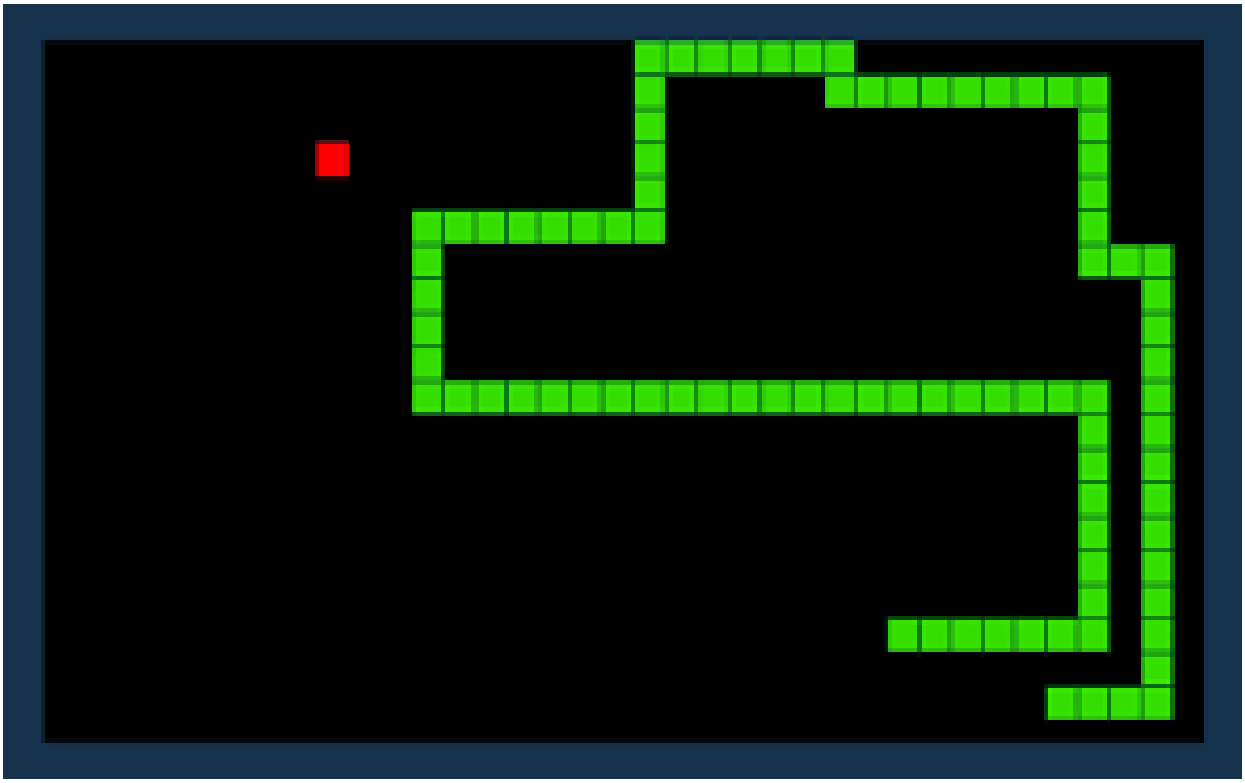
Snake Game



By -

Kalyan Ram P

P Sumanth Raj

Dhiran Kumar Reddy

INTRODUCTION

Python

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido Van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of codes.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

Reason for increasing popularity:

1. Emphasis on code readability, shorter codes, ease of writing.

2. Programmers can express logical concepts in fewer lines of code in comparison to languages such as C++ of Java.

3. Python supports multiple programming paradigms, like object-oriented, imperative and functional programming or procedural.

Pygame

Pygame 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.

Pygame was originally written by Pete Shinners to replace PySDL after its development stalled. It has been a community project since 2000 and is released under the open-source free software GNU Lesser General Public License.

Snake Game

Snake is the common name for a video game concept where the player maneuvers a line which grows in length, with the line itself being a primary obstacle. The concept originated in the 1976 arcade game *Blockade*, and the ease of implementing *Snake* has led to hundreds of versions (some of which have the word *snake* or *worm* in the title) for many platforms. After a variant was preloaded on Nokia mobile phones in 1998, there was a resurgence of interest in the snake concept as it found a larger audience.

Gameplay:

The player controls a dot, square, or object on a bordered plane. As it moves forward, it leaves a trail behind, resembling a moving snake. In some games, the end of the trail is in a fixed position, so the snake continually gets longer as it moves. In another common scheme, the snake has a specific length, so there is a moving tail a fixed number of units away from the head. The player loses when the snake runs into the screen border, a trail or other obstacle, or itself. The Snake concept comes in two major variants:

1. In the first, which is most often a two-player game, there are multiple snakes on the playfield. Each player attempts to block the other so the opponent runs into an existing trail and loses. The Light Cycles segment of the *Tron* arcade game is a single-player version where the other "snakes" are AI controlled.
2. In the second variant, a sole player attempts to eat items by running into them with the head of the snake. Each item eaten makes the snake longer, so avoiding collision with the snake becomes progressively

more difficult. Examples: *Nibbler*, *Snake Byte*.

EXTERNAL LIBRARY FUNCTIONS

- **Pygame:**
  - **pygame.display.set_mode():** controls the display and window screen.
  - **pygame.mouse.get_pos():** obtains the position of the mouse.
  - **pygame.draw():** draws shapes in the given position and colour.
  - **pygame.event.get():** interacts with events such as button clicks.
  - **pygame.display.set_caption():** displays the specified caption on the top left corner of the screen.

FUNCTIONS

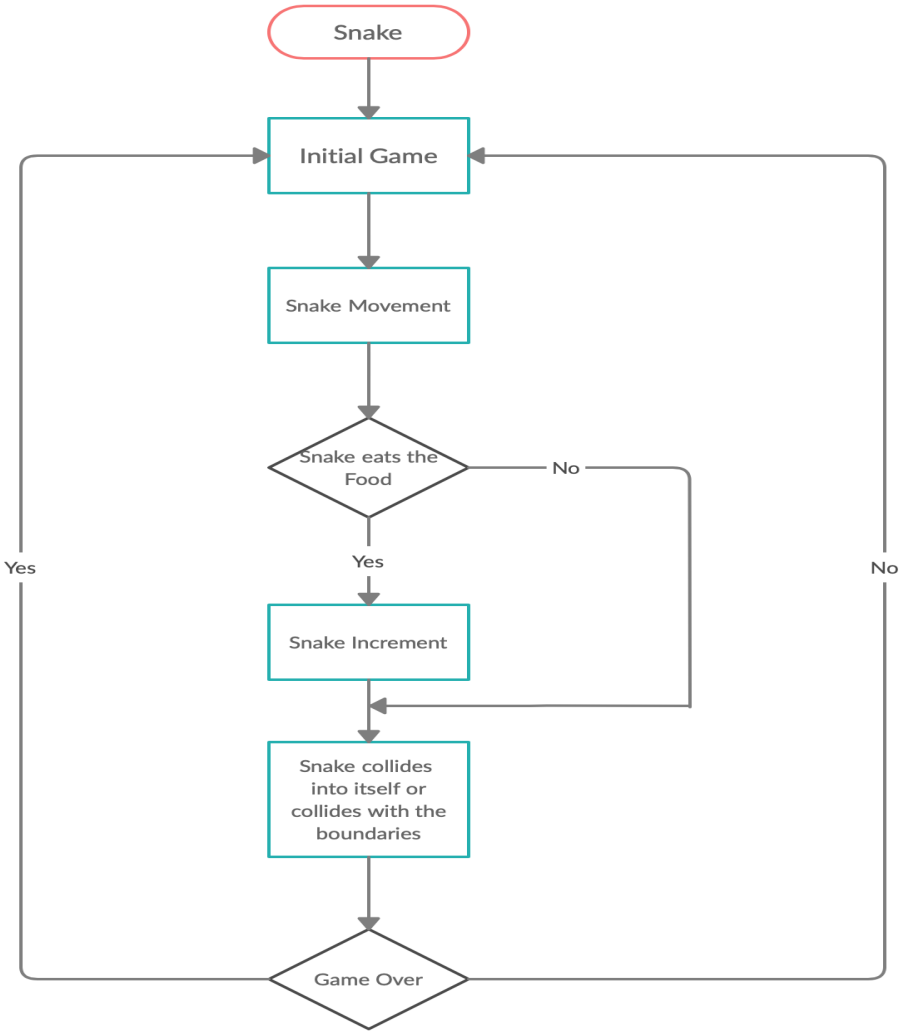
- **game_over():**
  This function is responsible for terminating the snake game either if the player collides the snake into its own tail or hits the boundaries of the game window. After terminating the game it shows a black screen saying ‘Game Over’ and it also shows the players final score.

- **show_score():**
  This function is responsible for showing the players score at the top left corner of the game window. It keeps incrementing the players score by one, every time the snake engulfs the food.

The following are the variables that have been used in the snake game:

1. snake_x
   Controls the length of the snake.
2. snake_y
   Controls the breadth of the snake.
3. food_x
   Controls the length of the food.
4. food_y
   Controls the breadth of the food.
5. frame_x
   This variable has been used to adjust the length of the game window.
6. frame_y
   This variable has been used to adjust the breadth of the game window

FLOWCHART


Snake
Initial Game
Snake Movement
Snake eats the Food
No
Yes
Yes
Snake Increment
Snake collides into itself or collides with the boundaries
Game Over
No

SOURCE CODE

```
import pygame
import sys
import time
import random

# Difficulty settings
# Easy       -  15
# Medium     -  30
# Hard       -  60

difficulty = 15

# Snake size
snake_x = 10
snake_y = 10

# Food size
food_x = 10
food_y = 10

# Window size
frame_size_x = 900
frame_size_y = 600

# Checks for errors encountered
check_errors = pygame.init()
if check_errors[1] > 0:
    print(f'[!] Had {check_errors[1]} errors when initializing the game, exiting...')
    sys.exit(-1)
else:
    print('Game successfully initialized')

# Initializing the game window
pygame.display.set_caption('Snake Game')
game_window = pygame.display.set_mode((frame_size_x, frame_size_y))

# 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)
```

```
# 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 function
def game_over():
    my_font = pygame.font.SysFont('times new roman', 90)
    game_over_surface = my_font.render('Game Over', 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, white, 'times', 60)
    pygame.display.flip()
    time.sleep(5)
    pygame.quit()
    sys.exit()

# Score function
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/8, 15)
    else:
        score_rect.midtop = (frame_size_x/2, frame_size_y/1.25)
    game_window.blit(score_surface, score_rect)

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

```
            sys.exit()
        elif event.type == pygame.KEYDOWN:
            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'
            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'

    # Movement of 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

    # Code for increasing the length of the Snake when it eats the food
    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

    # Game Graphics
    game_window.fill(black)
    for pos in snake_body:
        pygame.draw.rect(game_window, green, pygame.Rect(pos[0], pos[1], snake_x, snake_y))

    # Snake food
    pygame.draw.rect(game_window, red, pygame.Rect(food_pos[0], food_pos[1], food_x,
food_y))

    # Game Over conditions
    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()
    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', 40)

    # Refresh game screen
    pygame.display.update()

    # Refresh rate
    fps_controller.tick(difficulty)
```

SCREENSHOTS


Snake Game
Score : 0


Snake Game
Game Over
Score : 1

SYSTEM REQUIREMENTS

- AMD Athlon II or Intel Pentium IV
- 512 MB RAM
- Windows 7+
- Python 3.7+

BIBLIOGRAPHY

- Pygame Tutorials:
  https://www.youtube.com/watch?v=i6xMBig-pP4&list=PLzMcBGfZo4-lp3jAExUCewBfMx3UZFkh5

- Troubleshooting:
  https://stackoverflow.com

- *Computer Science with Python* - Sumita Arora

- Documentation:
  https://www.w3schools.com