Project proposal:

Snake Game using python

Submitted to:

Lecturer Faria

Group Members	Nutech ID
Syed Hamid Haider	F20605034
Usman Malik	F20605053
M. Irbaz Anwar	F20600537
Ahmed Ali Haider	F20605013

Table of Content

Chap # 1: Overview

1.1 Introduction

In this project we aim to create a snake game in python. This specific project was chosen to be done in this programming language due to its versatility and it is within the best interest of our future. We aim to own our skills so that it may help us to make a successful career out of this experience.

1.2 Abstract

This project's purpose is to make a fun and simple snake game with some features. The simplicity of this game makes it the best possible choice for a group of unexperienced programmer into the game development field. This project will allow is to focus on advanced topics like machine learning algorithms, automation and artificial intelligence which are all related to the language this game was programmed in.

1.3 Scope

This project allows us to have an insight on Indie game development. As in indie game development the task of creating a fully-fledged game taken up by only a few handful of people. Game development is a field that requires a person to have mastery of a programming language to constantly be innovative with ideas so that they can compete with other game studios.

1.4 Process Involved

The process of making this project requires extensive knowledge of the programming language python. This includes its libraries, modules and management of the memory. It also involves making the project as simple as possible and easily understandable in the case that modification from a third party group are required

1.5 Technology required

The project requires the following tech.

1.5.1 Hardware

For the making of this project the basic requirement include an average personal computer. Minimum requirements are as under

Processor: Intel core 2 Duo

RAM: 1G DDR1

Storage: 10GB HDD of free space

OS: Windows Xp

Drivers: Direct X11

1.5.2 Software

The software requirements include an IDE (Integrated Development Environment) that allows us to program in the required language and specific modules and drivers that allows us to generate the executable file for the game.

1.6 Benefits

1. Introduction to OOP based languages
2. Learning a new language
3. Understanding sprite formation
4. The concept of key configuration and assignment commands
5. Importation of modules and python packages
6. Understanding the IDE.
7. Learning the limitations of the programming language.
8. Community research
9. UI/UX interface introduction

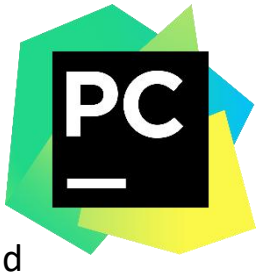
1.7 Reasons for choosing this project

This specific project was chosen to be done in this programming language due to its versatility and it is within the best interest of our future. The simplicity of this project makes it a challenge itself as the game to be made user friendly but at the same time challenging and fun so that it can draw attention. On the coding end it has to be made with simplistic, so that in the case if modification are required they can be done even by a third party without the presence of the original source code producer.

Chap # 2: Making of the Game

2.1 Introduction to the process

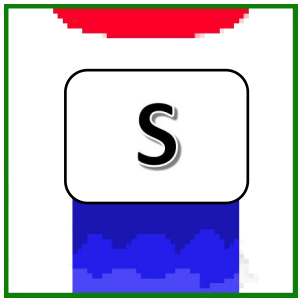
In this project the idea was to use the best compatible IDE with the language that was to be chosen. In our case the language was python and the IDE was Pycharm 2020 Edition.



This helped us immensely as the community has been built to a very solid point due to the contributions from its user that want this concept to succeed. The next process was to pick which of the hundred possible packages and modules that could have been used in making the game. In our case we used the ***turtle*** , ***time*** and ***random*** modules. These allowed to making the structure and mechanisms for the game.

2.2 General Idea for the game

As per stated above in the report, the main idea was to keep the game as simple as possible. This meant the game was based on the idea to be as user friendly as possible. The snake, fruit and the core game mechanism.



2.3 Comparison with other projects

The project list included the following pacman, ticktacktoe, OCR (optical character recognition), news feed app, shooting game, quiz game, notes app.

2.3.1 Similarities

The most common similarity for was that it was entirely software based. This meant all the project were coding intensive and perfectly suitable for our field. Other similarities included the use of other OOP languages i.e. C++, Java, the making of an application and the making of user interface so that it can be interacted with.

2.3.2 Differences

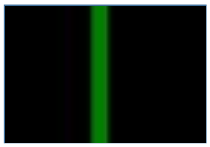
The differences is where the concept of the snake game stood out from the rest. The most noticeable differences included sprite formation, rendering the movement of the main sprite, mathematical calculations, platform formation, UI/UX interface and sprite manipulation.

2.4 Structure of the game

The structure for the game itself involved creating simple yet acceptable substitutes for the sprites so that the user who is interacting with the game can understand what they are controlling.

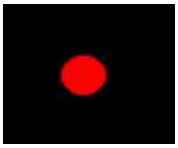
2.4.1 Game Border

The game border is the designated area within which the game operates. It has only two functions to act as a barrier for the game and to be a kill zone upon interaction. This means that it would be an immediate game over if the snake touches the border.



2.4.2 Fruit

The fruit is the main objective for the player. It is counted as the score. The more the player collects, the higher the score he can achieve. The colour was picked to be red so that it was easy for the human eye to perceive.

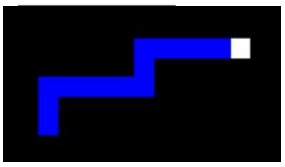


2.4.3 Snake Head

The snake head is the part where most of the interaction takes place. It must interact with the fruit in order for the game to progress. It is also the area where movement is coding.

2.4.4 Snake Body

After eating the fruit it is important for the game to progress and the best way to show progress is to increase the size of the snake itself and to make the game harder.



2.5 Future Possibilities

The future of this project looks bright as there is much things to improve i.e. using better libraries to making better sprites, making the game run at 60fps, making a proper launcher for the game, adding obstacles, designing many different and beautiful sprites for the snake and fruit and making the game more refined in order for a public release.

Chap # 3: Setup of the Game

3.1 Required Modules

The modules required are as under

Turtle: Module gives us a feature to draw on a drawing board.

Random: module will be used to generate random numbers

Time: module is an inbuilt module in python. It provides the functionality of time.

3.2 Creating Game Area

The game area is the main area where the game takes place. It is

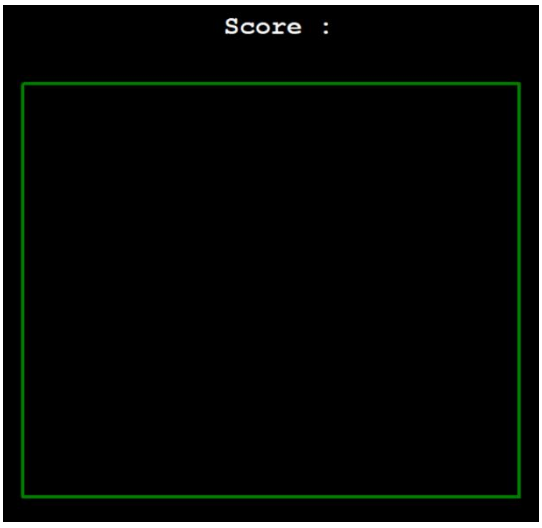
```
def game_function():
        #Creating Screen
        screen = turtle.Screen()
        screen.title("Snake Game")
        screen.setup(width = 700, height = 700)
        screen.tracer(0)
        turtle.bgcolor("black")


        #Creating a Border
        turtle.speed(5)
        turtle.pensize(4)
        turtle.penup()
        turtle.goto(-310,250)
        turtle.pendown()
        turtle.color("green")
        turtle.forward(600)
        turtle.right(90)
        turtle.forward(500)
        turtle.right(90)
        turtle.forward(600)
        turtle.right(90)
        turtle.forward(500)
        turtle.penup()
        turtle.hideturtle()
```

Defining parameters

Setting colour

Output



also generated using the turtle modules.

3.3 Creating Fruit

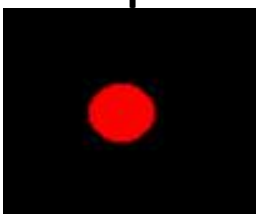
The fruit is the main objective for the player. It is counted as the score.

The more the player collects, the higher the score he can achieve. The coding allows the fruit to be generated anywhere within the game area and be collected by the snake head.

```
#Food
fruit = turtle.Turtle()
fruit.speed(0)
fruit.shape("circle")
fruit.color("red")
fruit.penup()
fruit.goto(30,30)

old_fruit=[]
```

Output

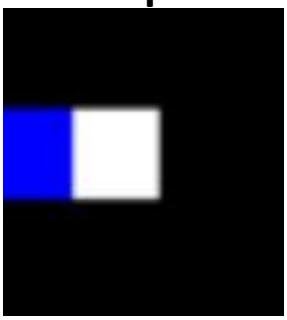


3.4 Creating Snake Head

The snake head is the part where most of the interaction takes place. It must interact with the fruit in order for the game to progress. The colour is in order to differentiate it from the body.

```
#Snake Head
snake = turtle.Turtle()
snake.speed(0)
snake.shape("square")
snake.color("white")
snake.penup()
snake.goto(0,0)
snake.direction = "stop"
```

Output

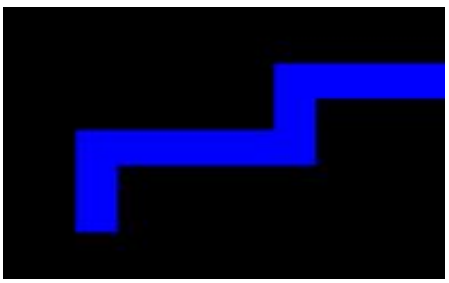


3.5 Creating Snake Body

After eating the fruit it is important for the game to progress and the best way to show progress is to increase the size of the snake body.

```
#Creating New Snake Body Part
new_fruit = turtle.Turtle()
new_fruit.speed(0)
new_fruit.shape("square")
new_fruit.color("blue")
new_fruit.penup()
old_fruit.append(new_fruit)
```

Output



Chap # 4: Mechanisms of the Game

4.1 Key configuration

The key configuration of the game is mainly focused around the arrow key button. They are the main interactive part of the game. With the press of these button the user is able to control the movement and direction of the snake allowing them to collect the fruit.

```
#Keyboard Bindings
screen.listen()
screen.onkeypress(snake_move_up, "Up")
screen.onkeypress(snake_move_down, "Down")
screen.onkeypress(snake_move_left, "Left")
screen.onkeypress(snake_move_right, "Right")
```

This is done with the help of the ***turtle*** package that allows us to assign the values to these keys.

Continuing to the part of the functionality of the keys. Their purpose to turn the snake according to their corresponding assigned value.

```
#Snake Movement
def snake_move_up():
        if snake.direction != "down":
                snake.direction = "up"
```

Code for moving upwards

```
def snake_move_down():
        if snake.direction != "up":
                snake.direction = "down"
```

Code for moving downwards

```
def snake_move_right():
        if snake.direction != "left":
                snake.direction = "right"
```

Code for moving right

```
def snake_move_left():
        if snake.direction != "right":
                snake.direction = "left"
```

Code for moving left

The important part to notice is the "***!=***" that is with the definition of the command. This statement allows us to fix a crucial problem with the game. If the snake is moving in one direction this command will not allow to turn the head 180 degrees and collide with the body behind it.

4.2 Snake and Border Collision

One of the most important mechanics of the game as it determine if the player has lost the game of not. It is important to set boundaries to the game area in the case that the player does not go beyond the area that is being displayed.

```
#Snake and Border Collision
if snake.xcor() > 280 or snake.xcor() < -300 or snake.ycor() > 240 or snake.ycor() < -240:
        time.sleep(1)
        screen.clear()
        screen.bgcolor('black')
        scoring.goto(0,0)
        scoring.write("Game Over\nYour Score is {}".format(score), align="center", font=("Courier", 30, "bold"))
        restart = screen.textinput("Retry", "Do you want to restart ? (y/n)")
        if restart == "y" or restart == "Y":
                screen.clearscreen()
                game_function()
        else:
                check = False
```

It is important to have this mechanism in order to avoid making an endless experience.

4.3 Snake and Snake Collision

Also one of the most important mechanics of the game as it determine if the player has lost the game of not. It is important to make the body of the snake and object as in if the head touches the

```
#Snake and Snake Collision
for food in old_fruit:
        if food.distance(snake) < 20:
                time.sleep(1)
                screen.clear()
                screen.bgcolor('black')
                scoring.goto(0,0)
                scoring.write("Game Over\nYour Score is {}".format(score), align="center", font=("Courier", 30, "bold"))
                restart = screen.textinput("Retry", "Do you want to restart ? (y/n)")
                if restart == "y" or restart == "Y":
                        screen.clearscreen()
                        game_function()
                else:
                        check = False
```

body it result in a game over.

4.4 Restart option

The restart option of given to the player in the case if he want to try again in order to obtain a better score or would like to experience

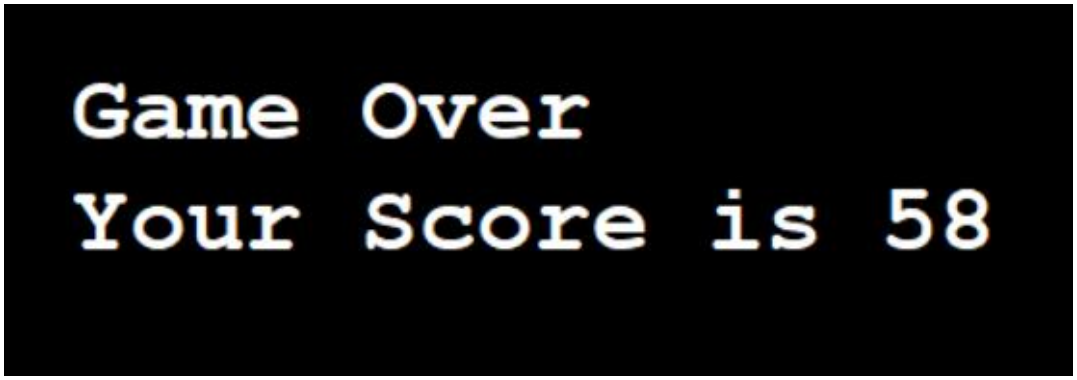
```
restart = screen.textinput("Retry", "Do you want to restart ? (y/n)")
if restart == "y" or restart == "Y":
        screen.clearscreen()
        game_function()
else:
        check = False
```

the game again.

This code allows us to show a prompt asking the player if he would like to try again. If the user enter the letter "***Y***" (capitalized or not) the score of the game is reset and begins anew.

4.5 Game over screen

The game over screen the prompted either when the player has touched the border or the snake head collides with the snake body. It is important for this to exist in order to indicate that the user has lost



and they cannot continue any further.

The added feature is that it also displays the score achieved of the game that the user just lost.