NIRMAL BHARTIA SCHOOL

2019-20

Submitted by: Akshay Warrier

COMPUTER SCIENCE PROJECT

Snake Game

**INDEX:**

1. CERTIFICATE
2. ACKNOWLEDGEMENT
3. INTRODUCTION
4. IMPORTANT DATA STRUCTURES INVOLVED
5. IMPORTANT LIBRARY FUNCTIONS USED
6. IMPORTANT FUNCTIONS
7. INPUT OUTPUT SCREENS
8. SOURCE CODE
9. BIBLOGRAPHY

**Certificate**

**Certified to be the bona fide work done by**

**Akshay Warrier**

**Of class XI**

**In COMPUTER SCIENCE during the**

**Year 2019-2020**

**Date: / /2020**

**Submitted for the CENTRAL BOARD OF SECONDARY EDUCATION**

**Examination held in the COMPUTER LAB at NIRMAL BHARTIA SCHOOL**

**Examiner:**

**Date:**

**Acknowledgement**

I am deeply indebted to my computer science teacher, Punam Ma’am.

Introduction

ABOUT THE PROJECT

* **Need for the project?**

**The main purpose of the project was to learn the module Pygame and also to increase my level of programming.**

* **What is the project about?**

**The project was about creating a game using python, so I created the snake game.**

**In the game you start as a snake with a length of one unit. You move around the grid to eat apples to increase your length, but there are also some apple with worms which once eaten reduce your lives by one. You lose once your snake’s head hits the body or you run out of lives.**

* **Flowchart depicting project flow**
* **Description of major modules.**

**Pygame- It is a module designed for creating video games.**

**Random- It is a standard python module used to generate random numbers or to select a random number from a sequence.**

**Time- It is a module which provides many ways of representing time in code, such as objects, numbers, and strings. It also provides functionality other than representing time, like waiting during code execution and measuring the efficiency of your code**

**IMPORTANT DATA STRUCTURES INVOLVED:**

**Important Variables:**

1. **run- A Boolean responsible for running the main game loop.**
2. **left, top- Lists that contain the x and y coordinates of each of snake’s body part.**
3. **left\_change, top\_change- Lists that contain the velocity of each of snake’s body part. Velocity is that value which is incremented in the snake’s position each iteration of the while loop.**
4. **n- A integer which determines the length of the snake.**
5. **north\_state, south\_state, east\_state, west\_state- These are Booleans which are set true whenever the user presses a arrow key in the corresponding direction.**
6. **north, south, east, west- These are integers which are used to iterate over the velocities of each of snakes body part. For example when north = 2, then the third body piece of the snake starts moving north irrespective of what direction it was previously moving in. These variables increase by 1 in each iteration of the while loop until they are equal to the length of the snake.**
7. **collision- It is a Boolean which is set true whenever the snake hits itself or its lives becomes equal to zero.**
8. **lives- It is the number of times the user can eat a apple with a worm before losing. It is a integer.**

**IMPORTANT LIBRARY FUNCTIONS USED:**

|  |  |  |
| --- | --- | --- |
| **MODULE NAME** | **LIBRARY FUNCTION** | **PURPOSE** |
| **pygame** | **display.set\_mode()**  **display.set\_caption()**  **display.update()**  **init()**  **blit()**  **font.Font()**  **render()**  **draw.rect()**  **image.load()**  **fill()**  **event.get()**  **event.type()** | **To create a window**  **To set the caption of window**  **To update the window**  **To initialize pygame**  **To display images on screen**  **To load a font**  **To render fonts**  **To draw rectangles**  **To load images**  **To colour the window**  **To get a list of all user inputs**  **To get the type of user input** |
| **random** | **randrange()** | **To generate a random number in the given range** |
| **time** | **sleep()** | **To pause the program for a certain amount of time** |
|  |  |  |
|  |  |  |

**Important Functions:**

( For all stand alone functions )

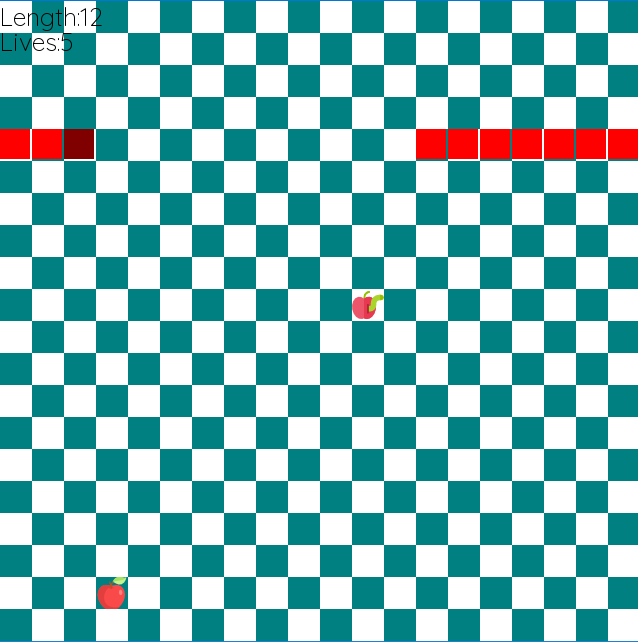
|  |  |  |  |
| --- | --- | --- | --- |
| **FUNCTION NAME** | **INPUTS** | **PURPOSE** | **RETURNS** |
| **body()** | (x,y) | It displays a red coloured rectangle at the given x and y coordinates which is part of the snake’s body. |  |
| **head()** | (x,y) | It is the head of the snake. It displays a dark red coloured rectangle at the given x and y coordinates. |  |
| **gameover\_text()**  **menu\_text()**  **start\_text()**  **show\_length()**  **length\_text()**  **restart\_text()**  **highscore\_text()**  **lives\_text()** |  | Displays text on the screen. |  |
| **grid()** |  | Displays a grid with a checkerboard pattern. |  |
| **food()**  **badfood()** | (x,y) | Displays an image of an apple at the given on the x and y coordinates. |  |
|  |  |  |  |

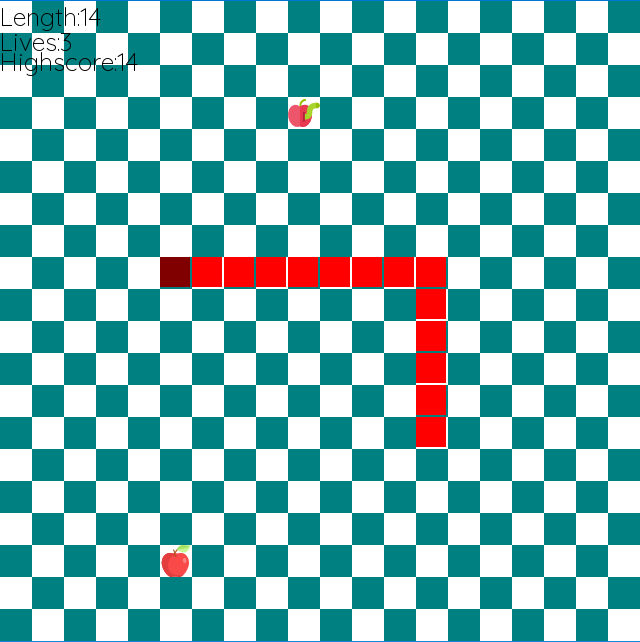
**Input Output Screens**

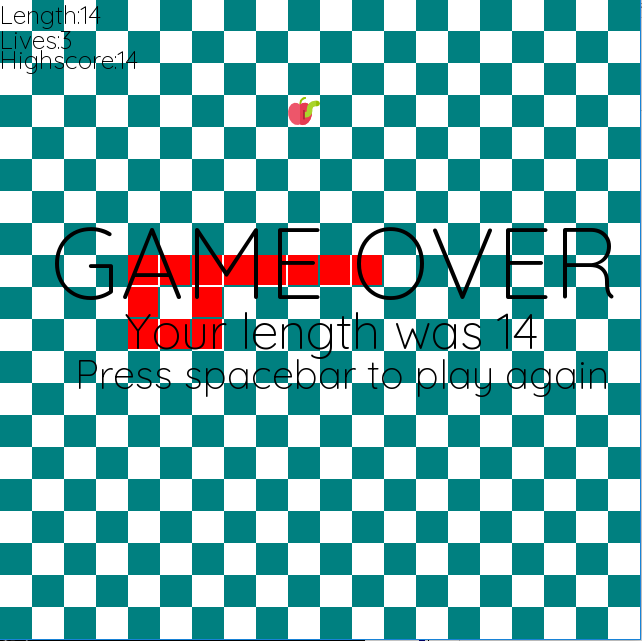
**Welcome Screen**



**Game play**



****

****

**Source code**

**SOURCE CODE**

import pygame as pg

import random

import time

pg.init()

win = pg.display.set\_mode((640, 640))

pg.display.set\_caption('Snake Game')

run = True

left = []

top = []

left\_change = []

top\_change = []

n = 10

for i in range(n):

top.append(320)

left.append(320 - 32 \* i)

left\_change.append(32)

top\_change.append(0)

x = 0.055

north = 0

south = 0

east = 0

west = 0

lives = 5

foodcount = 0

north\_state = False

south\_state = False

east\_state = False

west\_state = False

collision = False

highscore = []

start = False

restart = False

def body(x, y):

pg.draw.rect(win,(255,0,0),(x,y,30,30))

def head(x,y):

pg.draw.rect(win, (128, 0, 0), (x, y, 30, 30))

gameover\_font = pg.font.Font('Quicksand-VariableFont\_wght.ttf',100)

def gameover\_text():

gameover\_text = gameover\_font.render("GAME OVER",True,(0,0,0))

win.blit(gameover\_text,(50,200))

menu\_font = pg.font.Font('Quicksand-VariableFont\_wght.ttf',100)

def menu\_text():

menu\_text = menu\_font.render("SNAKE GAME",True,(0,0,0))

win.blit(menu\_text,(15,200))

start\_font = pg.font.Font('Quicksand-VariableFont\_wght.ttf',50)

def start\_text():

start\_text = start\_font.render("Press Spacebar To Start",True,(0,0,0))

win.blit(start\_text,(50,300))

show\_font = pg.font.Font('Quicksand-VariableFont\_wght.ttf',25)

def show\_length():

show\_text = show\_font.render("Length:"+str(n),True,(0,0,0))

win.blit(show\_text,(0,0))

length\_font = pg.font.Font('Quicksand-VariableFont\_wght.ttf',50)

def length\_text():

length\_text = length\_font.render("Your length was "+str(n), True, (0, 0, 0))

win.blit(length\_text, (125, 300))

restart\_font = pg.font.Font('Quicksand-VariableFont\_wght.ttf',40)

def restart\_text():

restart\_text = restart\_font.render("Press spacebar to play again", True, (0, 0, 0))

win.blit(restart\_text, (75, 350))

highscore\_font = pg.font.Font('Quicksand-VariableFont\_wght.ttf',25)

def highscore\_text():

highscore\_text = highscore\_font.render("Highscore:"+str(max(highscore)), True, (0, 0, 0))

win.blit(highscore\_text, (0, 45))

lives\_font = pg.font.Font('Quicksand-VariableFont\_wght.ttf',25)

def lives\_text():

lives\_text = lives\_font.render("Lives:"+str(lives), True, (0, 0, 0))

win.blit(lives\_text, (0, 25))

def grid():

for i in range(20):

if i%2 == 0:

x = 0

else:

x = 1

for j in range(20):

if j % 2 == x:

pg.draw.rect(win,(255,255,255),(32\*i,32\*j,32,32))

else:

pg.draw.rect(win, (0,128,128), (32 \* i, 32 \* j, 32, 32))

def food(x,y):

foodimg = pg.image.load('apple.png')

win.blit(foodimg,(x,y))

def badfood(x,y):

badfoodimg = pg.image.load('worm.png')

win.blit(badfoodimg,(x,y))

foodx = random.randrange(32,608,32)

foody = random.randrange(32,608,32)

badfoodx = random.randrange(32,608,32)

badfoody = random.randrange(32,608,32)

while run:

pg.display.update()

win.fill((0, 128, 128))

if start == False:

menu\_text()

start\_text()

else:

grid()

show\_length()

lives\_text()

if highscore != []:

if n > max(highscore):

highscore.append(n)

highscore\_text()

#create snake

if start == True:

for i in range(n):

time.sleep(x/n)

top[i] += top\_change[i]

left[i] += left\_change[i]

if i == 0:

head(left[i], top[i])

else:

body(left[i],top[i])

# make boundary

for i in range(n):

if left[i] > 640:

left[i] -= 704

elif left[i] < 0:

left[i] += 704

elif top[i] > 640:

top[i] -= 704

elif top[i] < 0:

top[i] += 704

#If the head of the snake has touched the body then end the game

for coordinates in zip(left[1:],top[1:]):

if coordinates == (left[0],top[0]):

collision = True

break

if lives == 0:

collision = True

if collision == True:

for i in range(n):

left\_change[i] = 0

top\_change[i] = 0

foodx = 6000

foody = 6000

gameover\_text()

length\_text()

restart\_text()

#movement

if collision == False and start == True:

if south == n:

south = 0

south\_state = False

elif south\_state == True:

top\_change[south] = 32

left\_change[south] = 0

south +=1

if north == n:

north = 0

north\_state = False

elif north\_state == True:

top\_change[north] = -32

left\_change[north] = 0

north +=1

if east == n:

east = 0

east\_state = False

elif east\_state == True:

top\_change[east] = 0

left\_change[east] = 32

east +=1

if west == n:

west = 0

west\_state = False

elif west\_state == True:

top\_change[west] = 0

left\_change[west] = -32

west +=1

if start == True:

food(foodx,foody)

badfood(badfoodx,badfoody)

#check if food is eaten

if (foodx,foody) == (left[0],top[0]):

foodx = random.randrange(32, 608, 32)

foody = random.randrange(32, 608, 32)

left\_change.append(left\_change[-1])

top\_change.append(top\_change[-1])

if top\_change[-1] == 32:

top.append(top[-1]- 32)

left.append(left[-1])

elif top\_change[-1] == -32:

top.append(top[-1] + 32)

left.append(left[-1])

if left\_change[-1] == 32:

top.append(top[-1])

left.append(left[-1]- 32)

if left\_change[-1] == -32:

top.append(top[-1])

left.append(left[-1] + 32)

n = n+1

if (badfoodx,badfoody) == (left[0],top[0]):

badfoodx = random.randrange(32, 608, 32)

badfoody = random.randrange(32, 608, 32)

lives -= 1

for event in pg.event.get():

if event.type == pg.QUIT:

run = False

if event.type == pg.KEYDOWN:

if event.key == pg.K\_SPACE:

start = True

if event.key == pg.K\_SPACE and collision == True:

start = False

collision = False

lives = 5

highscore.append(n)

n = 1

left\_change = [left\_change[0],]

top\_change = [top\_change[0],]

left = [left[0],]

top = [top[0],]

north = 0

south = 0

east = 0

west = 0

foodx = random.randrange(32,608,32)

foody = random.randrange(32,608,32)

badfoodx = random.randrange(32,608,32)

badfoody = random.randrange(32,608,32)

if event.key == pg.K\_DOWN and (top\_change[0]!=-32):

south\_state = True

elif event.key == pg.K\_UP and (top\_change[0]!=32):

north\_state = True

elif event.key == pg.K\_RIGHT and (left\_change[0]!=-32):

east\_state = True

elif event.key == pg.K\_LEFT and (left\_change[0]!=32):

west\_state = True

**bibliography**

**(None)**

**\*\*\*\*\*\*\*\*\*\*\*End of Report\*\*\*\*\*\*\*\***