

*“Think. Work. Serve.”*

Department: Electrical/ Computer Engineering

Course: Adv. Programming

Author(s): Ahmad Sedr

Title: Snake Game

Date: 16/11/2021

----------------------------------------------------------------------------------------------------

Evaluation:

|  |  |
| --- | --- |
| Sections | Points |
| A. Introduction (20) |  |
| B. Equipment (20) |  |
| C. Procedure (30) |  |
| D. Results and Discussion (30) |  |
| Bonus |  |

Overall:

Comments:

Table of Contents

[Introduction 3](#_Toc88090165)

[Code Explanation 3](#_Toc88090166)

[Libraries Used 3](#_Toc88090167)

[Initializing pygame Setup 3](#_Toc88090168)

[Creating Game variables and snake Body 4](#_Toc88090169)

[Main Game Loop 4](#_Toc88090170)

[Main Functions 6](#_Toc88090171)

[Score Function 6](#_Toc88090172)

[Game Over Function 7](#_Toc88090173)

[EXECUTION 7](#_Toc88090174)

[CONCLUSION 9](#_Toc88090175)

**Snake Game**

**Using: pygame-python**

# **Introduction**

In this task, I have studied and implemented the pygame module to create a basic game which has blocks on its screen which move continuously. I have controlled that blocks with the help of keyboard key to give a specific direction. A series of that blocks create snake body. In this project I have taken help from random library to generate random numbers during the execution of this game.

# **Code Explanation**

In this section I will go through the lines of my code to show how this game will work.

## **Libraries Used**

Following line of code shows the main libraries are being used in this project:

import pygame, sys, time, random

## **Initializing pygame Setup**

To initialize the game setup, I have defined two variables to fix the frame save of game window first and then I have initialized a pygame variable to check whether the game has been started successfully or not. Then I setup the display and moved to define some basic colors of the screen for throughout the game. Finally, I setup the clock to synch the game performance after a specified number of seconds:

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')

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()

## **Creating Game variables and snake Body**

In this section few basic settings are completed to show the snake body, food dots and the initial direction of the snake. These lists will be used in further steps below.

# 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

## **Main Game Loop**

This is the main logic part of this game. While this loop is true, the game will keep going. As it is set to false, game will stop. While game is working, it will always check few conditions. It will check which key of keyboard is pressed while the game is going. If Left, Right, Up or Down key if pressed then this loop will change some parameters to change the body direction and body size of the snake. Meanwhile it will check the position of the snake body, if it is close to the boundary lines the game will go to call game over function and also if the snake body collides with itself even then the game will call the game over function. Otherwise if the body is close to the food position then score variable will count one and the body size of snake is increase by increasing the dots of the series of body list. In this way the game will keep going.

# 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//20)) \* 10, random.randrange(1, (frame\_size\_y//20)) \* 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)

# **Main Functions**

Following functions are used during execution of main while loop. Let’s have a look at these functions

## **Score Function**

This function always updates the score and displays the string on screen

# Score

def show\_score(choice, color, font, size):

score\_font = pygame.font.SysFont(font, size)

score\_surface = score\_font.render('Ahmad 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()

## **Game Over Function**

This function when called closes the game frame and displays the score board and saves the result to a text file by writing into the file.

# 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(white)

game\_window.blit(game\_over\_surface, game\_over\_rect)

show\_score(0, red, 'times', 20)

pygame.display.flip()

doc = open("scores.txt","w+")

doc.write("Scores: "+str(score)+"\n")

doc.close()

time.sleep(3)

pygame.quit()

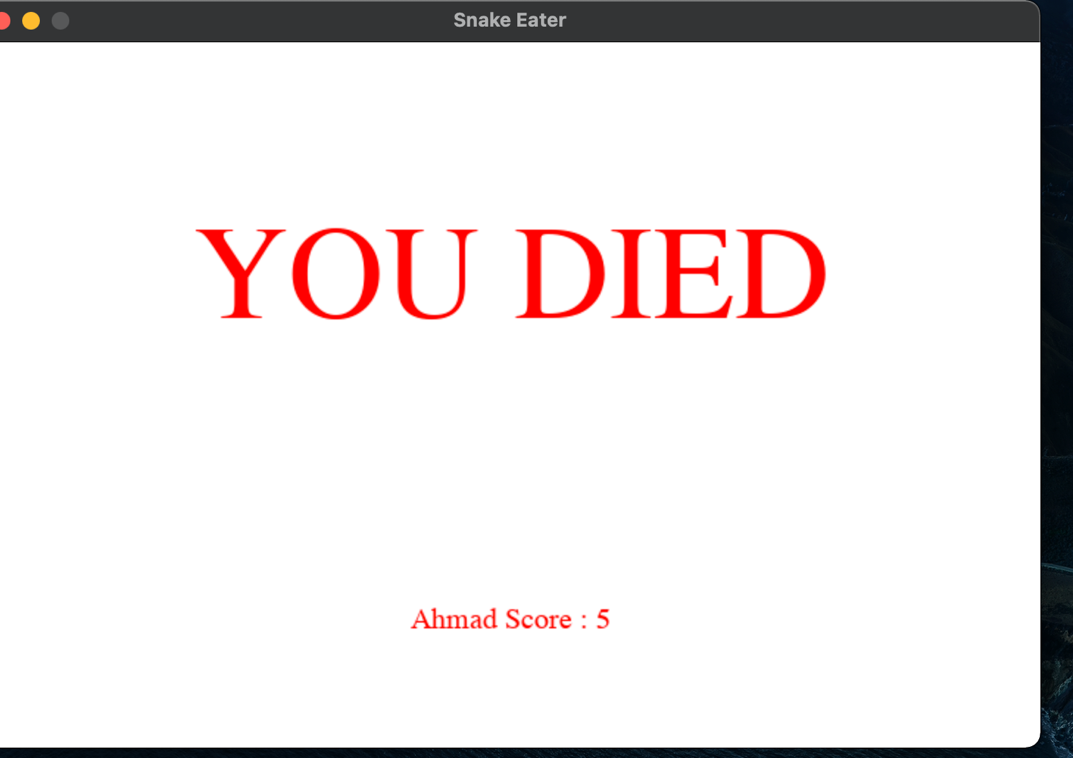
sys.exit()

# **EXECUTION**

Each screen shot below shows game interface at different position:







Parser = argparse.ArgumentParser()

parser.add\_argument (--id, type-str, default='anonymous', help='id')

parser. add\_argument ('--file', type-str, default='file.txt', help='file')

args = parser.parse\_args()

scorefile file.open(args.file, 'a')

scorefile.write(args.id + str(length\_of\_snake 1) + '\n')|

# **CONCLUSION**

In this project I learnt how to develop a game logic and I opened the doors for me in the world of game developers. I believe it was a good experience to learn this library which has interactive interface and easy to use tools