B.Tech Program First Year

Course: Experiential Learning

Course Code: DA1001

## “Snake Game”

by

**Akhilesh Kolambekar**

(Reg. No: 209309110)

Under the guidance

of

**Mr. Vijay Prakash Sharma**

**Assistant Professor (Senior Scale)**

Name of Dept and School, Manipal University Jaipur, Jaipur (Raj.).

**Department of Information Technology**

**School of Computing and Information Technology**

**Faculty of Engineering**

**Manipal University Jaipur, India**

June, 2021

# Certificate

This is to certify that the project titled **“Snake Game”** is a record of the bona fide work done by Akhilesh Kolambekar(Reg No:209309110) submitted for the partial fulfilment of the requirements for the completion of the Experiential Learning (DA1001) course in the Department of Information Technology of Manipal University Jaipur, during the academic session July-November 2021.

|  |
| --- |
| *Signature of the mentor* |
| Mr. Vijay Prakash Sharma  Assistant Professor (Senior Scale)  Department of Information Technology |
| *Signature of the HoD* |
| Dr. Pankaj Vyas  Head of the Department  Department of Information Technology |

**Abstract**

I have developed a snake game from scratch using the turtle module in python. It is a two player game, that is each player controls a snake and the players compete with each other. The snakes can be controlled using “w, a, s, d” and “i, j, k, l”. This game has powerups which are: speed increase, speed decrease, invert controls and teleport. The powerups spawn after every 15 seconds. In this version of game, even if the snakes collide with the border or with their own grown body, the game doesn’t end. There is no time limit and the game only ends when both the snake heads collide with each other.

**Introduction**

**Python**

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible and much Python 2 code does not run unmodified on Python 3. Python 2 was discontinued with version 2.7.18 in 2020.

Python consistently ranks as one of the most popular programming languages.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective

**Turtle Module**

Turtle graphics is a popular way for introducing programming to kids. It was part of the original Logo programming language developed by Wally Feurzeig, Seymour Papert and Cynthia Solomon in 1967. The turtle module is an extended reimplementation of the same-named module from the Python standard distribution up to version Python 2.5.

It tries to keep the merits of the old turtle module and to be (nearly) 100% compatible with it. This means in the first place to enable the learning programmer to use all the commands, classes and methods interactively when using the module from within IDLE run with the -n switch.

The turtle module provides turtle graphics primitives, in both object-oriented and procedure-oriented ways. Because it uses tkinter for the underlying graphics, it needs a version of Python installed with Tk support.

**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.

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, other obstacle, or itself.

In 2017, Google released their version of the game as an easter egg, whenever the phrases "snake", "play snake", "snake game" and "snake video game" are typed. The game has multiple customization options, such as snake color, speed, number of fruits, and appearance of fruits. Multiple modes are also present.

**Module to be designed**

I have chosen snake game as my topic but it will be a slight variation of it. Its main specialty is that it is a two player game. There are going to be powerups so as to make it a bit more interesting. The game does not end until and unless both the snake heads collide, that is you have infinite lives. This makes the game more interesting and increases the suspense as it is anybody’s game.

**Literature Review**

The Snake design dates back to the arcade game Blockade, developed and published by Gremlin in 1976. It was cloned as Bigfoot Bonkers the same year. In 1977, Atari released two Blockade-inspired titles: the arcade game Dominos and Atari VCS game Surround. Surround was one of the nine Atari VCS (later the Atari 2600) launch titles in the United States and was also sold by Sears under the name Chase. That same year, a similar game was launched for the Bally Astrocade as Checkmate.

The first known personal computer version, titled Worm, was programmed in 1978 by Peter Trefonas of the US on the TRS-80, and published by CLOAD magazine in the same year. This was followed shortly afterwards with versions from the same author for the Commodore PET and Apple II. A microcomputer clone of the Hustle arcade game, itself a clone of Blockade, was written by Peter Trefonas in 1979 and published by CLOAD. An authorized version of Hustle was published by Milton Bradley for the TI-99/4A in 1980. In 1982's Snake for the BBC Micro, by Dave Bresnen, the snake is controlled using the left and right arrow keys relative to the direction it is heading in. The snake increases in speed as it gets longer, and there's only one life; one mistake means starting from the beginning.An analog joystick-controlled variant of Snake, called Anaconda, was included as a hidden mini game in Time Splitters 2, which featured free rotation instead of a fixed four direction system, and multiple types of food.

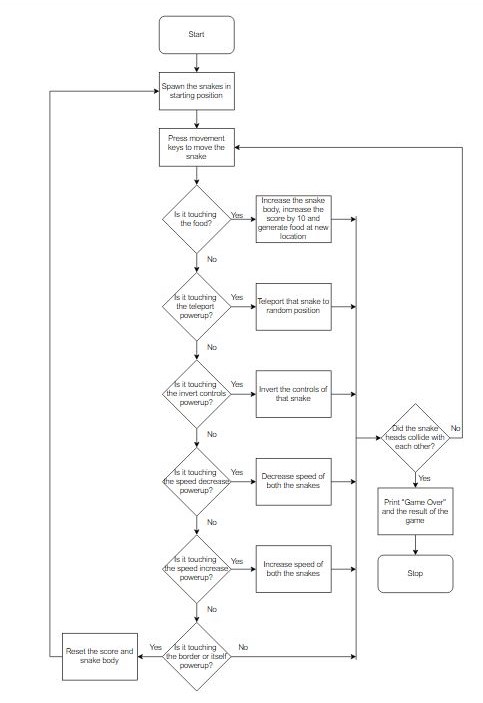
From thereon, Snake became an important part of mobile gaming history. The game was addictive and appropriate for people of all ages. It was like the Game of Life, but for the mobile phone user. Nokia went on to evolve Snake, and some would argue, ruin it. If you owned a Nokia 1600, you would remember Snake Xenzia, which was similar to Snake 2, but meant for monochrome displays.

The first real improvement in graphics came with a game called Snake EX, which kept all the key features of Snake 2, but with improved graphics. You got a top-down view of a green snake eating bugs, while you navigate the thing around. The snake could even open its mouth when eating now.

Things were fine till this point, when Nokia came up with its failed n-Gage platform. With it, the company decided to continue making Snake a part of its game offerings. However, many would argue that Snakes, the game meant for the n-Gage platform was the downfall of Snake.

Made in 3D, you now got a third person view of the Snake, following behind it at is it ate psychedelic looking bugs, in a psychedelic world. While Snakes was still addictive, for many, it just didn’t live up to what they were used to from the iconic game. Nokia made a new version to this, later, called Snake Subsonic, but that didn’t make much of a difference either. Snakes was pre-loaded on the Nokia N70, N73 and N80 smartphones.

Snake can be played on Youtube videos that use the 2010 version of the player. When a video is selected and starts buffering, the user can press the “left key”, ”right key”, ”home key”, ”end key” for any period of time before immediately pressing the “up key”. The game will appear and can be played in the video screen. Until 2012, a version called Old Snake could be played inside Gmail

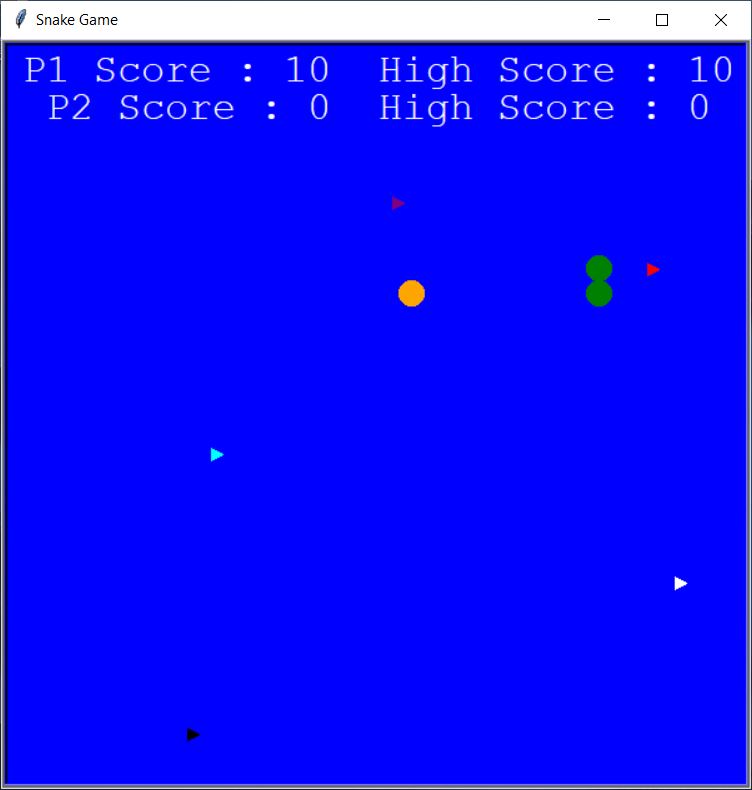
**Methodology/Flowchart and its description**

The game loads with the two snakes at a fixed position. To start the game, you either of the player has to press one of the movement keys (w, a, s, d and/or i, j, k, l). Once the movement keys are pressed, the program continuously checks whether any one of the snake is touching the food. If the snake is touching the food then the body of snake is increased, score is increased by 10 and the food is sent to a new location. If the snake is not touching the food then the program checks whether the snake is touching the teleport powerup. If it is then the snake that touched the powerup will be teleported to a random location. If it is not then the program checks whether the snake is touching the invert controls powerup. If it is touching then the controls of snake that touched that powerup gets its controls inverted. If it is not touching then it checks whether the snake is touching the speed decrease powerup. If the snake is touching then the speed of both the snakes is decreased else the program checks whether the snake is touching the speed increase powerup. If it is touching the speed increase powerup then speed of both the snakes increased. If the snake is not touching the speed increase powerup then the program checks whether the snake is touching the border or its own grown body. If it is touching then the snakes body is reset, the score is set to 0 and the snake is sent back to its original spawn location for a fresh run. If the snake is not touching itself or the border then the program checks whether bot the snake heads collide with each other. If they did collide then “Game Over” is print on the screen along with the results. If they did not collide then the entire process is repeated again.

**Results and Discussion**

****

This is the first screen that will appear when the game will start. The position of the snakes is fixed every time a new game is started but the food will keep changing its location. At first there is only food and snakes, the powerups start spawning after 15 seconds into the game and will keep changing their location if not eaten before 15 seconds. The Score and High Score will be set to 0 at the start by default.



As you can see here, as soon as I ate the food, the P1 Score went up by 10 and High Score is updated. After eating the food, the food went to a different location and the snake body is also increase by 1. For the grown body, the logic that I used is that the head moves forward and segment which is next to head (first segment) takes the head’s position, the second segment takes the first segments position and so on.



After colliding with the border, the snake lost its grown body and went back to its original position. The score was also set back to 0. Since I ate 3 foods maximum before colliding, the High Score is updated to 30. The food and powerups remain in the same position as before colliding. For the powerups, I have used the “ontimer” function which spawns the powerup after 15 seconds and then used a “for loop” to keep spawning the powerups at regular intervals (in this case it is 15 seconds)



After giving it another go, I was able to eat 5 food, hence my High Score was updated to 50. Since both the snake heads collided, the game ended. The “Game Over” text is displayed and since player 1’s high score is more than player 2’s high score, “Player 1 wins” text is displayed. If player 2’s high score was more than player 1’s high score then “Player 2 wins” will be displayed on the screen.



If the high scores of both the players are tied then instead of “Player 1 wins” , “Tie” will be displayed as shown in picture. After the results are displayed you will have to completely close the game and start again in order to play a new game.

**Conclusion and Future Work**

**Conclusion**

Snake Game is a gaming project that aims to bring the simplicity and fun of the traditional snake game with some new and advanced features. The game incorporates two players competing with each other to gain a higher score along with a bunch of powerups to make it more interesting. The coding of Snake was extremely difficult with many errors arising. Many systems had to be written numerous ways before a final working solution was found. The coding of powerups was challenging as well because it was causing a lot of glitches in the game but in the end I made it work fine. It is recommended that anyone who wishes to recreate this game starts simply when writing the code. It is advisable that they first perfect the snakes movement controls before messing with the food generation and the powerups.

**Future Work**

In the future, I would like to make the game much smoother than now. I would like to add sound and music to it as well. I would also like to make it aesthetically more pleasing by using Pygame. I would implement an online mode in the game, which will allow more than just two players to play the game and may add a much more challenging gameplay. I would also like to make it available for the mobile users so that people can play it on the go. I would also like to implement computer controlled snakes. I would also like to add a leaderboard which will keep track of the highest scores in the game.

**References**

1. Tieturi 2/1985 ISSN 0780-9778

2. "Other Snake Games". Archived from the original on August 18, 2016. Retrieved July 5, 2016.

3. Gerard Goggin (2010), Global Mobile Media, Taylor & Francis, p. 101, ISBN 978-0-415-46917-3, retrieved April 7, 2011

4. Rusel DeMaria & Johnny L. Wilson (2003). High score!: the illustrated history of electronic games (2 ed.). McGraw-Hill Professional. p. 24. ISBN 0-07-223172-6. Retrieved April 7, 2011.

5. "Blockade video game, Gremlin Ind, inc. (1976)". Arcade-history.com. April 4, 2008. Archived from the original on June 11, 2011. Retrieved November 4, 2011.

6. "Retrogaming Times Monthly 7". My.stratos.net. January 1, 2005. Archived from the original on September 22, 2011. Retrieved November 4, 2011.

7. <https://www.python.org/doc/essays/blurb/>

8. <https://docs.python.org/3/library/turtle.html>

9. [www.wikipedia.com](http://www.wikipedia.com)

10. https://www.digit.in/features/mobile-phones/a-brief-history-of-snake-33913.html

**Acknowledgment/Source Code**

**Acknowledgement**

I take this opportunity to present my votes of thanks to all those guidepost who really acted as lightening pillars to enlighten our way throughout this project that has led to successful completion of this project. I am highly thankful of our sir Vijay Prakash Sharma for their active support, valuable time and advice, whole-hearted guidance and sincere cooperation during the completion of project.

**Source Code**

# -\*- coding: utf-8 -\*-

"""

Created on Sat Apr 3 16:12:23 2021

@author: Akhilesh

"""

import turtle

import time

import random

score1 = 0

high\_score1 = 0

score2 = 0

high\_score2 = 0

delay = 0.08

win = turtle.Screen()

win.title("Snake Game")

win.bgcolor("blue")

win.setup(width=600,height = 600)

win.tracer(0)

snakehead1 = turtle.Turtle()

snakehead1.shape("circle")

snakehead1.color("green")

snakehead1.penup()

snakehead1.goto(-25,100)

snakehead1.speed(0)

snakehead1.direction = "stop"

snakehead2 = turtle.Turtle()

snakehead2.shape("circle")

snakehead2.color("orange")

snakehead2.penup()

snakehead2.goto(25,100)

snakehead2.speed(0)

snakehead2.direction = "stop"

def movement1():

if snakehead1.direction == "up":

y = snakehead1.ycor()

snakehead1.sety(y+20)

if snakehead1.direction == "down":

y = snakehead1.ycor()

snakehead1.sety(y-20)

if snakehead1.direction == "right":

x = snakehead1.xcor()

snakehead1.setx(x+20)

if snakehead1.direction == "left":

x = snakehead1.xcor()

snakehead1.setx(x-20)

def movement2():

if snakehead2.direction == "up":

y = snakehead2.ycor()

snakehead2.sety(y+20)

if snakehead2.direction == "down":

y = snakehead2.ycor()

snakehead2.sety(y-20)

if snakehead2.direction == "right":

x = snakehead2.xcor()

snakehead2.setx(x+20)

if snakehead2.direction == "left":

x = snakehead2.xcor()

snakehead2.setx(x-20)

def go\_up1():

if snakehead1.direction != "down":

snakehead1.direction = "up"

def go\_down1():

if snakehead1.direction != "up":

snakehead1.direction = "down"

def go\_right1():

if snakehead1.direction != "left":

snakehead1.direction = "right"

def go\_left1():

if snakehead1.direction != "right":

snakehead1.direction = "left"

def go\_up2():

if snakehead2.direction != "down":

snakehead2.direction = "up"

def go\_down2():

if snakehead2.direction != "up":

snakehead2.direction = "down"

def go\_right2():

if snakehead2.direction != "left":

snakehead2.direction = "right"

def go\_left2():

if snakehead2.direction != "right":

snakehead2.direction = "left"

win.listen()

win.onkey(go\_up1, "w")

win.onkey(go\_down1, "s")

win.onkey(go\_right1, "d")

win.onkey(go\_left1, "a")

win.onkey(go\_up2, "i")

win.onkey(go\_down2, "k")

win.onkey(go\_right2, "l")

win.onkey(go\_left2, "j")

food = turtle.Turtle()

food.speed(0)

food.shape("triangle")

food.color("red")

food.penup()

food.shapesize(0.50, 0.50)

food.goto(random.randint(-280,280),random.randint(-280,220))

speed\_decrease = turtle.Turtle()

speed\_decrease.speed(0)

speed\_decrease.shape("triangle")

speed\_decrease.color("white")

speed\_decrease.penup()

speed\_decrease.shapesize(0.50, 0.50)

speed\_decrease.goto(1000,1000)

speed\_decrease.hideturtle()

def speed\_down():

speed\_decrease.showturtle()

speed\_decrease.goto(random.randint(-280,280),random.randint(-280,220))

for i in range(50):

win.ontimer(speed\_down,t = 15000\*(i+1))

speed\_increase = turtle.Turtle()

speed\_increase.speed(0)

speed\_increase.shape("triangle")

speed\_increase.color("cyan")

speed\_increase.penup()

speed\_increase.shapesize(0.50, 0.50)

speed\_increase.goto(1000,1000)

speed\_increase.hideturtle()

def speed\_up():

speed\_increase.showturtle()

speed\_increase.goto(random.randint(-280,280),random.randint(-280,220))

for i in range(50):

win.ontimer(speed\_up,t = 15000\*(i+1))

teleport = turtle.Turtle()

teleport.speed(0)

teleport.shape("triangle")

teleport.color("purple")

teleport.penup()

teleport.shapesize(0.50, 0.50)

teleport.goto(1000,1000)

teleport.hideturtle()

def tel\_power():

teleport.showturtle()

teleport.goto(random.randint(-280,280),random.randint(-280,220))

for i in range(50):

win.ontimer(tel\_power,t = 15000\*(i+1))

invert = turtle.Turtle()

invert.speed(0)

invert.shape("triangle")

invert.color("black")

invert.penup()

invert.shapesize(0.50, 0.50)

invert.hideturtle()

def inv\_power():

invert.showturtle()

invert.goto(random.randint(-280,280),random.randint(-280,220))

for i in range(50):

win.ontimer(inv\_power,t = 15000\*(i+1))

segments1 = []

segments2 = []

pen1 = turtle.Turtle()

pen1.speed(0)

pen1.shape("square")

pen1.color("white")

pen1.penup()

pen1.hideturtle()

pen1.goto(0, 260)

pen1.write("P1 Score : 0 High Score : 0",align = "center",font = ("Courier",24,"normal"))

pen2 = turtle.Turtle()

pen2.speed(0)

pen2.shape("square")

pen2.color("white")

pen2.penup()

pen2.hideturtle()

pen2.goto(0, 230)

pen2.write("P2 Score : 0 High Score : 0",align = "center",font = ("Courier",24,"normal"))

end = turtle.Turtle()

end.speed(0)

end.shape("square")

end.color("white")

end.penup()

end.hideturtle()

end.goto(0, 100)

result = turtle.Turtle()

result.speed(0)

result.shape("square")

result.color("white")

result.penup()

result.hideturtle()

result.goto(0, 0)

while True:

win.update()

if snakehead1.distance(food)<15:

x = random.randint(-280,280)

y = random.randint(-280,220)

food.goto(x,y)

new\_segment1 = turtle.Turtle()

new\_segment1.speed(0)

new\_segment1.shape("circle")

new\_segment1.color("green")

new\_segment1.penup()

segments1.append(new\_segment1)

score1 += 10

if score1> high\_score1:

high\_score1 = score1

pen1.clear()

pen1.write("P1 Score : {} High Score : {}".format(score1,high\_score1),align = "center",font = ("Courier",24,"normal"))

if snakehead2.distance(food)<15:

x = random.randint(-280,280)

y = random.randint(-280,220)

food.goto(x,y)

new\_segment2 = turtle.Turtle()

new\_segment2.speed(0)

new\_segment2.shape("circle")

new\_segment2.color("orange")

new\_segment2.penup()

segments2.append(new\_segment2)

score2 += 10

if score2> high\_score2:

high\_score2 = score2

pen2.clear()

pen2.write("P2 Score : {} High Score : {}".format(score2,high\_score2),align = "center",font = ("Courier",24,"normal"))

elif snakehead1.distance(teleport)<15:

time.sleep(0.35)

x = random.randint(-250,250)

y = random.randint(-250,220)

snakehead1.goto(x,y)

teleport.hideturtle()

elif snakehead2.distance(teleport)<15:

time.sleep(0.35)

x = random.randint(-250,250)

y = random.randint(-250,220)

snakehead2.goto(x,y)

teleport.hideturtle()

elif snakehead1.distance(invert)<15:

win.onkey(go\_up1, "s")

win.onkey(go\_down1, "w")

win.onkey(go\_right1, "a")

win.onkey(go\_left1, "d")

invert.hideturtle()

elif snakehead2.distance(invert)<15:

win.onkey(go\_up2, "k")

win.onkey(go\_down2, "i")

win.onkey(go\_right2, "j")

win.onkey(go\_left2, "l")

invert.hideturtle()

elif snakehead1.distance(speed\_decrease)<15:

delay = 0.15

speed\_decrease.hideturtle()

elif snakehead2.distance(speed\_decrease)<15:

delay = 0.15

speed\_decrease.hideturtle()

elif snakehead1.distance(speed\_increase)<15 :

delay = 0.05

speed\_increase.hideturtle()

elif snakehead2.distance(speed\_increase)<15 :

delay = 0.05

speed\_increase.hideturtle()

for index in range(len(segments1)-1, 0, -1):

x1 = segments1[index-1].xcor()

y1 = segments1[index-1].ycor()

segments1[index].goto(x1, y1)

for index in range(len(segments2)-1, 0, -1):

x2 = segments2[index-1].xcor()

y2 = segments2[index-1].ycor()

segments2[index].goto(x2, y2)

if len(segments1)> 0:

x1 = snakehead1.xcor()

y1 = snakehead1.ycor()

segments1[0].goto(x1, y1)

if len(segments2)> 0:

x2 = snakehead2.xcor()

y2 = snakehead2.ycor()

segments2[0].goto(x2, y2)

if snakehead1.xcor() > 290 or snakehead1.xcor() < -290 or snakehead1.ycor() > 290 or snakehead1.ycor() < -290:

time.sleep(0.1)

snakehead1.goto(-25, 100)

snakehead1.direction = "stop"

for segment in segments1:

segment.goto(1000, 1000)

segments1.clear()

score1 = 0

delay = 0.08

win.onkey(go\_up1, "w")

win.onkey(go\_down1, "s")

win.onkey(go\_right1, "d")

win.onkey(go\_left1, "a")

pen1.clear()

pen1.write("P1 Score : {} High Score : {}".format(score1,high\_score1),align = "center",font = ("Courier",24,"normal"))

if snakehead2.xcor() > 290 or snakehead2.xcor() < -290 or snakehead2.ycor() > 290 or snakehead2.ycor() < -290:

time.sleep(0.1)

snakehead2.goto(25, 100)

snakehead2.direction = "stop"

for segment2 in segments2:

segment2.goto(1000, 1000)

segments2.clear()

score2 = 0

delay = 0.08

win.onkey(go\_up2, "i")

win.onkey(go\_down2, "k")

win.onkey(go\_right2, "l")

win.onkey(go\_left2, "j")

pen2.clear()

pen2.write("P2 Score : {} High Score : {}".format(score2,high\_score2),align = "center",font = ("Courier",24,"normal"))

movement1()

movement2()

for segment1 in segments1:

if segment1.distance(snakehead1) < 20:

time.sleep(0.1)

snakehead1.goto(-25, 100)

snakehead1.direction = "stop"

for segment1 in segments1:

segment1.goto(1000, 1000)

segments1.clear()

score1 = 0

delay = 0.08

win.onkey(go\_up1, "w")

win.onkey(go\_down1, "s")

win.onkey(go\_right1, "d")

win.onkey(go\_left1, "a")

pen1.clear()

pen1.write("P1 Score : {} High Score : {}".format(score1,high\_score1),align = "center",font = ("Courier",24,"normal"))

for segment2 in segments2:

if segment2.distance(snakehead2) < 20:

time.sleep(0.1)

snakehead2.goto(25, 100)

snakehead2.direction = "stop"

for segment2 in segments2:

segment2.goto(1000, 1000)

segments2.clear()

score2 = 0

delay = 0.08

win.onkey(go\_up2, "i")

win.onkey(go\_down2, "k")

win.onkey(go\_right2, "l")

win.onkey(go\_left2, "j")

pen2.clear()

pen2.write("P2 Score : {} High Score : {}".format(score2,high\_score2),align = "center",font = ("Courier",24,"normal"))

time.sleep(delay)

if snakehead1.distance(snakehead2)<15:

snakehead1.goto(-25,100)

snakehead2.goto(25,100)

snakehead1.direction = "stop"

snakehead2.direction = "stop"

end.write("Game Over",align = "center",font = ("Courier",30,"normal"))

if high\_score1==high\_score2:

result.write("Tie",align = "center",font = ("Courier",30,"normal"))

elif high\_score2>high\_score1:

result.write("Player 2 wins",align = "center",font = ("Courier",30,"normal"))

elif high\_score1>high\_score2:

result.write("Player 1 wins",align = "center",font = ("Courier",30,"normal"))

break

win.mainloop()

turtle.done()