**1.Introduction**

Snake game is one of the most popular arcade games of all time. Snake was an incredibly popular game, mostly remembered from the 1990s era cell phones. For many people, it was the only game on their phone at that time. In the most basic form, it's a super simple game, but still strangely entertaining. It's also a great game to learn the basics of game making from. In this game, the main objective of the player is to catch the maximum number of fruits without hitting the wall or itself. Creating a snake game can be taken as a challenge while learning Python or Pygame. It is one of the best beginner-friendly projects that every novice programmer should take as a challenge. Learning to build a video game is kind of interesting and fun learning. We will be using [pygame](https://www.geeksforgeeks.org/introduction-to-pygame/) to create this snake game. **Pygame** is an open-source library that is designed for making video games. It has inbuilt graphics and sounds libraries. It is also beginner-friendly, and cross-platform. The snake in the Snake game is controlled using the four direction buttons relative to the direction it is headed in.

This snake game is done in Python 3.

So, we will be creating a Python-based-game using the following modules:

* Pygame: This is one of the python modules which helps us in creation of snake game.

We can install it by running the command: pip install pygame

* Time: This function is used to count the number of seconds elapsed since the epoch.
* Random: This function is used to generate random numbers in Python by using random module.

The **Snake Game** with Source Code is a single-player game where the player must get a food that pop out in the board window. The game is very crucial and fun in every aspect, the player must get a food in order to receive a score. The game was built using a turtle module where it is a simple drawing module that draw an object and command it to move forward. This is played in a simple window with background where it contains the snake head and the food. The player can move the snake head via keyboard bindings. This game is a score-based game in which the player must attain more score from time to time. The **Snake Game** is simple but a very addictive arcade style game where your goal is to maintain your score and protect your snake.

**2.Importing modules**

**2.1 Pygame Module**

Python is the most popular programming language or nothing wrong to say that it is the next-generation programming language. In every emerging field in computer science, Python makes its presence actively. Python has vast libraries for various fields such as Machine Learning (Numpy, Pandas, Matplotlib), Artificial intelligence (Pytorch, TensorFlow), and Game development (Pygame,Pyglet). Pygame is a cross-platform set of Python modules which is used to create video games.It consists of computer graphics and sound libraries designed to be used with the Python programming language. The pygame package represents the top-level package for others to use. Pygame itself is broken into many submodules, but this does not affect programs that use Pygame.

As a convenience, most of the top-level variables in pygame have been placed inside a module named ‘pygame.locals’. This is meant to be used with ‘from [pygame.localspygame constants](http://man.hubwiz.com/docset/PyGame.docset/Contents/Resources/Documents/ref/locals.html" \l "module-pygame.locals" \o ") import \*’, in addition to ‘import pygame’.

When you ‘import pygame’ all available pygame submodules are automatically imported. Be aware that some of the pygame modules are considered “optional”, and may not be available. In that case, Pygame will provide a placeholder object instead of the module, which can be used to test for availability.

**Table2.1.1:Pygame Methods**

|  |  |  |
| --- | --- | --- |
| [pygame.init](http://man.hubwiz.com/docset/PyGame.docset/Contents/Resources/Documents/ref/pygame.html#pygame.init) | — | initialize all imported pygame modules |
| [pygame.quit](http://man.hubwiz.com/docset/PyGame.docset/Contents/Resources/Documents/ref/pygame.html#pygame.quit) | — | uninitialize all pygame modules |
| [pygame.error](http://man.hubwiz.com/docset/PyGame.docset/Contents/Resources/Documents/ref/pygame.html#pygame.error) | — | standard pygame exception |
| [pygame.get\_error](http://man.hubwiz.com/docset/PyGame.docset/Contents/Resources/Documents/ref/pygame.html#pygame.get_error) | — | get the current error message |
| [pygame.set\_error](http://man.hubwiz.com/docset/PyGame.docset/Contents/Resources/Documents/ref/pygame.html#pygame.set_error) | — | set the current error message |
| [pygame.get\_sdl\_version](http://man.hubwiz.com/docset/PyGame.docset/Contents/Resources/Documents/ref/pygame.html#pygame.get_sdl_version) | — | get the version number of SDL |
| [pygame.get\_sdl\_byteorder](http://man.hubwiz.com/docset/PyGame.docset/Contents/Resources/Documents/ref/pygame.html#pygame.get_sdl_byteorder) | — | get the byte order of SDL |
| [pygame.register\_quit](http://man.hubwiz.com/docset/PyGame.docset/Contents/Resources/Documents/ref/pygame.html#pygame.register_quit) | — | register a function to be called when pygame quits |
| [pygame.encode\_string](http://man.hubwiz.com/docset/PyGame.docset/Contents/Resources/Documents/ref/pygame.html#pygame.encode_string) | — | Encode a unicode or bytes object |
| [pygame.encode\_file\_path](http://man.hubwiz.com/docset/PyGame.docset/Contents/Resources/Documents/ref/pygame.html#pygame.encode_file_path) | — | Encode a unicode or bytes object as a file system path |

**2.2 Time Module**

The Python time module provides many ways of representing time in code, such as objects, [numbers](https://realpython.com/python-numbers/), and strings. It also provides functionality other than representing time, like waiting during code execution and measuring the efficiency of your code. A Python program can handle date and time in several ways. Converting between date formats is a common chore for computers. Python's time and calendar modules help track dates and times.

An explanation of some terminology and conventions is in order.

* The epoch is the point where the time starts and is platform dependent. For Unix, the epoch is January 1, 1970, 00:00:00 (UTC). To find out what the epoch is on a given platform, look at time.gmtime(0).
* The term seconds since the epoch refers to the total number of elapsed seconds since the epoch, typically excluding [leap seconds](https://en.wikipedia.org/wiki/Leap_second). Leap seconds are excluded from this total on all POSIX-compliant platforms.
* The functions in this module may not handle dates and times before the epoch or far in the future. The cut-off point in the future is determined by the C library; for 32-bit systems, it is typically in 2038.
* Function [**strptime()**](https://docs.python.org/3/library/time.html#time.strptime) can parse 2-digit years when given %y format code. When 2-digit years are parsed, they are converted according to the POSIX and ISO C standards: values 69–99 are mapped to 1969–1999, and values 0–68 are mapped to 2000–2068.
* The precision of the various real-time functions may be less than suggested by the units in which their value or argument is expressed. E.g., on most Unix systems, the clock “ticks” only 50 or 100 times a second.
* On the other hand, the precision of [**time()**](https://docs.python.org/3/library/time.html#time.time)and [**sleep()**](https://docs.python.org/3/library/time.html#time.sleep) is better than their Unix equivalents: times are expressed as floating point numbers, [**time()**](https://docs.python.org/3/library/time.html#time.time) returns the most accurate time available (using Unix **gettimeofday()** where available), and [**sleep()**](https://docs.python.org/3/library/time.html#time.sleep) will accept a time with a nonzero fraction (Unix **select()** is used to implement this, where available).
* The time value as returned by [**gmtime()**](https://docs.python.org/3/library/time.html#time.gmtime), [**localtime()**](https://docs.python.org/3/library/time.html#time.localtime), and [**strptime()**](https://docs.python.org/3/library/time.html#time.strptime), and accepted by [**asctime()**](https://docs.python.org/3/library/time.html#time.asctime), [**mktime()**](https://docs.python.org/3/library/time.html#time.mktime) and [**strftime()**](https://docs.python.org/3/library/time.html#time.strftime), is a sequence of 9 integers. The return values of [**gmtime()**](https://docs.python.org/3/library/time.html#time.gmtime), [**localtime()**](https://docs.python.org/3/library/time.html#time.localtime), and [**strptime()**](https://docs.python.org/3/library/time.html#time.strptime) also offer attribute names for individual fields.

**2.3 RANDOM MODULE**

This module implements pseudo-random number generators for various distributions. For integers, there is uniform selection from a range. For sequences, there is uniform selection of a random element, a function to generate a random permutation of a list in-place, and a function for random sampling without replacement.

On the real line, there are functions to compute uniform, normal (Gaussian), lognormal, negative exponential, gamma, and beta distributions. For generating distributions of angles, the von Mises distribution is available.

Almost all module functions depend on the basic function [**random()**](https://docs.python.org/3/library/random.html#random.random), which generates a random float uniformly in the semi-open range [0.0, 1.0). Python uses the Mersenne Twister as the core generator. It produces 53-bit precision floats and has a period of 2\*\*19937-1. The underlying implementation in C is both fast and threadsafe. The Mersenne Twister is one of the most extensively tested random number generators in existence. However, being completely deterministic, it is not suitable for all purposes, and is completely unsuitable for cryptographic purposes.

The functions supplied by this module are actually bound methods of a hidden instance of the [random.Random](https://docs.python.org/3/library/random.html" \l "random.Random" \o "random.Random) class. You can instantiate your own instances of [Random](https://docs.python.org/3/library/random.html#random.Random) to get generators that don’t share state.

Class [Random](https://docs.python.org/3/library/random.html#random.Random) can also be subclassed if you want to use a different basic generator of your own devising: in that case, override the **random()**, **seed()**, **getstate()**, and **setstate()** methods. Optionally, a new generator can supply a **getrandbits()** method — this allows **[randrange()](https://docs.python.org/3/library/random.html" \l "random.randrange" \o "random.randrange)** to produce selections over an arbitrarily large range.

**2.4 Sys Module**

The sys module in Python provides various functions and variables that are used to manipulate different parts of the Python runtime environment. It allows operating on the interpreter as it provides access to the variables and functions that interact strongly with the interpreter.The **sys** module provides system specific parameters and functions sys.argv,sys.executable,sys.exit,sys.modules,sys.path,sys.platform,sys.stdin/stdout/stderr

**3.Working**

**3.1 Game Window Initialization and settings**

# Difficulty settings  
# Easy -> 10  
# Medium -> 25  
# Hard -> 40  
# Harder -> 60  
# Impossible-> 120  
difficulty = 10  
# Window size  
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')  
  
# Initialise game window  
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()  
  
# 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

**3.2 Functions game\_over and show\_score**

# 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(black)  
 game\_window.blit(game\_over\_surface, game\_over\_rect)  
 show\_score(0, red, 'times', 20)  
 pygame.display.flip()  
 time.sleep(3)  
 pygame.quit()  
 sys.exit()  
  
# Score  
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 / 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()

**3.3 Main Logic**

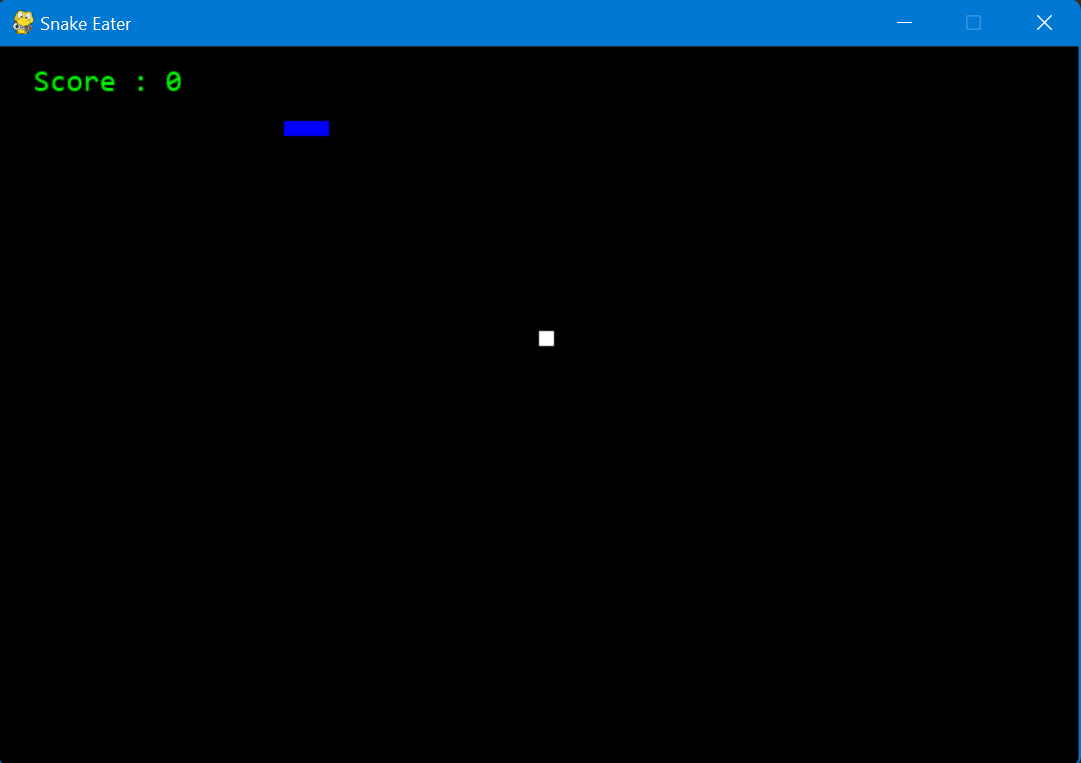
while True:  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 pygame.quit()  
 sys.exit()  
 # Whenever a key is pressed down  
 elif event.type == pygame.KEYDOWN:  
 # W -> Up; S -> Down; A -> Left; D -> Right  
 if event.key == pygame.K\_UP or event.key == ord('w'):  
 change\_to = '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))  
 # pause logic  
 if event.key == pygame.K\_SPACE:  
 pause = True  
  
 while pause==True:  
 for event in pygame.event.get():  
 if event.type == pygame.KEYDOWN:  
 if event.key == pygame.K\_SPACE:  
 pause = False  
 # 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 // 10)) \* 10, random.randrange(1, (frame\_size\_y // 10)) \* 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, blue, 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, green, 'consolas', 20)  
 # Refresh game screen  
 pygame.display.update()  
 # Refresh rate  
 fps\_controller.tick(difficulty)

**4.Implementation**

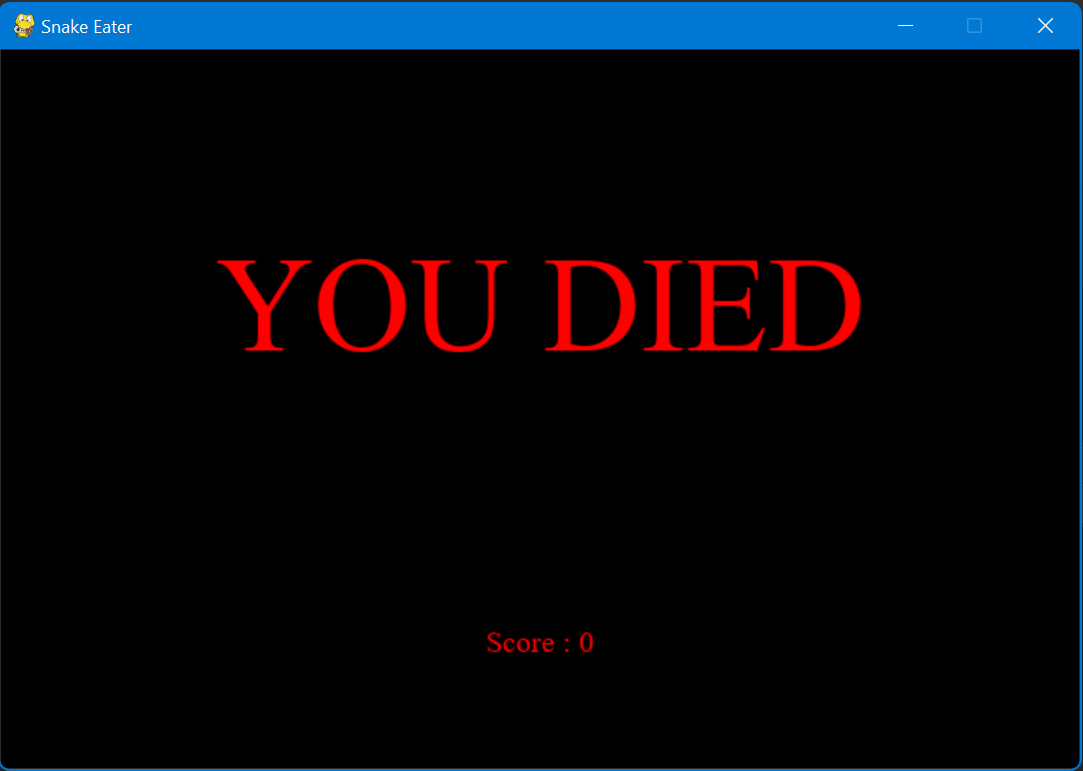
**4.1 Source code**

import pygame  
import sys  
import time  
import random  
  
# Difficulty settings  
# Easy -> 10  
# Medium -> 25  
# Hard -> 40  
# Harder -> 60  
# Impossible-> 120  
difficulty = 10  
# Window size  
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')  
  
# Initialise game window  
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()  
  
# 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  
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(black)  
 game\_window.blit(game\_over\_surface, game\_over\_rect)  
 show\_score(0, red, 'times', 20)  
 pygame.display.flip()  
 time.sleep(3)  
 pygame.quit()  
 sys.exit()  
  
  
# Score  
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 / 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()  
  
  
# 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))  
 # pause logic  
 if event.key == pygame.K\_SPACE:  
 pause = True  
   
 while pause==True:  
 for event in pygame.event.get():  
 if event.type == pygame.KEYDOWN:  
 if event.key == pygame.K\_SPACE:  
 pause = False  
 # 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 // 10)) \* 10, random.randrange(1, (frame\_size\_y // 10)) \* 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, blue, 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, green, 'consolas', 20)  
 # Refresh game screen  
 pygame.display.update()  
 # Refresh rate  
 fps\_controller.tick(difficulty)

**5.Results**

****

**5.1:Snake Game Window**

****

**5.2:Snake Game Termination**

**6.Conclusion:**

* This game gives us more thrilling, frustrating, and gives us more pleasure.
* It helps us in many sectors like- planning, designing, developing, managing, programming skill, socket programming and so on.
* The snake game gives us clear idea on turtle module.
* This will give more fun as it is simple and easily understood by everyone.

**7.References:**

* [**https://itsourcecode.com/free-projects/python-projects/snake-game-in-python-code/**](https://itsourcecode.com/free-projects/python-projects/snake-game-in-python-code/)
* [**https://www.edureka.co/blog/snake-game-with-pygame/**](https://www.edureka.co/blog/snake-game-with-pygame/)
* [**https://www.youtube.com/watch?v=9F8zD42hQCA**](https://www.youtube.com/watch?v=9F8zD42hQCA)
* [**https://www.pygame.org/news**](https://www.pygame.org/news)