SNAKE BITE

A PROJECT REPORT

Submitted by

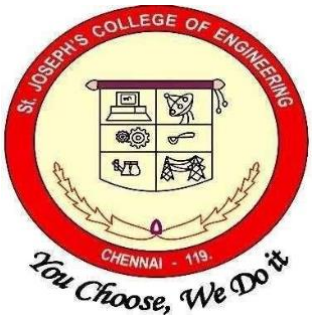
Fuaad Ahmad – 312322205303
Guruprakash 312322205054

Of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY



St. JOSEPH'S COLLEGE OF ENGINEERING

(An Autonomous Institution)

St. Joseph's Group of Institutions

OMR, Chennai 600 119

ANNA UNIVERSITY: CHENNAI

Table of Contents:

1. Project Overview
2. Project Components
3. Features
4. Configuration
5. Development Details
6. Conclusion

Project Overview:

Snake Eater is a classic arcade-style game implemented using PyGame, a popular Python library for game development. The game provides a nostalgic experience reminiscent of the original Snake game found on early mobile phones. Players control a snake that moves around the game window, consuming food items to grow in length while avoiding collision with obstacles and its own body.

Project Components:

Main Script (snake.py):

- The main script serves as the entry point of the game.
- It handles game initialization, event handling, game logic, and rendering.
- Implements the main game loop responsible for continuously updating the game state and rendering graphics.

Modules and Libraries:

- PyGame: PyGame is a cross-platiorm set of Python modules designed for writing video games. It provides functionality for handling graphics, user input, sound, and more.
- Sys: The sys module provides access to some variables used or maintained by the Python interpreter and to functions that interact strongly with the interpreter.

- Time: The me module provides various me-related functions, such as introducing delays or measuring me intervals.
- Random: The random module provides functions for genera ng random numbers, which are used in this project for spawning food items at random positions.
  - """

Snake Eater Source code :

```
import pygame, sys, time, random


# Difficulty settings
# Easy      ->  10
# Medium    -> 25
# Hard      ->  40
# Harder    ->  60
# Impossible->  120
difficulty = 25

# Window size
frame_size_x = 720
frame_size_y = 480

# Checks for errors encountered
check_errors = pygame.init()
# pygame.init() example output -> (6, 0)
# second number in tuple gives number of errors
if check_errors[1] > 0:
    print(f'[!] Had {check_errors[1]} errors when initialising game,
exiting...')
    sys.exit(-1)
else:
    print('[+] Game successfully initialised')
```

```
# Initialise game window
pygame.display.set_caption('Snake Eater')
game_window = pygame.display.set_mode((frame_size_x,
frame_size_y))


# Colors (R, G, B)
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)


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


# Game variables
snake_pos = [100, 50]
snake_body = [[100, 50], [100-10, 50], [100-(2*10), 50]]

food_pos = [random.randrange(1, (frame_size_x//10)) * 10,
random.randrange(1, (frame_size_y//10)) * 10]
food_spawn = True

direction = 'RIGHT'
change_to = direction

score = 0


# Game Over
```

```
def game_over():
    my_font = pygame.font.SysFont('times new roman', 90)
    game_over_surface = my_font.render('YOU DIED', True, red)
    game_over_rect = game_over_surface.get_rect()
    game_over_rect.midtop = (frame_size_x/2, frame_size_y/4)
    game_window.fill(black)
    game_window.blit(game_over_surface, game_over_rect)
    show_score(0, red, 'times', 20)
    pygame.display.flip()
    time.sleep(3)
    pygame.quit()
    sys.exit()


# Score
def show_score(choice, color, font, size):
    score_font = pygame.font.SysFont(font, size)
    score_surface = score_font.render('Score : ' + str(score), True,
color)
    score_rect = score_surface.get_rect()
    if choice == 1:
        score_rect.midtop = (frame_size_x/10, 15)
    else:
        score_rect.midtop = (frame_size_x/2, frame_size_y/1.25)
    game_window.blit(score_surface, score_rect)
    # pygame.display.flip()


# Main logic
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()
```

```
            # Whenever a key is pressed down
            elif event.type == pygame.KEYDOWN:
                # W -> Up; S -> Down; A -> Left; D -> Right
                if event.key == pygame.K_UP or event.key == ord('w'):
                    change_to = 'UP'
                if event.key == pygame.K_DOWN or event.key == ord('s'):
                    change_to = 'DOWN'
                if event.key == pygame.K_LEFT or event.key == ord('a'):
                    change_to = 'LEFT'
                if event.key == pygame.K_RIGHT or event.key == ord('d'):
                    change_to = 'RIGHT'
                # Esc -> Create event to quit the game
                if event.key == pygame.K_ESCAPE:
                    pygame.event.post(pygame.event.Event(pygame.QUIT))

        # Making sure the snake cannot move in the opposite direction
instantaneously
        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
        if direction == 'UP':
            snake_pos[1] -= 10
        if direction == 'DOWN':
            snake_pos[1] += 10
        if direction == 'LEFT':
            snake_pos[0] -= 10
        if direction == 'RIGHT':
```

```
        snake_pos[0] += 10

    # Snake body growing mechanism
    snake_body.insert(0, list(snake_pos))
    if snake_pos[0] == food_pos[0] and snake_pos[1] == food_pos[1]:
        score += 1
        food_spawn = False
    else:
        snake_body.pop()

    # Spawning food on the screen
    if not food_spawn:
        food_pos = [random.randrange(1, (frame_size_x//10)) * 10,
random.randrange(1, (frame_size_y//10)) * 10]
    food_spawn = True

    # GFX
    game_window.fill(black)
    for pos in snake_body:
        # Snake body
        # .draw.rect(play_surface, color, xy-coordinate)
        # xy-coordinate -> .Rect(x, y, size_x, size_y)
        pygame.draw.rect(game_window, green, pygame.Rect(pos[0],
pos[1], 10, 10))

    # Snake food
    pygame.draw.rect(game_window, white, pygame.Rect(food_pos[0],
food_pos[1], 10, 10))

    # Game Over conditions
    # Getting out of bounds
    if snake_pos[0] < 0 or snake_pos[0] > frame_size_x-10:
        game_over()
    if snake_pos[1] < 0 or snake_pos[1] > frame_size_y-10:
```

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

    show_score(1, white, 'consolas', 20)
    # Refresh game screen
    pygame.display.update()
    # Refresh rate
    fps_controller.tick(difficulty)
```

- Features:
- Snake Movement and Controls:
    - Players can control the movement of the snake using arrow keys or the WASD keys.
    - The snake moves continuously in the direction specified by the player un l a change in direction is initiated.

- Food Spawning and Consumption:
    - Food items spawn at random positions within the game
        - window.
    - When the snake's head collides with a food item, the snake grows in length, and the player's score increases.

- Score Tracking and Display:
    - The game keeps track of the player's score, which increments each me the snake consumes food.
    - The current score is displayed on the game screen for the player to see during gameplay.

- Game Over Conditions:

- The game ends when the snake collides with the boundaries of the game window or with its own body. o Upon game over, a "YOU DIED" message is displayed along with the player's final score.

➢ Configuration:

➢ Difficulty Levels:

- The game offers adjustable difficulty levels, ranging from Easy to Impossible.
- Difficulty affects the speed of the snake's movement, making the game more challenging at higher difficulty levels.

➢ Window Size and Colors:

- The size of the game window is customizable, allowing players to adjust the display according to their preferences.
- Colors for the snake, food items, and other game elements can be customized to enhance visual appeal.

➢ Development Details:

➢ Error Handling:

- The project includes error handling mechanisms to detect initializa on errors during startup.
- If initializa on errors occur, the game gracefully exits with an error message, ensuring a smooth user experience.

➢ Key Event Handling:

- The game listens for key press events to control the snake's movement and to handle game exit events.
- Key mappings for movement (arrow keys and WASD) are implemented to provide flexibility for players.

➢ Snake Movement and Growth:

- The snake's movement is updated based on the player's input and game logic, ensuring smooth and responsive controls.
- When the snake consumes food, its length increases, simulating growth and adding to the challenge of navigating the game environment.

Food Spawning Logic:

- Food items spawn at random positions within the game window, ensuring variety in gameplay and preventing predictability.
- A mechanism is in place to prevent food from spawning on top of the snake's body, ensuring fair gameplay.

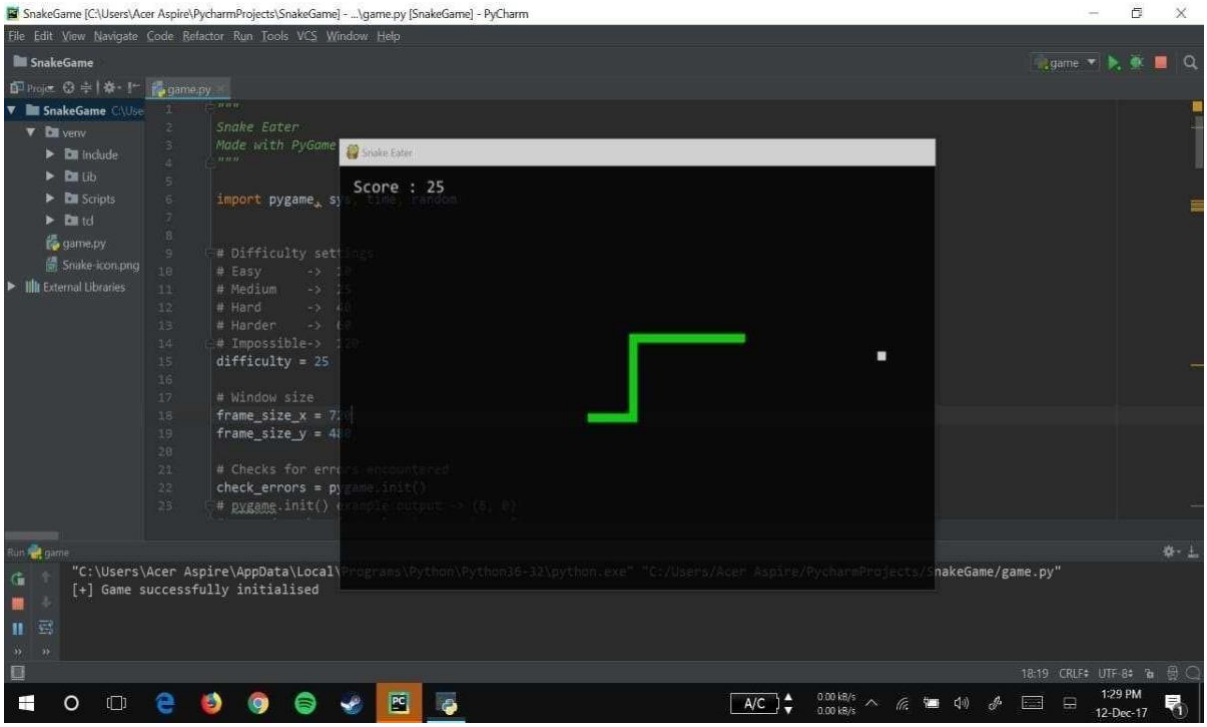
Game Over Screen:

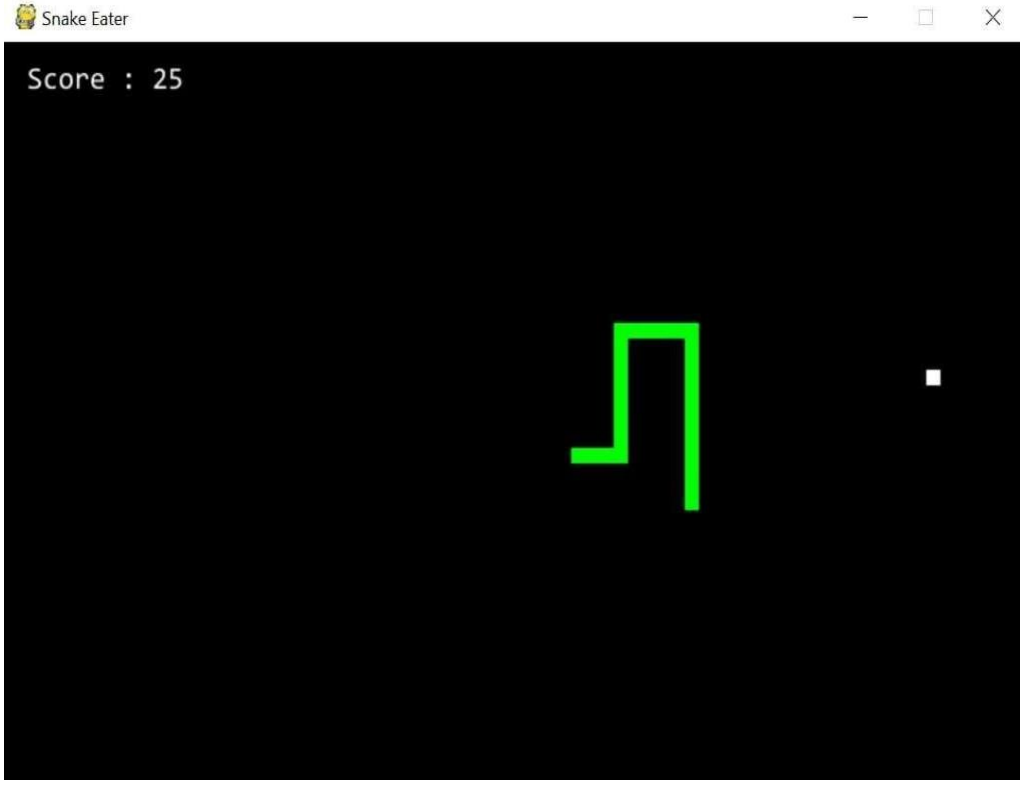
- Upon detecting game over conditions, such as collision with boundaries or the snake's own body, the game displays a "YOU DIED" message.
- The final score achieved by the player is also shown on the game over screen, providing feedback and encouraging replayability.

Conclusion:

- The Snake Eater project showcases the implementation of a classic arcade game using Python and PyGame. With its engaging gameplay mechanics, customizable and intuitive controls, the game offers players an enjoyable gaming experience. Whether you're a seasoned player looking for a nostalgic throwback or a beginner exploring game development, Snake Eater provides entertainment and learning opportunities in equal measure.

- This should make it easier for readers to navigate through the document and find specific sections they're interested in. Let me know if you need any further adjustments!


SnakeGame [C:\Users\Acer Aspire\PycharmProjects\SnakeGame] - ...\game.py [SnakeGame] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
SnakeGame
game.py
SnakeGame C:\Use
venv
Include
Lib
Scripts
tcl
game.py
Snake-icon.png
External Libraries
Snake Eater
Score : 25
Snake Eater
Made with PyGame
import pygame, sy
# Difficulty set
# Easy -> 1
# Medium -> 1
# Hard -> 4
# Harder -> 6
# Impossible-> 1
difficulty = 25
# Window size
frame_size_x = 7
frame_size_y = 4
# Checks for err
check_errors = py
# pygame.init()
Run game
"C:\Users\Acer Aspire\AppData\Local\Programs\Python\Python36-32\python.exe" "C:/Users/Acer Aspire/PycharmProjects/SnakeGame/game.py"
[+] Game successfully initialised
18:19 CRLF UTF-8
A/C
1:29 PM
12-Dec-17


Snake Eater
Score : 25