Snake Game

**Introduction: -**

This game has been around since the earliest days of home computing and has reemerged in recent years on platforms Our Game written by python 3.

The Game give player competitive environment by arranged the players and display the high score

**GAME: -**

The game we will be building is a simple Snake game. Based on python3. In this game you control an ever-growing snake. You turn the snake head left and right and try to avoid the head hitting the rest of the body or the border.

We are going to start building the game with simple graphics and look some attractive.

The game based in python libraries like:

* tkinter
* turtle
* Time
* random
* PIL

Contents

[1. Graphic user interface 3](#_Toc81262272)

[2. File manipulation 5](#_Toc81262273)

[3. Snake body and food 6](#_Toc81262274)

[4. Snake Move functions 7](#_Toc81262275)

[5. Main game loop 8](#_Toc81262276)

[I. Snake collision with border 8](#_Toc81262277)

[II. Snake ate it self 9](#_Toc81262278)

[III. Snake eat the food 9](#_Toc81262279)

[IV. Make the snake body follow the head 10](#_Toc81262280)

[V. Check for save the high score 10](#_Toc81262281)

[6. Sounds in the game 11](#_Toc81262282)

**Game ingredients: -**

# **Graphic user interface**

Graphical user interface

Description automatically generated

* We had used tkinter library to setup the gui of the game.
* import tkinter as tk
* import turtle
* import time
* import random
* delay = 0.1
* #code of the background
* from PIL import ImageTk
* from PIL import Image
* root = tk.Tk()
* root.title('Snake Game')
* root.state('zoomed')
* The game has main background changing when the player rich score 100
* # Frame Desert (fram no 1)
* dayImage = Image.open('desert.jpg')
* dayImage = dayImage.resize((width,height),Image.ANTIALIAS)
* resized1 = ImageTk.PhotoImage(dayImage)
* # Frame Jungle (fram no 2)
* nightImage = Image.open('desert.jpg')
* nightImage = nightImage.resize((width,height),Image.ANTIALIAS)
* resized2 = ImageTk.PhotoImage(nightImage)
* backgroundLabel = tk.Label(image=resized1)
* backgroundLabel.place\_configure(x=0,y=0)
* In the middle of the main window there is a canvas to be the snake area
* #Create a turtle screen
* canvas = tk.Canvas(root,width=600, height=600)
* turtle\_screen=turtle.TurtleScreen(canvas)
* turtle\_screen.bgcolor('#ffe6cc')
* canvas.pack(pady=70)
* Display the player score in the right of the snake area
* # SCORE
* global score
* score = 0
* # Display Score
* def display\_Score():
* w =tk.Label(root,text="Score: {}".format(score),
* font=('Arial',40,'bold'),
* fg='white',
* bg='#135370',
* bd=0,
* padx=50,
* pady=10,
* compound='bottom')
* w.place(x=1100,y=100)
* Display the high score that had reached in the game
* # Display High Score
* def display\_H\_score():
* w =tk.Label(root,text="High Score: {}".format(high\_score.read\_H\_score()),
* font=('Arial',30,'bold'),
* fg='#135370',
* bd=0,
* padx=50,
* pady=10,
* compound='bottom')
* w.place(x=1100,y=200)

# **File manipulation**

The game store scores of the players in text file for arrange them by the score and display the high score in the screen.

We made a class named High\_Score with two methods on for write scores in the file and second for read data from it

class High\_score:

    # Write the score

    def write\_score(self, sc):

        write = open("High Score.txt","w")

        write.write(str(sc))

        write.close()

    # Read High Score

    def read\_H\_score(self):

        self.read = open("High Score.txt","r")

        for line in self.read:

            self.score\_str = int(line)

        self.read.close()

        return self.score\_str

# **Snake body and food**

The snake body consist of two part

The head with in the front of the snake

# Snake head

head = turtle.RawTurtle(turtle\_screen)

head.shape("circle")

head.color("#33691e")

head.penup()

head.goto(0,0)

head.direction = "stop"

and the teal consist of segments

theirs function named add\_segmant its role is adding segment in the snake when it eats on food

# Add segment

def add\_segment():

    new\_segment = turtle.RawTurtle(turtle\_screen)

    new\_segment.speed(0)

    new\_segment.shape("circle")

    new\_segment.color('#76ff03')

    new\_segment.penup()

    segments.append(new\_segment)

snake food it’s looks like red circles display in random places when the snake eat it grow with one segment

# Snake food

food = turtle.RawTurtle(turtle\_screen)

food.speed(0)

food.shape("circle")

food.color("red")

food.penup()

food.goto(0,100)

# **Snake Move functions**

Snake move based on group of functions each on define direction

def go\_up():

    head.direction = "up"

def go\_down():

    head.direction = "down"

def go\_left():

    head.direction = "left"

def go\_right():

    head.direction = "right"

then their function called move it’s role is make the snake move in the direction witch has been defined.

By make the snake head move positive or negative 20 in x axis and y axis

def move():

    if head.direction == "up":

        y = head.ycor()

        head.sety(y + 20)

    if head.direction == "down":

        y = head.ycor()

        head.sety(y - 20)

    if head.direction == "left":

        x = head.xcor()

        head.setx(x - 20)

    if head.direction == "right":

        x = head.xcor()

        head.setx(x + 20)

then we setup the keyboard buttons roles make

“W” to go up “S” to go down

“A” to go left “D” to go right

# Keyboard bingings

turtle\_screen.listen()

turtle\_screen.onkeypress(go\_up,"w")

turtle\_screen.onkeypress(go\_down,"s")

turtle\_screen.onkeypress(go\_right,"d")

turtle\_screen.onkeypress(go\_left,"a")

# **Main game loop**

In the main loop of the games there is the ways to finish the game this ways is

## Snake collision with border

Here we check if the snake collision the border of the canvas or not it collision the border the snake area will turn into red color and the score will be displayed then reset the snake head position , canvas color and segments to zero

# Check for a collision with border

    if head.xcor()>290 or head.xcor()<-290 or head.ycor()>290 or head.ycor()<-290:

        turtle\_screen.bgcolor('red')

        time.sleep(1)

        head.goto(0,0)

        head.direction = "stop"

        turtle\_screen.bgcolor('#ffe6cc')

        score = 0

        display\_Score()

        # Hide the segment list

        for segment in segments:

            segment.goto(1000,1000)

        # Clear the segment list

            segments = []

## Snake ate it self

Check for head collision with the body segments (eat tour self) here if the snake head collision one of the body segment the function will do the same like the previous

# Check for head collision with the body segments (Eat yourself)

    for index, segment in enumerate(segments):

        if(index > 1):

            # Check for collision with segment

            if head.distance(segment) < 20:

                turtle\_screen.bgcolor('red')

                time.sleep(1)

                head.goto(0,0)

                head.direction = "stop"

                # Hide the segment list

                for segment in segments:

                  segment.goto(1000,1000)

                # Clear the segment list

                segments = []

                score = 0

                turtle\_screen.bgcolor('#ffe6cc')

                display\_Score()

## Snake eat the food

when the snake collision the food node the game will make new food node by random module and we will call the function add\_segment to add new segment to the snake and add 10 to the score

# Check for collision with food

    if head.distance(food) < 20:

        # Move the food to a random spot

        x = random.randint(-200, 200)

        y = random.randint(-200, 200)

        food.goto(x,y)

        # Add a segment

        if score == 0:

            add\_segment()

        add\_segment()

        score += 10

        display\_Score()

## Make the snake body follow the head

Loop to make each segment in the body follow the head in continuity

# Move the eng segment first in reverse order

    for index in range(len(segments)-1, 0, -1):

        x = segments[index-1].xcor()

        y = segments[index-1].ycor()

        segments[index].goto(x,y)

    # Move segment 0 to where the head is

    if len(segments) > 0:

        x = head.xcor()

        y = head.ycor()

        segments[0].goto(x,y)

## Check for save the high score

Check is the player score is the high score or not and display if true store it in the file else ignore it

# Check High Score

    if high\_score.read\_H\_score() <= score:

        high\_score.write\_score(score)

        display\_H\_score()

# **Sounds in the game**

The Game make sound when the snake eat an apple

# eat sound

def eat\_music():

    mixer.music.load("eat (online-audio-converter.com).wav")

    mixer.music.play()

If player cross the high score, then die by collision the border or eat your self the screen will turn on green and give wining sound

# Win sound

def Win\_music():

    mixer.music.load("win (online-audio-converter.com).wav")

    mixer.music.play()

If you die by collision the border or eat your self the screen will turn on red and give losing alarm

# Game over sound

def game\_over\_music():

    mixer.music.load("game Over (online-audio-converter.com).wav")

    mixer.music.play()