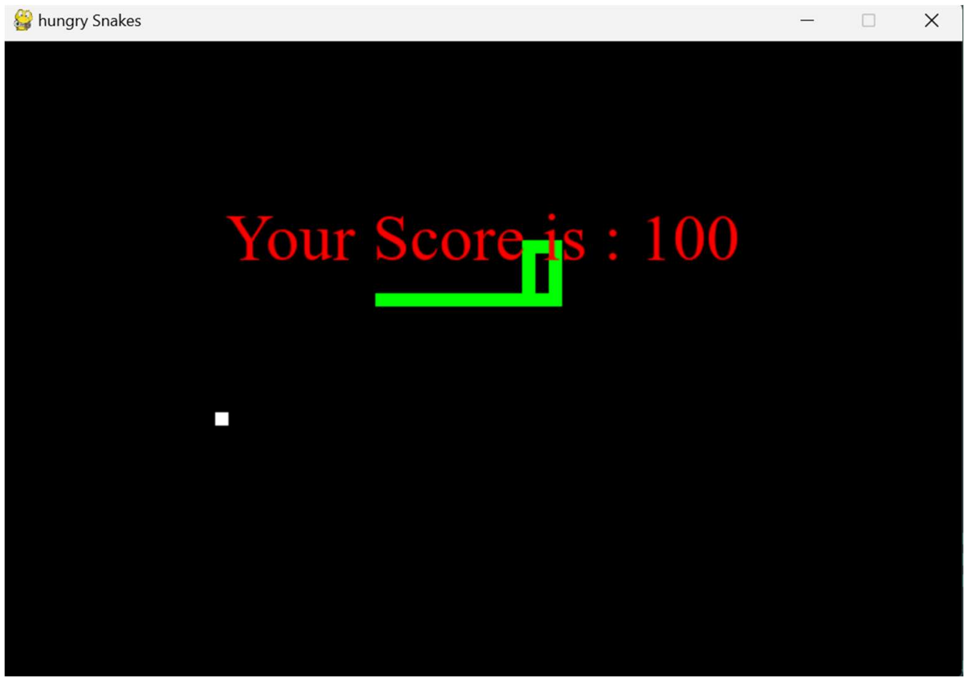



hungry Snakes
Score : 95


hungry Snakes
Score : 80



3. END OF GAME:



Your Score is : 100

CONCLUSION

In conclusion, creating the “HUNGRY SNAKE” game in Python can be a fun and rewarding programming project. Throughout this process, I have learned about various concepts such as user input, game logic, game state, and display. By implementing these components and understanding their interaction, I have successfully built a functioning snake game.

Python's simplicity and versatility make it a great language choice for developing games like the “HUNGRY SNAKE” game. Its extensive libraries and frameworks provide ample support for handling user input, graphics rendering, and game logic. Additionally, Python's readability and ease of use make the development process more enjoyable. This snake game is just a starting point. One can further enhance the game by adding features like different levels of difficulty, power-ups, or multiplayer functionality. With creativity and imagination, one can customize the game to your liking and challenge yourself to improve your programming skills.

Enjoy the game I have created and continue exploring Python's vast possibilities for developing more exciting projects!

REFERENCES

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

2. https://www.youtube.com/@CodeWithHarry

3. https://melmagazine.com/en-us/story/snake-nokia-6110- oral-history-taneli-armanto

4. https://en.wikipedia.org/wiki/Snake? (video_game_genre)

5. https://www.geeksforgeeks.org/snake-game-in-python-using-pygame-module/

6. https://pythonspot.com/snake-with-pygame

7. https://docs.python.org/3/tutorial

8. https://www.learnpython.org