**A PROJECT REPORT**

**ON**

**SNAKE GAME USING PYTHON**

***Submitted by***

**SHREY JAIN (1/18/FET/BCS/092)**

***Under the Guidance of***

**DR. SUMAN BHATIA**

**ASSOCIATE PROFESSOR**

***in partial fulfillment for the award of the degree of***

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**



**Faculty of Engineering & Technology**

**Manav Rachna International Institute of Research and Studies, Faridabad**

**May 2021**

**ACKNOWLEDGEMENT**

The successful realization of project is an outgrowth of a consolidated effort of people from desperate fronts. We are thankful to **DR. SUMAN BHATIA** (Professor/Associate Professor/Assistant Professor) for his/her variable advice and support extended to us without which we could not be able to complete our project for a success.

We are thankful to **Ms. Shubhangi Srivastava,** Project Coordinator, Assistant Professor, CSE department for her guidance and support.

We express our deep gratitude to **Prof.** (**Dr.) Supriya P. Panda**, Head of Department (CSE) for her endless support and affection towards us. Her constant encouragement has helped to widen the horizon of our knowledge and inculcate the spirit of dedication to the purpose.

We would like to express our sincere gratitude to **Prof. (Dr.) Pardeep Kumar**, PVC & Dean, FET for providing us the facilities in the Institute for completion of our work.

Words cannot express our gratitude for all those people who helped us directly or indirectly in our Endeavour. We take this opportunity to express our sincere thanks to all staff members of CSE department for the valuable suggestion and also to our family and friends for their support.

SHREY JAIN (1/18/FET/BCS/092)

*SAIF KHAN (1/18/FET/BCS/066)*

*ANSHUL BHARDWAJ (1/18/FET/BCS/080)*

**Declaration**

We hereby declare that this project report entitled “**SNAKE GAME USING PYTHON**

” *by* **SHREY JAIN (1/18/FET/BCS/092), *SAIF KHAN (1/18/FET/BCS/066), ANSHUL BHARDWAJ (1/18/FET/BCS/080)*** being submitted in partial fulfillment of the requirements for the degreeof Bachelor of Technology in **Computer Science & Engineering** under Faculty of Engineering & Technology of Manav Rachna International Institute of Research and Studies, Faridabad, during the academic year 2020-2021 is a bonafide record of our original work carried out under the guidance **DR. SUMAN BHATIA, PROFESSOR, FET**

We further declare that we have not submitted the matter presented in this Project for the award of any other Degree/Diploma of this University or any other University/Institute.

1. SHREY JAIN (1/18/FET/BCS/092)
2. *SAIF KHAN (1/18/FET/BCS/066)*
3. *ANSHUL BHARDWAJ (1/18/FET/BCS/080)*



**Manav Rachna International Institute of Research and Studies, Faridabad**

**Faculty of Engineering & Technology**

**Department of Computer Science and Engineering**

MAY,2021

**Certificate**

This is to certify that this project report entitled “**SNAKE GAME USING PYTHON**” *by*

**SHREY JAIN (1/18/FET/BCS/092), *SAIF KHAN (1/18/FET/BCS/066), ANSHUL BHARDWAJ (1/18/FET/BCS/080*),** submitted in partial fulfillment of the requirements for the degreeof Bachelor of Technology in **COMPUTER SCIENCE** under Faculty of Engineering & Technology of Manav Rachna International Institute of Research and Studies, Faridabad, during the academic year 2020-2021, is a bonafide record of work carried out under my guidance and supervision. I hereby declare that the work has been carried out under my supervision and has not been submitted elsewhere for any other purpose.

**(DR. SUMAN BHATIA) (DR SUPRIYA PANDEY)**

Dr suman Bhatia Dr. Supriya Panda

Professor Head of Department

Department of Computer Science and Engineering Department. of Computer Science and Engineering

FET, MRIIRS, Faridabad FET, MRIIRS, Faridabad

**Role and responsibility of Team members-:**

* Window Screen -: Shrey Jain
* Head of snake -: Saif Khan
* Food in the Game -: Anshul Bharadwaj
* Assigning Key Direction-: Saif & Shrey
* Main Gameplay-: Saif, Shrey & Anshul

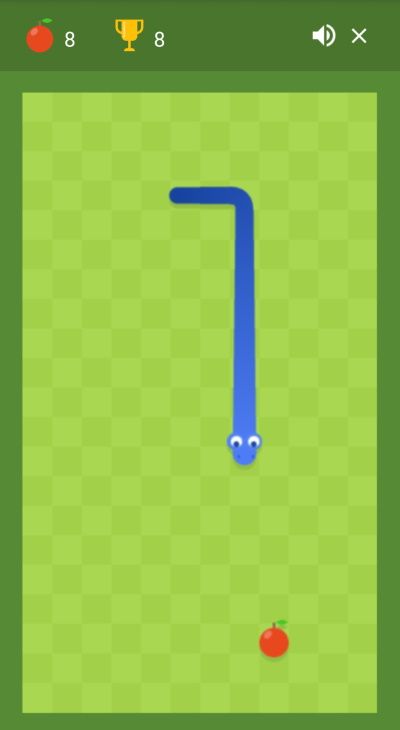
**Abstract :-**

A snake game is an **arcade maze game** which has been developed by Gremlin Industries and published by Sega in October 1976. It is considered to be a skilful game and has popularized among people for generations. The objective of this python project is to build a snake game project. In this python project, the player has to move a snake so it touches the fruit. If the snake touches itself or the border of the game then the game will over. Snake game is one of the most popular arcade games of all time. 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 kinda 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.

**INTRODUCTION :-**

Snake is a video game which originated during the late 1970s in arcades. To be precise, it is a concept, without details. Therefore there is no definitive version of the game. As a result, there are various versions of implementations for this game. Figure 1 shows a version of Snake. After it became the standard pre-loaded game on Nokia mobile phones in 1998, there was a resurgence of interest in the game as it found a larger audience. Thus, the Nokia version is used as the reference for the design of our Snake.



With a quick glance, most people would take this game as a simple game, which does not require sophisticated strategies. Admittedly, it is quite easy for the player at earlier stage, as paths are obvious and obstacles are few. While, it does require advanced strategies when the ‘snake’ is long enough, due to inconspicuous solutions and increasing number of obstacles(the body of the ‘snake’ itself is also consider as obstacles, and with the game going, it increases, which means more obstacles are generated).

**Objectives :-**

* The snake in the Snake game is controlled using the four direction buttons relative to the direction it is headed in.
* The player’s objective in the game is to achieve maximum points as possible by collecting food or fruits.
* The player loses once the snake hits the wall or hits itself.

**Problem definition:-**

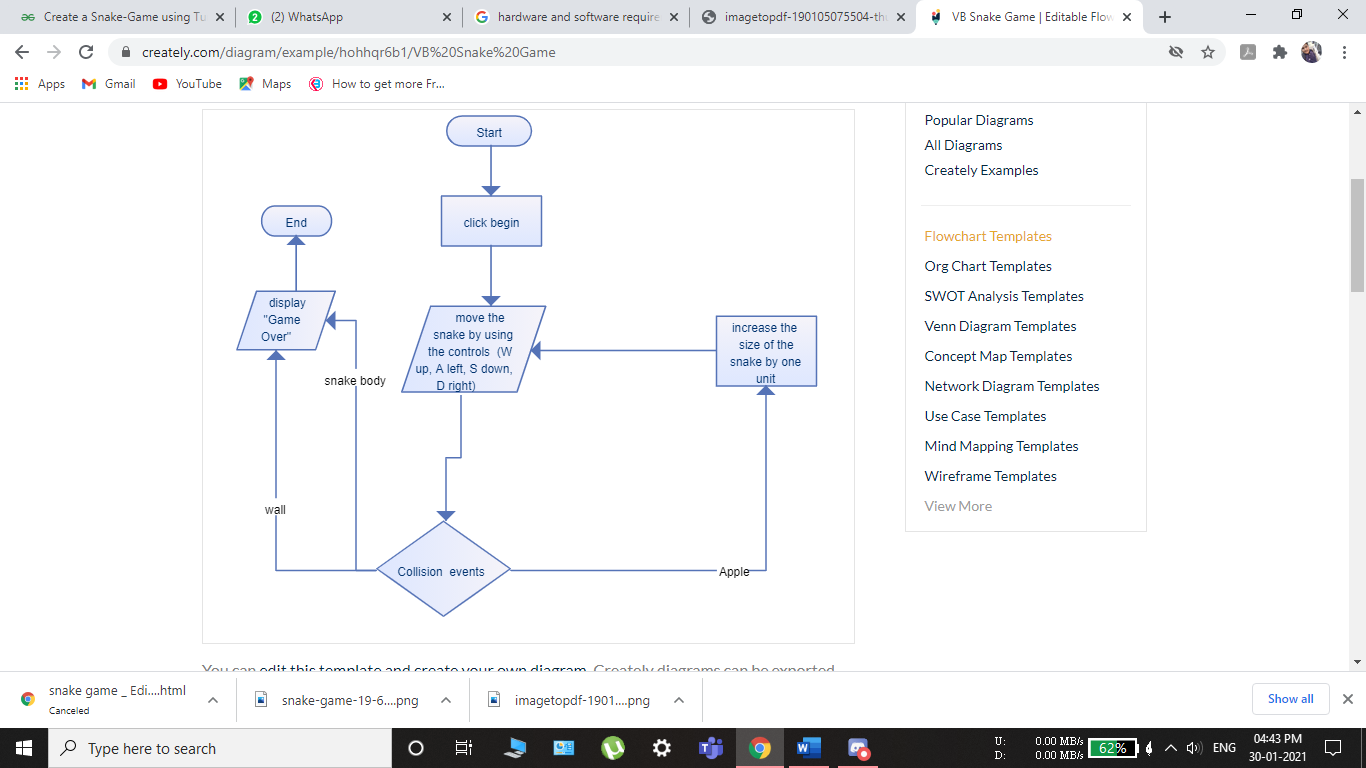
 we have created a **Python-based-game** using the following modules:

* **Turtle:** It is a pre-installed python library that enables users to create shapes and pictures by providing them a virtual canvas.
* **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.
* **PyGame Module:**

|  |  |
| --- | --- |
| * Function | * Description |
| * init() | * Initializes all of the imported Pygame modules (returns a tuple indicating success and failure of initializations) |
| * display.set\_mode() | * Takes a tuple or a list as its parameter to create a surface (tuple preferred) |
| * update() | * Updates the screen |
| * quit() | * Used to uninitialize everything |
| * set\_caption() | * Will set the caption text on the top of the display screen |
| * event.get() | * Returns list of all events |
| * Surface.fill() | * Will fill the surface with a solid color |
| * time.Clock() | * Helps track time time |
| * font.SysFont() | * Will create a Pygame font from the System font resources |

**Proposed System:-**

* **Flowchart of Snake Game**



* To move the snake, use 'up arrow' for up, 'down arrow' for down, 'left arrow for left and right arrow' for right.
* Again, there are constants you can change if you want to alter these settings. Press 'ESC' to exit the game at any time.
* The aim of the game is to collect the dots (food) and avoid the obstacles (crosses, borders, and the snake itself). As you collect food, the snake gets longer, so increasing your likelihood of crashing into yourself.

**REQUIREMENTS:**

SOFTWARE REQUIREMENTS:

OPERATING SYSTEM :-WINDOWS

APPLICATION SOFTWRE :-JUPYTER NOTEBOOK (PYTHON)

TOOLKIT :-TURTLE & PYGAME

HARDWARE REQUIREMENT:

HARDISK :- 32 GB

RAM :- 128 MB

PROCESSOR :- ANY PENTIUM VERSION

**PROJECT SCOPE :**

Our project will be able to implement in future after making some changes and modifications as we make our project at a very low level. So the modifications that can be done in our project are:

1. It will bring back the old classic game memories and will not harm children as much as games are harming these days such as PubG it is not a addictive game.

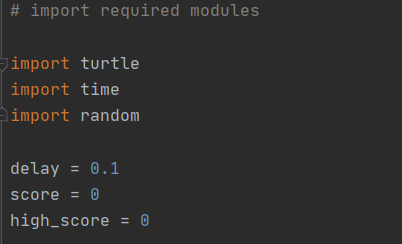
2.We can promote this game more and more using different OS platforms.

3.We can add some study tricks and quiz so kids can learn with game too.

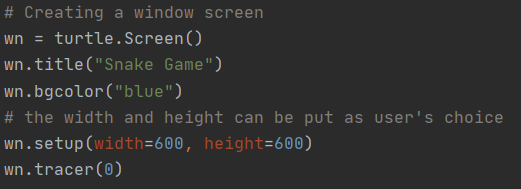
4.We can add some study tricks and quiz so kids can learn with game too.

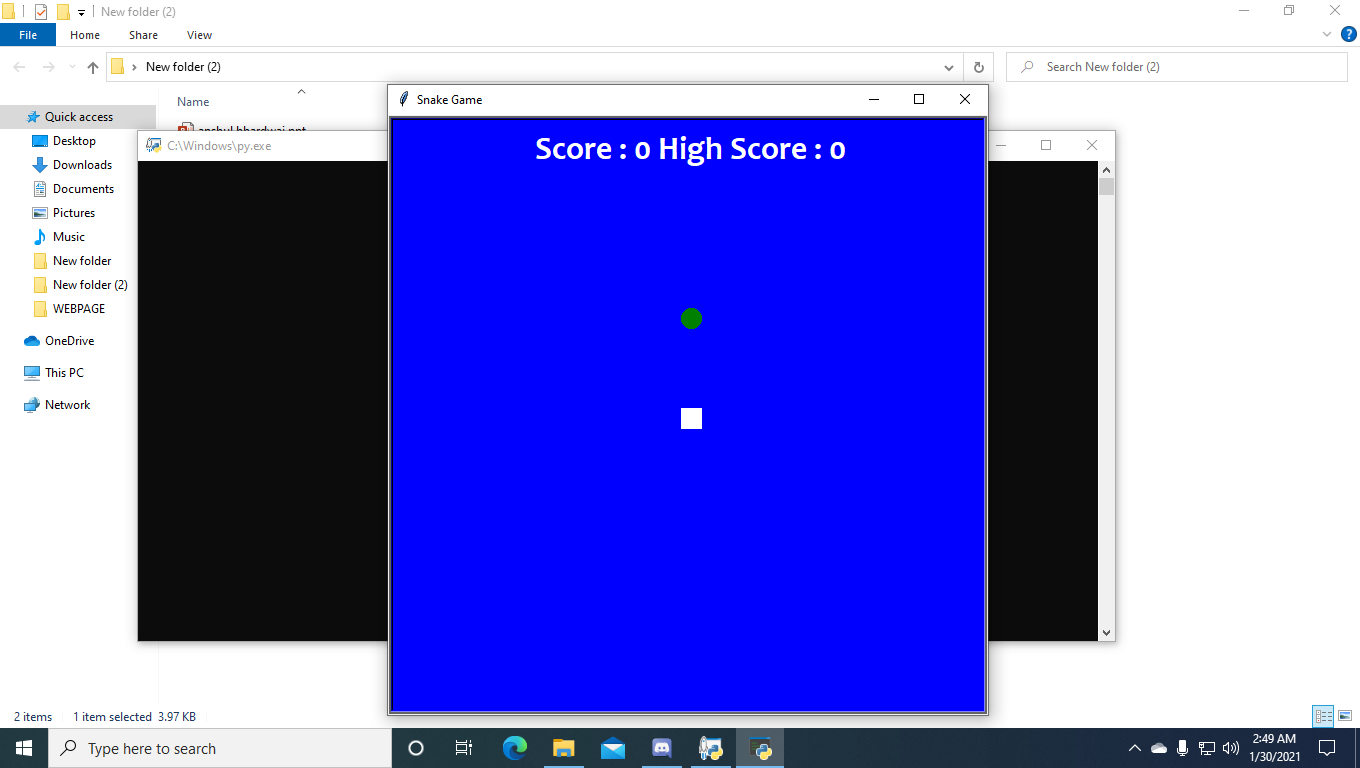
**OLD GAME**

**Step 1: importing modules**

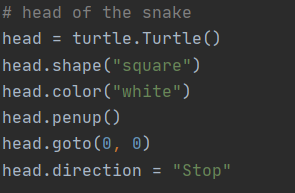


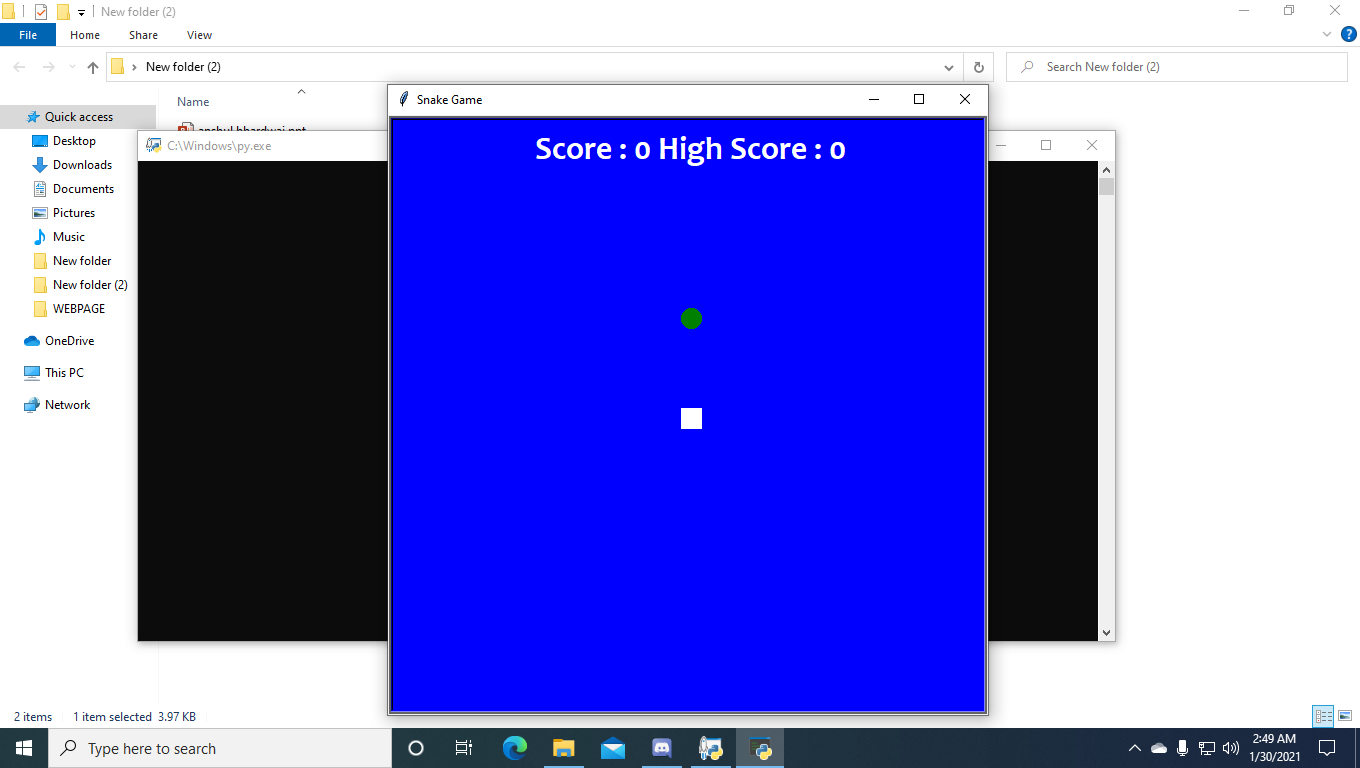
**Step 2: Now , Create the display of the game**



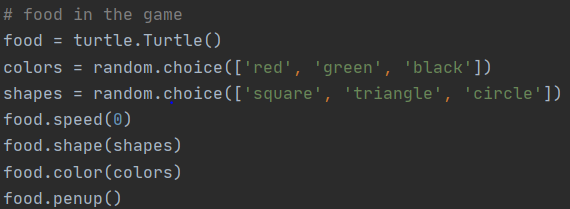


**Step 3: Now , Creating Head of snake**

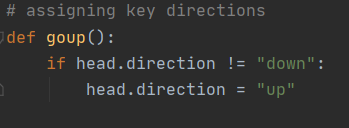


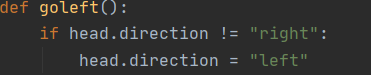


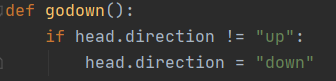
**Step 4: Now , Creating Food Particle**

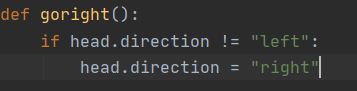


**Step 5: Validating the key for the snake’s movements.**









**Step 6: Now, lastly, we will create the gameplay where the following will be happening**

* The snake will grow its body when the snake eats the fruits.
* Giving color to the snake’s tail.
* After the fruit is eaten, the score will be counted.
* Checking for the snake’s head collisions with the body or the wall of the window screen.
* Restarting the game automatically from the start after the collision.
* The new shape and color of the fruit will be introduced every time the window is restarted.

The score will be returned to zero and a high score will be retained until the window is not closed

**First Phase Output**

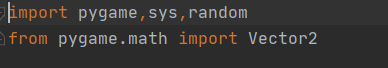


**NEW UPDATE!!**

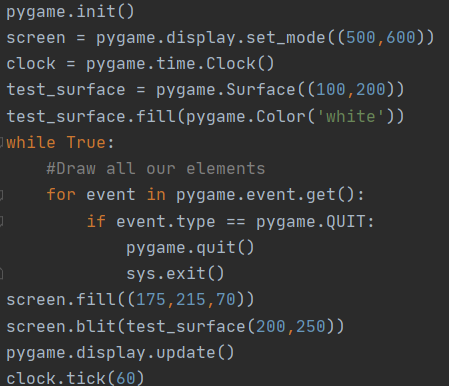
* Earlier we have designed the snake game by using basic turtle and random module. Now in this version we have introduced new modules and new user interface, which had completely changed the face of our game and to make it more user interactive we have added sound of churping the food🌝
* The new modules are-:
* Pygame
* Sys (System-specific parameters and functions)
* Vector

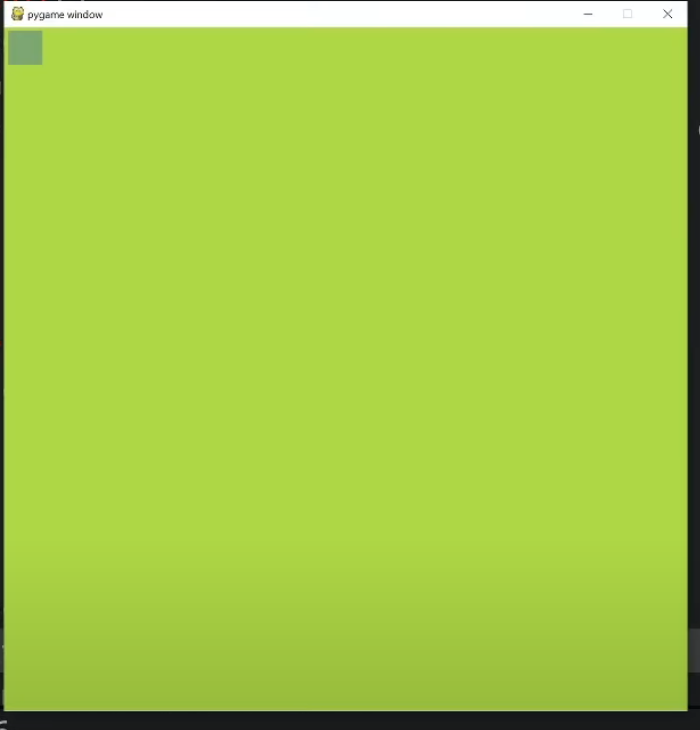
**Importing New Modules**

* **Random:** This function is used to generate random numbers in Python by using *random* module
* Pygame: This is a cross-platform set of Python modules which is used to create video games.
* Sys: System-specific parameters and functions. This module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter. It is always available.

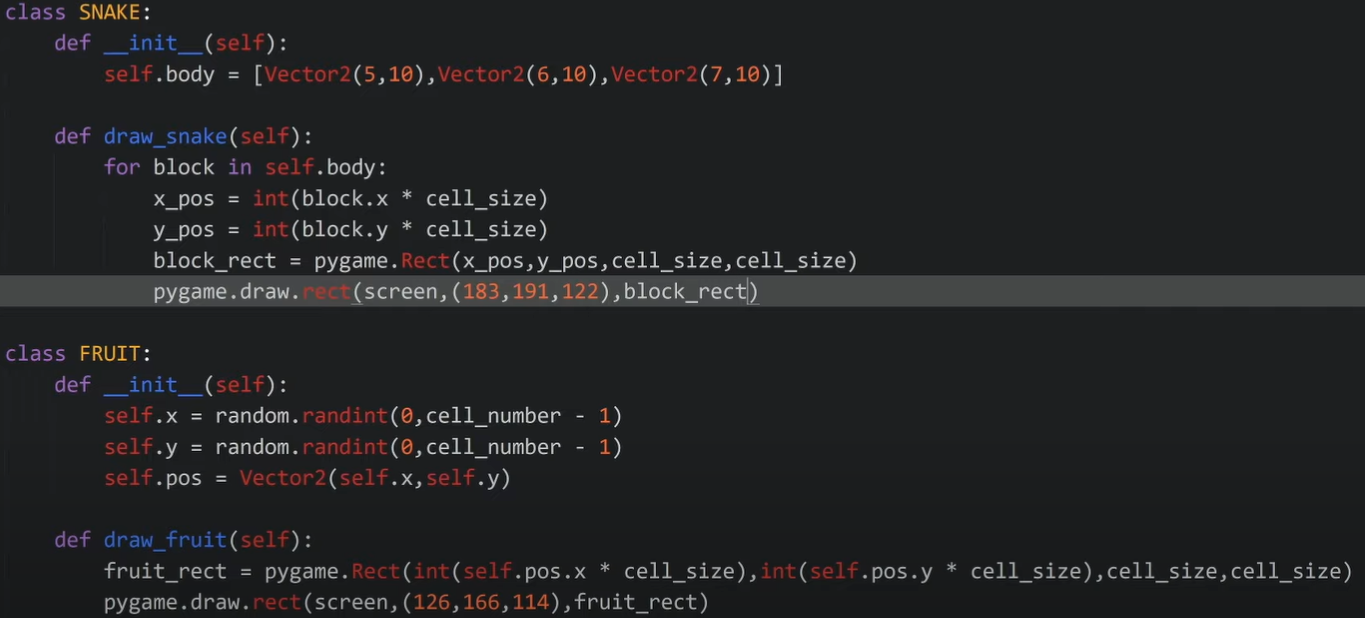


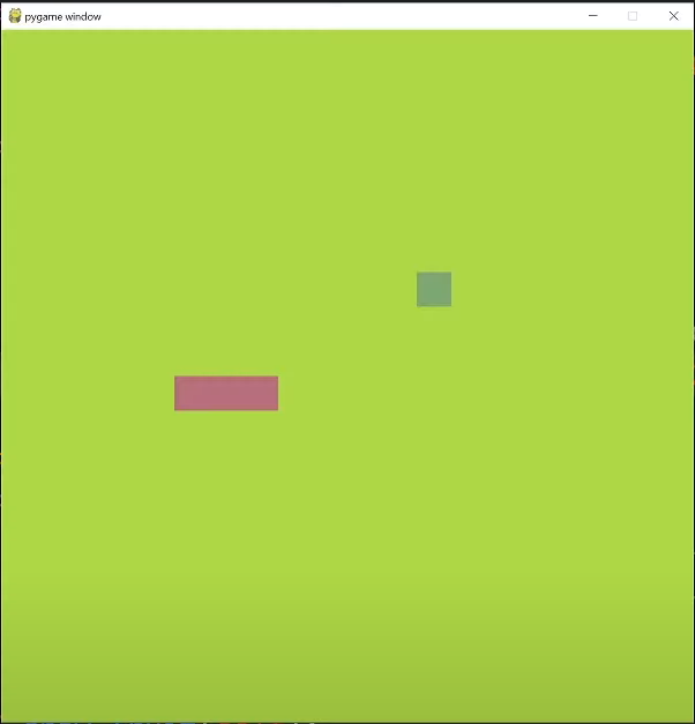
**Creating main screen**





**Creating the snake and fruit**





**Movement of Snake**

* Now, moving the snake

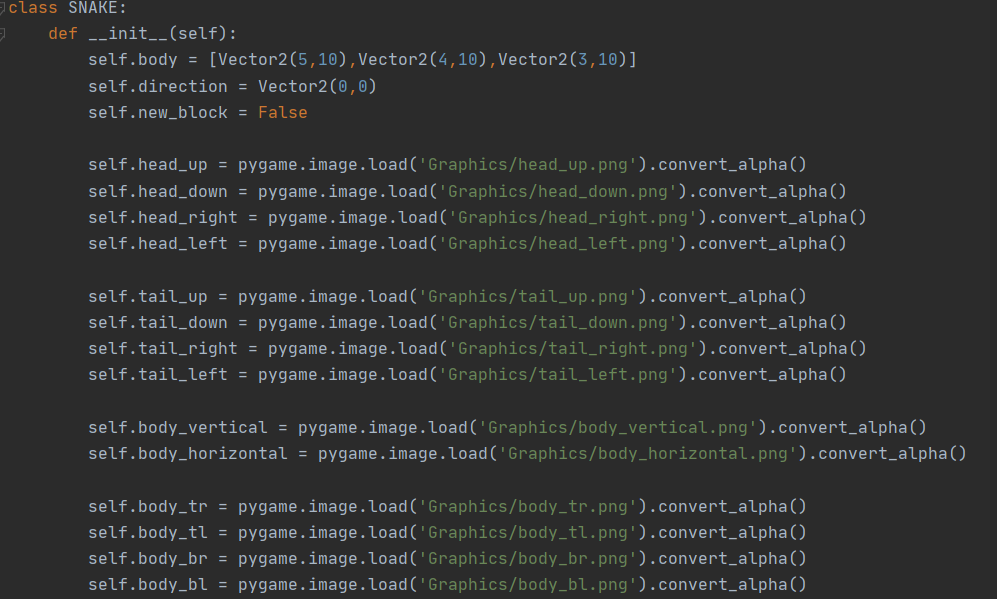
The head is moved to a new block and the block before the head gets the position where the head used to be and each block is moved to the position of the before that used to be before it (this deletes the last block)

Like:  
[ pos1, pos2, pos3 ] [ direction ]

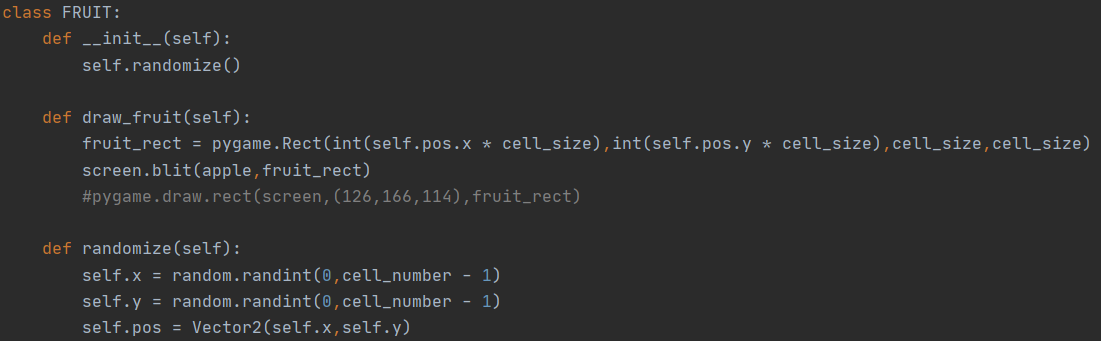


**Adding graphics in fruit and snake**

**For snake**

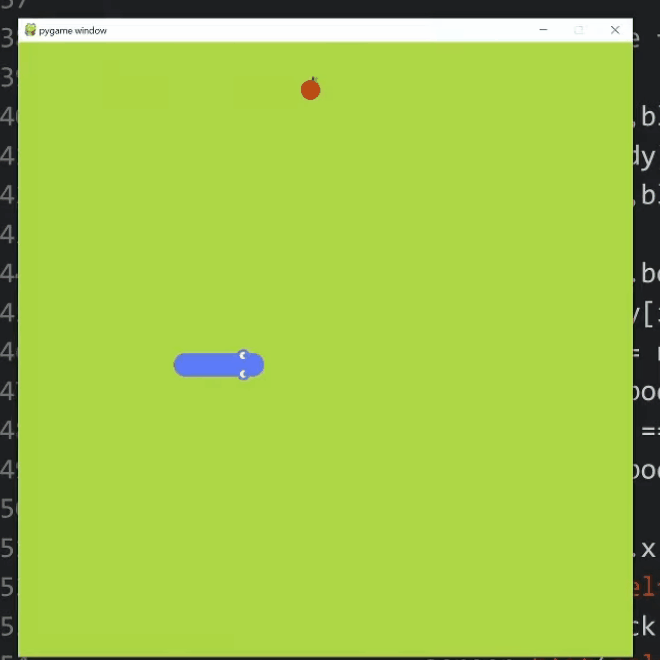


**For Fruit**

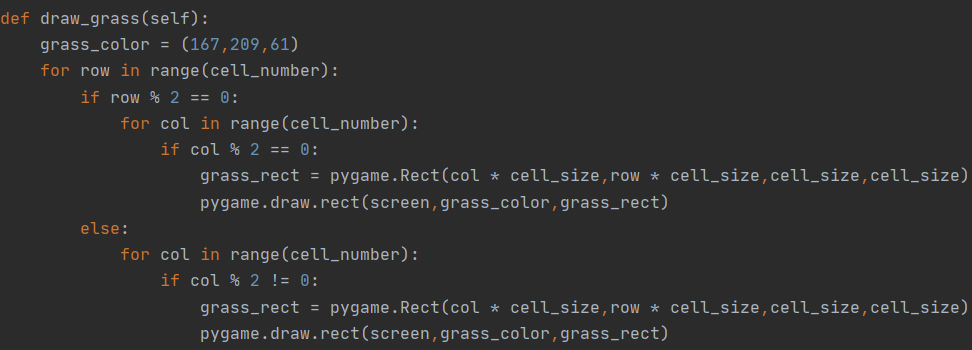


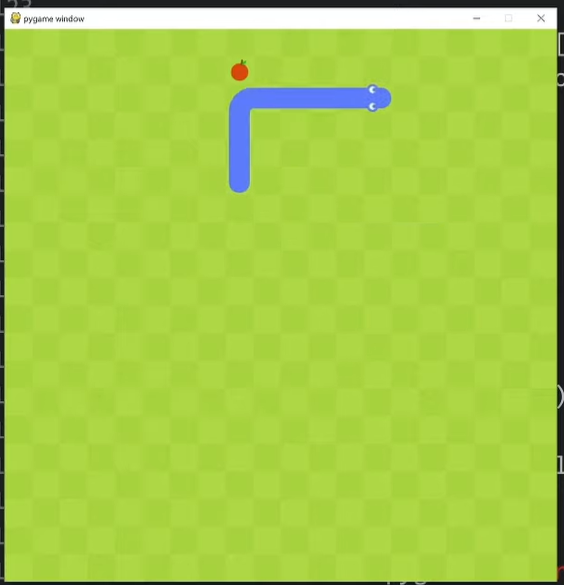


**Output**

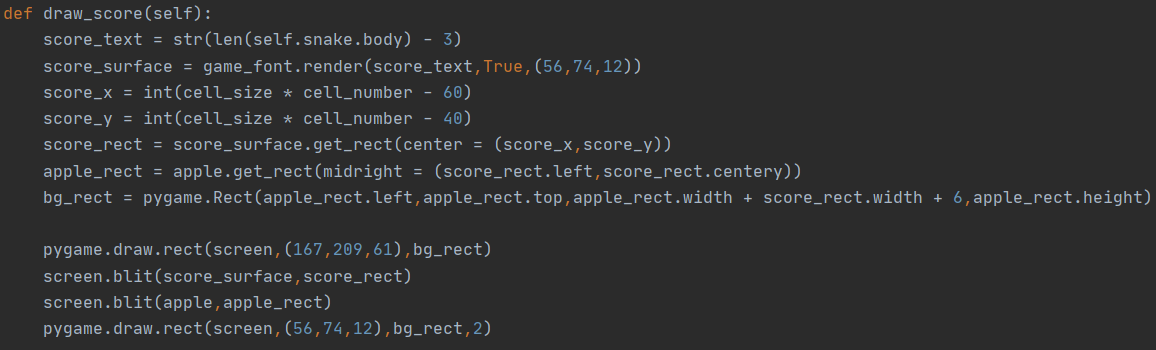


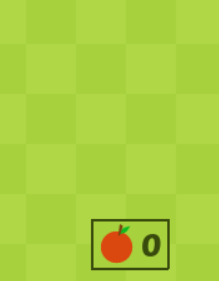
**Now, Adding Grass in it**





**Now, Adding the scoreboard**



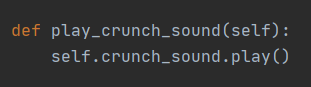


**Now, Adding the sound**

* Location of Sound File



* **Function of Sound**



**SOURCE CODE:**

import pygame,sys,random

from pygame.math import Vector2

class SNAKE:

    def \_\_init\_\_(self):

        self.body = [Vector2(5,10),Vector2(4,10),Vector2(3,10)]

        self.direction = Vector2(0,0)

        self.new\_block = False

        self.head\_up = pygame.image.load('Graphics/head\_up.png').convert\_alpha()

        self.head\_down = pygame.image.load('Graphics/head\_down.png').convert\_alpha()

        self.head\_right = pygame.image.load('Graphics/head\_right.png').convert\_alpha()

        self.head\_left = pygame.image.load('Graphics/head\_left.png').convert\_alpha()

        self.tail\_up = pygame.image.load('Graphics/tail\_up.png').convert\_alpha()

        self.tail\_down = pygame.image.load('Graphics/tail\_down.png').convert\_alpha()

        self.tail\_right = pygame.image.load('Graphics/tail\_right.png').convert\_alpha()

        self.tail\_left = pygame.image.load('Graphics/tail\_left.png').convert\_alpha()

        self.body\_vertical = pygame.image.load('Graphics/body\_vertical.png').convert\_alpha()

        self.body\_horizontal = pygame.image.load('Graphics/body\_horizontal.png').convert\_alpha()

        self.body\_tr = pygame.image.load('Graphics/body\_tr.png').convert\_alpha()

        self.body\_tl = pygame.image.load('Graphics/body\_tl.png').convert\_alpha()

        self.body\_br = pygame.image.load('Graphics/body\_br.png').convert\_alpha()

        self.body\_bl = pygame.image.load('Graphics/body\_bl.png').convert\_alpha()

        self.crunch\_sound = pygame.mixer.Sound('Sound/crunch.wav')

    def draw\_snake(self):

        self.update\_head\_graphics()

        self.update\_tail\_graphics()

        for index,block in enumerate(self.body):

            x\_pos = int(block.x \* cell\_size)

            y\_pos = int(block.y \* cell\_size)

            block\_rect = pygame.Rect(x\_pos,y\_pos,cell\_size,cell\_size)

            if index == 0:

                screen.blit(self.head,block\_rect)

            elif index == len(self.body) - 1:

                screen.blit(self.tail,block\_rect)

            else:

                previous\_block = self.body[index + 1] - block

                next\_block = self.body[index - 1] - block

                if previous\_block.x == next\_block.x:

                    screen.blit(self.body\_vertical,block\_rect)

                elif previous\_block.y == next\_block.y:

                    screen.blit(self.body\_horizontal,block\_rect)

                else:

                    if previous\_block.x == -1 and next\_block.y == -1 or previous\_block.y == -1 and next\_block.x == -1:

                        screen.blit(self.body\_tl,block\_rect)

                    elif previous\_block.x == -1 and next\_block.y == 1 or previous\_block.y == 1 and next\_block.x == -1:

                        screen.blit(self.body\_bl,block\_rect)

                    elif previous\_block.x == 1 and next\_block.y == -1 or previous\_block.y == -1 and next\_block.x == 1:

                        screen.blit(self.body\_tr,block\_rect)

                    elif previous\_block.x == 1 and next\_block.y == 1 or previous\_block.y == 1 and next\_block.x == 1:

                        screen.blit(self.body\_br,block\_rect)

    def update\_head\_graphics(self):

        head\_relation = self.body[1] - self.body[0]

        if head\_relation == Vector2(1,0): self.head = self.head\_left

        elif head\_relation == Vector2(-1,0): self.head = self.head\_right

        elif head\_relation == Vector2(0,1): self.head = self.head\_up

        elif head\_relation == Vector2(0,-1): self.head = self.head\_down

    def update\_tail\_graphics(self):

        tail\_relation = self.body[-2] - self.body[-1]

        if tail\_relation == Vector2(1,0): self.tail = self.tail\_left

        elif tail\_relation == Vector2(-1,0): self.tail = self.tail\_right

        elif tail\_relation == Vector2(0,1): self.tail = self.tail\_up

        elif tail\_relation == Vector2(0,-1): self.tail = self.tail\_down

    def move\_snake(self):

        if self.new\_block == True:

            body\_copy = self.body[:]

            body\_copy.insert(0,body\_copy[0] + self.direction)

            self.body = body\_copy[:]

            self.new\_block = False

        else:

            body\_copy = self.body[:-1]

            body\_copy.insert(0,body\_copy[0] + self.direction)

            self.body = body\_copy[:]

    def add\_block(self):

        self.new\_block = True

    def play\_crunch\_sound(self):

        self.crunch\_sound.play()

    def reset(self):

        self.body = [Vector2(5,10),Vector2(4,10),Vector2(3,10)]

        self.direction = Vector2(0,0)

class FRUIT:

    def \_\_init\_\_(self):

        self.randomize()

    def draw\_fruit(self):

        fruit\_rect = pygame.Rect(int(self.pos.x \* cell\_size),int(self.pos.y \* cell\_size),cell\_size,cell\_size)

        screen.blit(apple,fruit\_rect)

        #pygame.draw.rect(screen,(126,166,114),fruit\_rect)

    def randomize(self):

        self.x = random.randint(0,cell\_number - 1)

        self.y = random.randint(0,cell\_number - 1)

        self.pos = Vector2(self.x,self.y)

class MAIN:

    def \_\_init\_\_(self):

        self.snake = SNAKE()

        self.fruit = FRUIT()

    def update(self):

        self.snake.move\_snake()

        self.check\_collision()

        self.check\_fail()

    def draw\_elements(self):

        self.draw\_grass()

        self.fruit.draw\_fruit()

        self.snake.draw\_snake()

        self.draw\_score()

    def check\_collision(self):

        if self.fruit.pos == self.snake.body[0]:

            self.fruit.randomize()

            self.snake.add\_block()

            self.snake.play\_crunch\_sound()

        for block in self.snake.body[1:]:

            if block == self.fruit.pos:

                self.fruit.randomize()

    def check\_fail(self):

        if not 0 <= self.snake.body[0].x < cell\_number or not 0 <= self.snake.body[0].y < cell\_number:

            self.game\_over()

        for block in self.snake.body[1:]:

            if block == self.snake.body[0]:

                self.game\_over()

    def game\_over(self):

        self.snake.reset()

    def draw\_grass(self):

        grass\_color = (167,209,61)

        for row in range(cell\_number):

            if row % 2 == 0:

                for col in range(cell\_number):

                    if col % 2 == 0:

                        grass\_rect = pygame.Rect(col \* cell\_size,row \* cell\_size,cell\_size,cell\_size)

                        pygame.draw.rect(screen,grass\_color,grass\_rect)

            else:

                for col in range(cell\_number):

                    if col % 2 != 0:

                        grass\_rect = pygame.Rect(col \* cell\_size,row \* cell\_size,cell\_size,cell\_size)

                        pygame.draw.rect(screen,grass\_color,grass\_rect)

    def draw\_score(self):

        score\_text = str(len(self.snake.body) - 3)

        score\_surface = game\_font.render(score\_text,True,(56,74,12))

        score\_x = int(cell\_size \* cell\_number - 60)

        score\_y = int(cell\_size \* cell\_number - 40)

        score\_rect = score\_surface.get\_rect(center = (score\_x,score\_y))

        apple\_rect = apple.get\_rect(midright = (score\_rect.left,score\_rect.centery))

        bg\_rect = pygame.Rect(apple\_rect.left,apple\_rect.top,apple\_rect.width + score\_rect.width + 6,apple\_rect.height)

        pygame.draw.rect(screen,(167,209,61),bg\_rect)

        screen.blit(score\_surface,score\_rect)

        screen.blit(apple,apple\_rect)

        pygame.draw.rect(screen,(56,74,12),bg\_rect,2)

pygame.mixer.pre\_init(44100,-16,2,512)

pygame.init()

cell\_size = 40

cell\_number = 20

screen = pygame.display.set\_mode((cell\_number \* cell\_size,cell\_number \* cell\_size))

clock = pygame.time.Clock()

apple = pygame.image.load('Graphics/apple.png').convert\_alpha()

game\_font = pygame.font.Font('Font/PoetsenOne-Regular.ttf', 25)

SCREEN\_UPDATE = pygame.USEREVENT

pygame.time.set\_timer(SCREEN\_UPDATE,150)

main\_game = MAIN()

while True:

    for event in pygame.event.get():

        if event.type == pygame.QUIT:

            pygame.quit()

            sys.exit()

        if event.type == SCREEN\_UPDATE:

            main\_game.update()

        if event.type == pygame.KEYDOWN:

            if event.key == pygame.K\_UP:

                if main\_game.snake.direction.y != 1:

                    main\_game.snake.direction = Vector2(0,-1)

            if event.key == pygame.K\_RIGHT:

                if main\_game.snake.direction.x != -1:

                    main\_game.snake.direction = Vector2(1,0)

            if event.key == pygame.K\_DOWN:

                if main\_game.snake.direction.y != -1:

                    main\_game.snake.direction = Vector2(0,1)

            if event.key == pygame.K\_LEFT:

                if main\_game.snake.direction.x != 1:

                    main\_game.snake.direction = Vector2(-1,0)

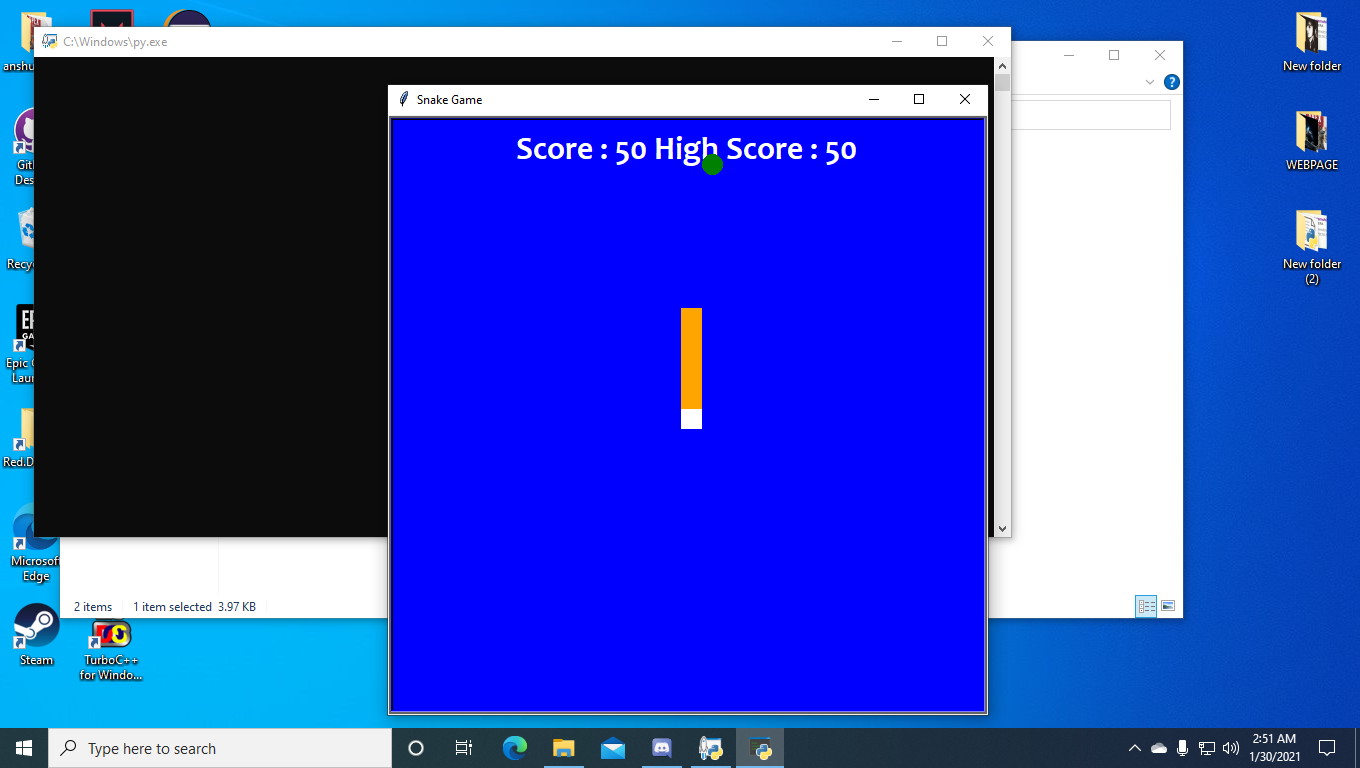
    screen.fill((175,215,70))

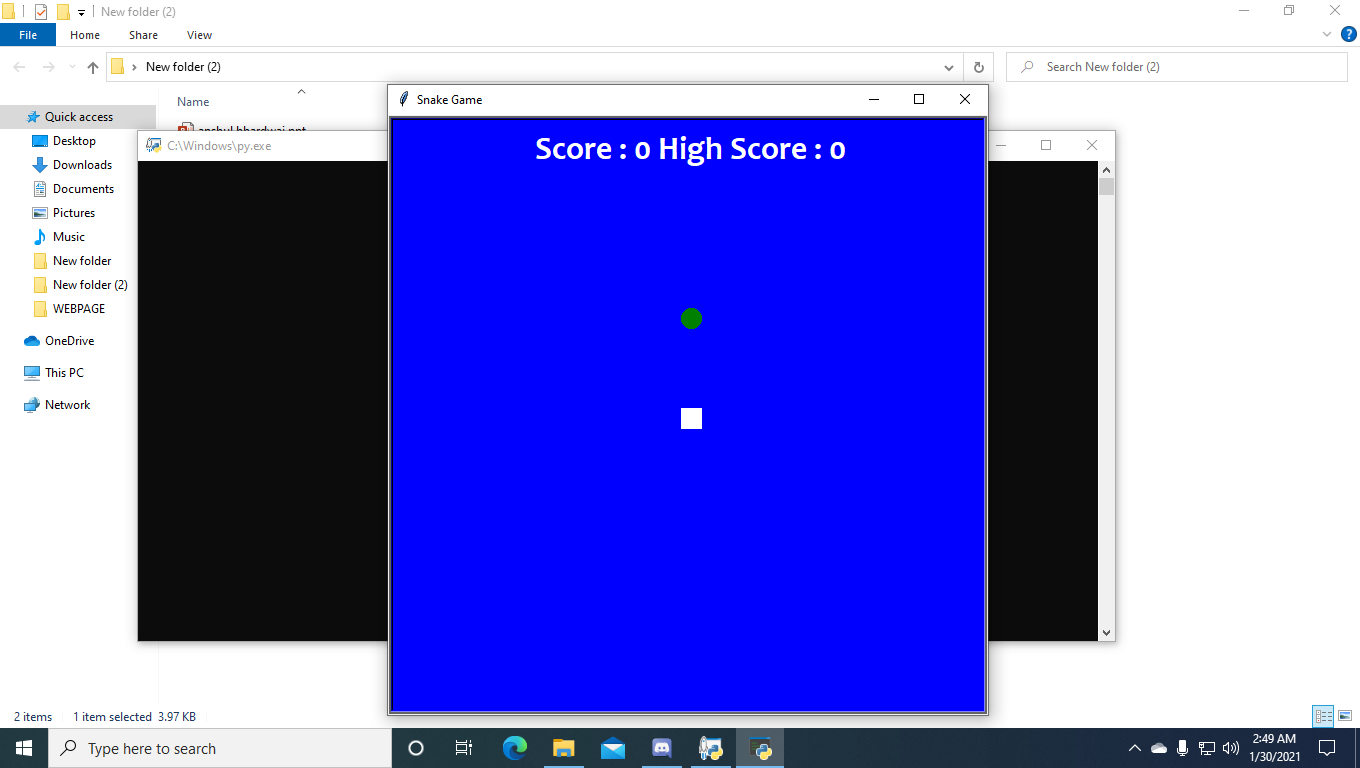
    main\_game.draw\_elements()

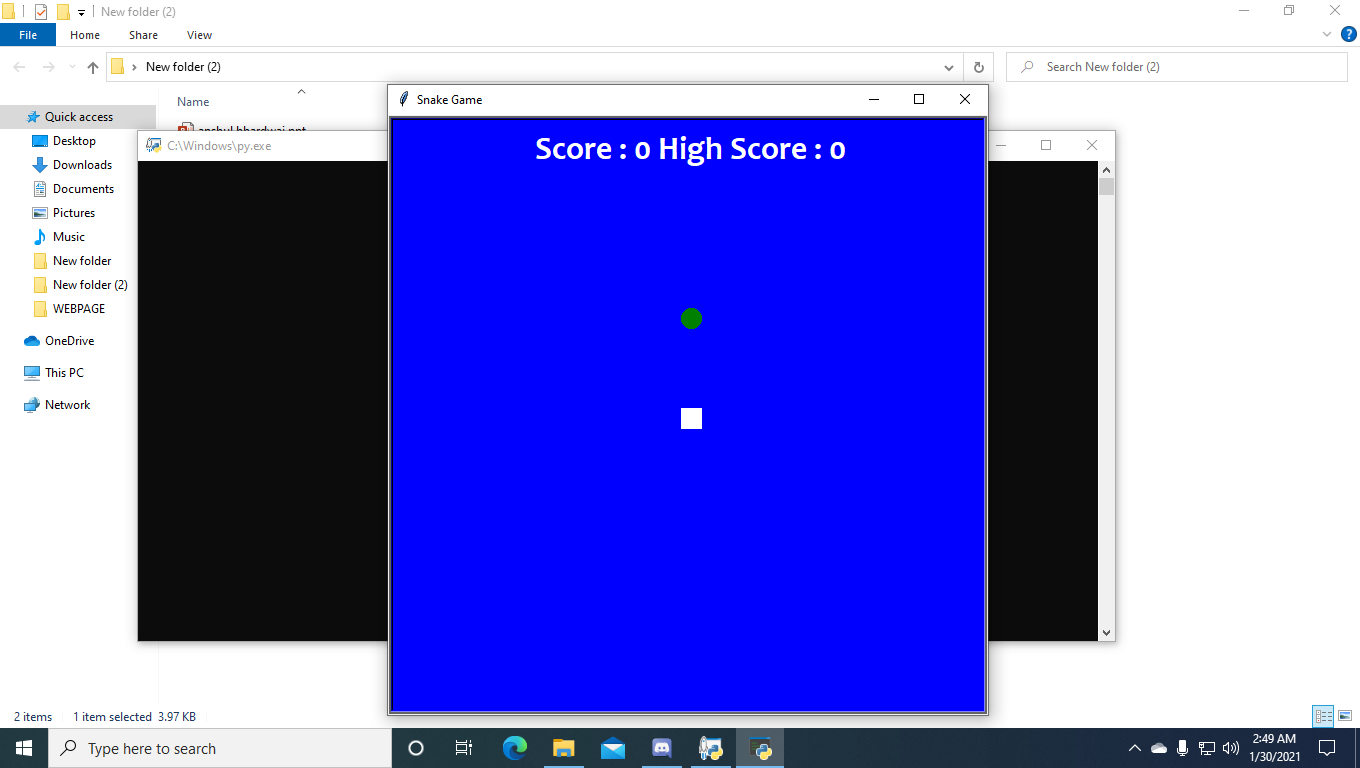
    pygame.display.update()

    clock.tick(60)

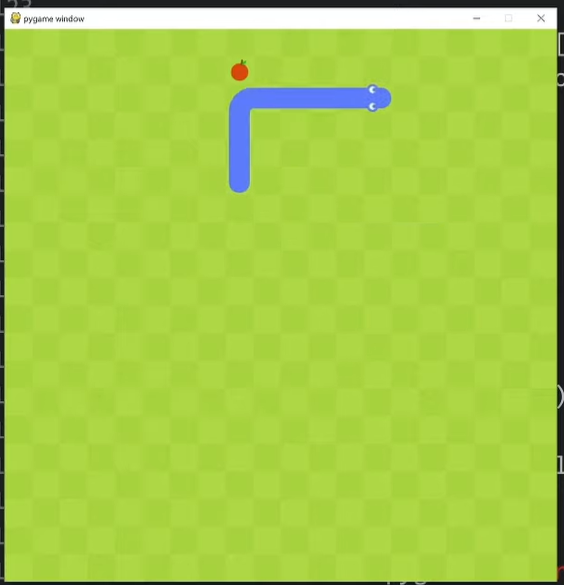
**OLD GAME:-**

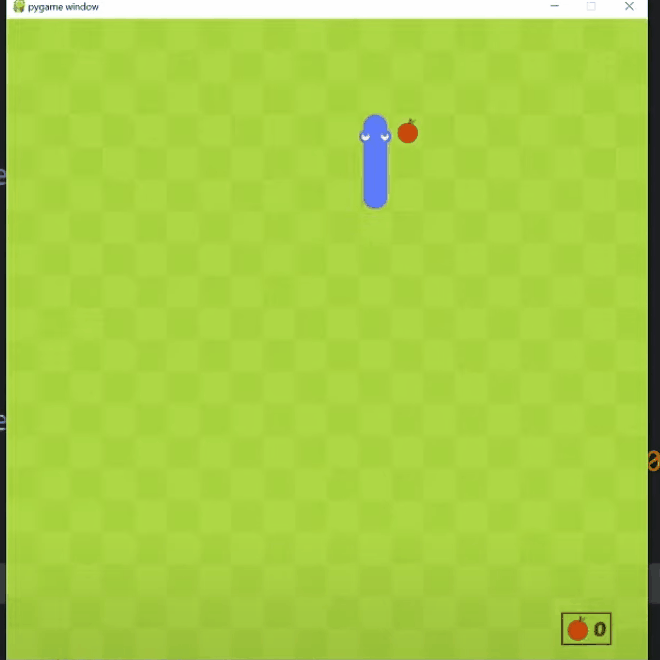
****

****



**GAME UPDATED:-**





**REFRENCES** :

* https://en.wikipedia.org/wiki
* https://www.youtube.com
* https://www.cs.bham.ac.uk