 **DOCUMENTATION**

Project Name: Snake Game

Development Language: Python3

Developed By:

* Vinay Kumar Singh

CSE

**Table of Content:**

1. Introduction
2. Acknowledgement(s)
3. Problem
4. Solution
5. Game Attributes
6. Approach
7. Tools Used
8. Code
9. Output

Reference

1. **Introduction**

Project documentation is concerned with describing the delivered software product, in this case the Single player Snake game project. Project documentation includes user documentation which tells users how to use the software product and system documentation which is principally intended for further development and understanding.

1. **Acknowledgement(s)**

Thank **yo**u **Harsha Mahati** Sir for pushing us and inspiring to build and understand a Single player Snake game. Thanks for the effort and giving us lectures and explaining the architecture. Thanks for helping when we were stop in the middle during the development phase. We could not have built such a challenging project without your help.

1. **Problem:**

The problem is to make a simple project using Python3 and its library. This game can’t be played online because it has no Network connectivity module applied, possibly the code can be shared and then played, and single player can play only at a time.

1. **Solution:**

We have develop a simple snake game using some python3 module like “turtle, time, random” and looping as well as conditional statement too.

First of all we draw a simple architecture of the game, what steps need to be followed, so on and so forth.

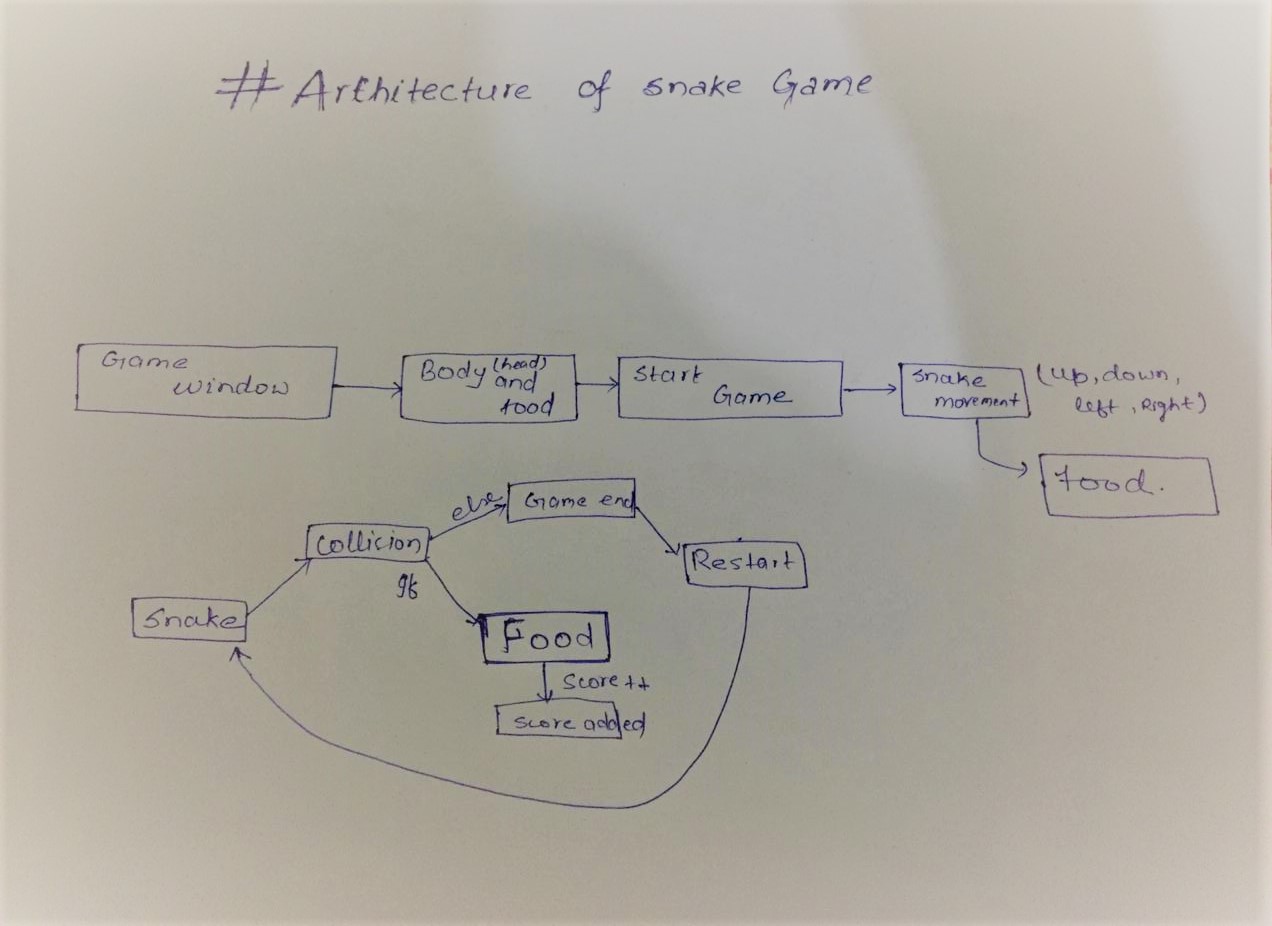


Fig: 4.1 (Architecture of the game)

The Game Architecture is the simplified graphical view of the game. It shows how the components work and the basic view of the game at action. The architectural view of the game is very important. Simply it gives an overview of the game functionality and it makes the game easy to understand.

1. **Game Attributes:**

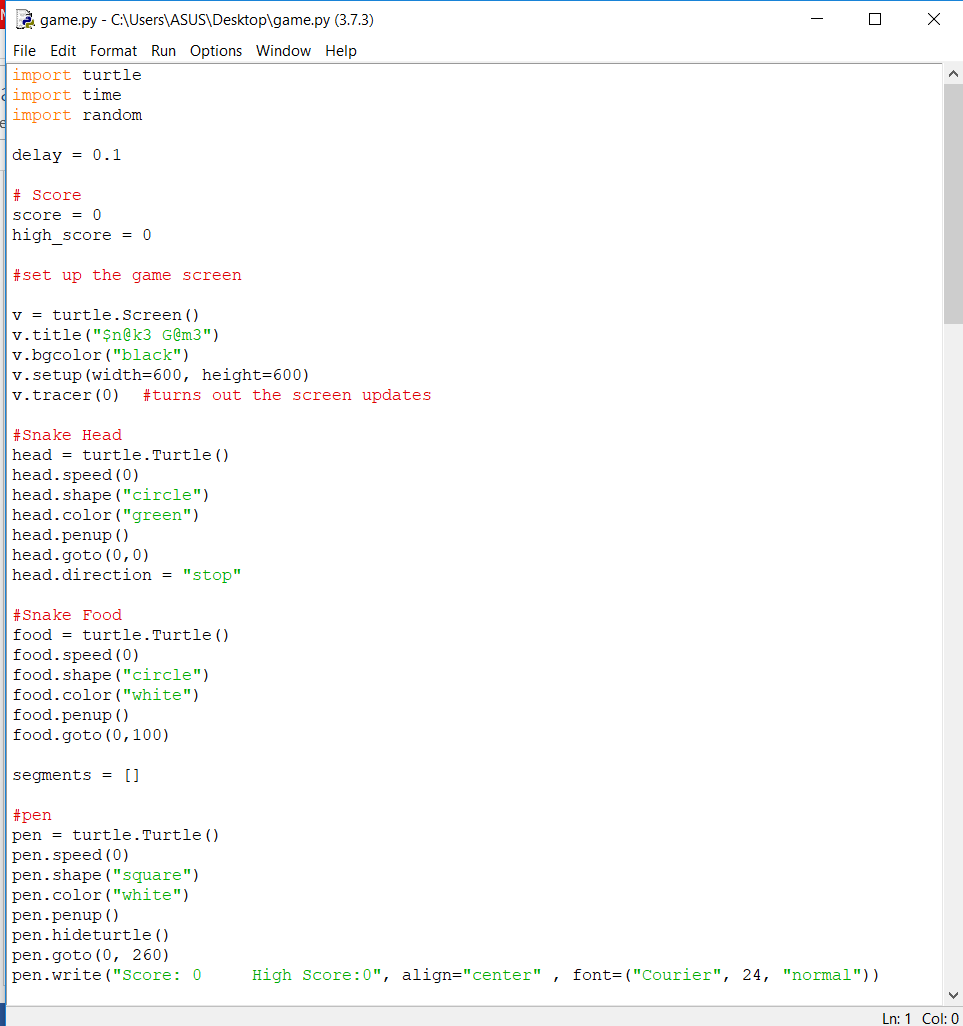
The name of the game is “Sn@k3 G@m3”. It’s a single player game which can be played in someone Computer or Laptop. In this game we use four arrow keys and “W”,”S”,”D” & “A” to control the snake direction. By typing the direction key the game start and

By default the game will be stop if the direction key are not pressed at the start.



Fig: 5.1 (Game Window)

The body code for the head and the food at the start of the game window(screen) is



1. **Approach:**

Here we develop the game and test the game and ultimately play the game. Construction phase can be divided into 2 phases:

* Building the Game
* Game play

**6.1.) Building the Game:**

When we first start the game we first plan, find out the requirements, draw the architecture.

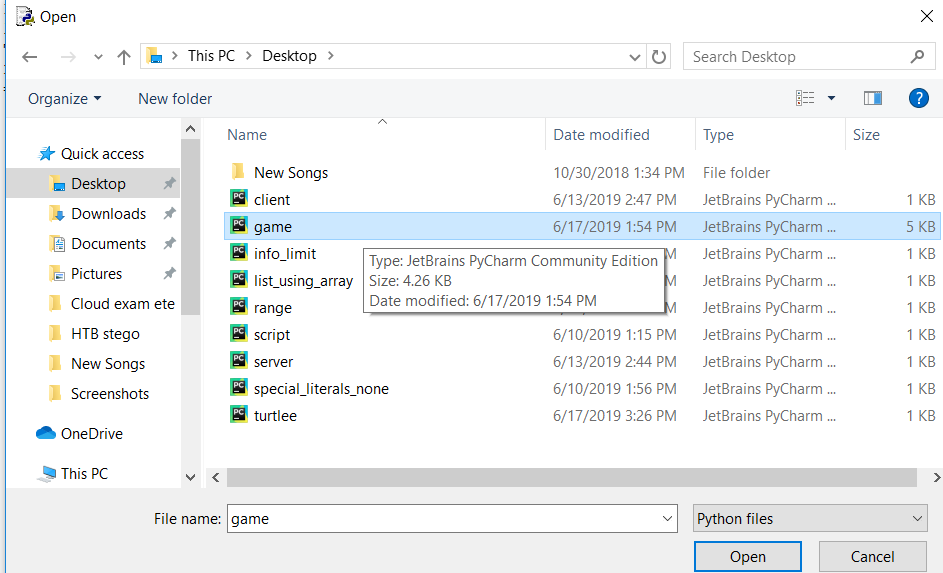
To build the game we need to know programming language (Python3 and its libraries).

The development phases are:

* Creating a single Snake.
* Creating a Food.
* Snake collision with food.
* Snake collision with each other body.
* Snake don’t move two directions at a same time.
* Snake collision with the boundaries.

**6.2.) Game Play:**

The steps of game play are given below:-

* Open the game application:
* Press the direction arrow and the game will start.

1. **Tools Used:**

The only Python3 module are used like “turtle (for the screen and the game functioning), time, random”.

1. **Code:**

import turtle

import time

import random

delay = 0.1

# Score

score = 0

high\_score = 0

#set up the game screen

v = turtle.Screen()

v.title("$n@k3 G@m3")

v.bgcolor("black")

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

v.tracer(0) #turns out the screen updates

#Snake Head

head = turtle.Turtle()

head.speed(0)

head.shape("circle")

head.color("green")

head.penup()

head.goto(0,0)

head.direction = "stop"

#Snake Food

food = turtle.Turtle()

food.speed(0)

food.shape("circle")

food.color("white")

food.penup()

food.goto(0,100)

segments = []

#pen

pen = turtle.Turtle()

pen.speed(0)

pen.shape("square")

pen.color("white")

pen.penup()

pen.hideturtle()

pen.goto(0, 260)

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

#Function

def go\_up():

if head.direction != "dov":

head.direction = "up"

def go\_dov():

if head.direction != "up":

head.direction = "dov"

def go\_left():

if head.direction != "right":

head.direction = "left"

def go\_right():

if head.direction != "left":

head.direction = "right"

#functions

def move():

if head.direction == "up":

y = head.ycor()

head.sety(y+20)

if head.direction == "dov":

y = head.ycor()

head.sety(y-20)

if head.direction == "left":

x = head.xcor()

head.setx(x-20)

if head.direction == "right":

x = head.xcor()

head.setx(x+20)

#Keyboard bindings

v.listen()

v.onkeypress(go\_up, "w")

v.onkeypress(go\_dov, "s")

v.onkeypress(go\_left, "a")

v.onkeypress(go\_right, "d")

#Main game Loop

while True:

v.update()

#check for a collision with the border

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

time.sleep(1)

head.goto(0,0)

head.direction = "stop"

#Hide the segments

for segment in segments:

segment.goto(1000, 1000)

#clear the segments list

segments.clear()

# Reset the score

score = 0

# Reset the delay

delay = 0.1

pen.clear()

pen.write("Score: {} High Score: {}".format(score, high\_score), align="center", font=("Courier", 24, "normal"))

#check for a collision with the food

if head.distance(food) < 20:

#Move the food to a random place

x = random.randint(-290,290)

y = random.randint(-290, 290)

food.goto(x,y)

#Add a segment

new\_segment=turtle.Turtle()

new\_segment.speed(0) #Animation speed

new\_segment.shape("square")

new\_segment.color("white")

new\_segment.penup()

segments.append(new\_segment) #to append the empty segments to new segment

# Shorten the delay

delay -= 0.001

# Increase the score

score += 10

if score > high\_score:

high\_score = score

pen.clear()

pen.write("Score: {} High Score: {}".format(score, high\_score), align="center", font=("Courier", 24, "normal"))

#move the end segments first in reverse order

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

x = segments[index-1].xcor()

y = segments[index-1].ycor()

segments[index].goto(x, y)

#move segment 0 to where the head is

if len(segments)>0:

x = head.xcor()

y = head.ycor()

segments[0].goto(x,y)

move()

# check for head collision with the body segments

for segment in segments:

if segment.distance(head)<20:

time.sleep(1)

head.goto(0, 0)

head.direction = "stop"

# hide the segments

for segment in segments:

segments.goto(1000, 1000)

# Clear the segments list

segments.clear()

# Reset the score

score = 0

# Reset the delay

delay = 0.1

# Update the score display

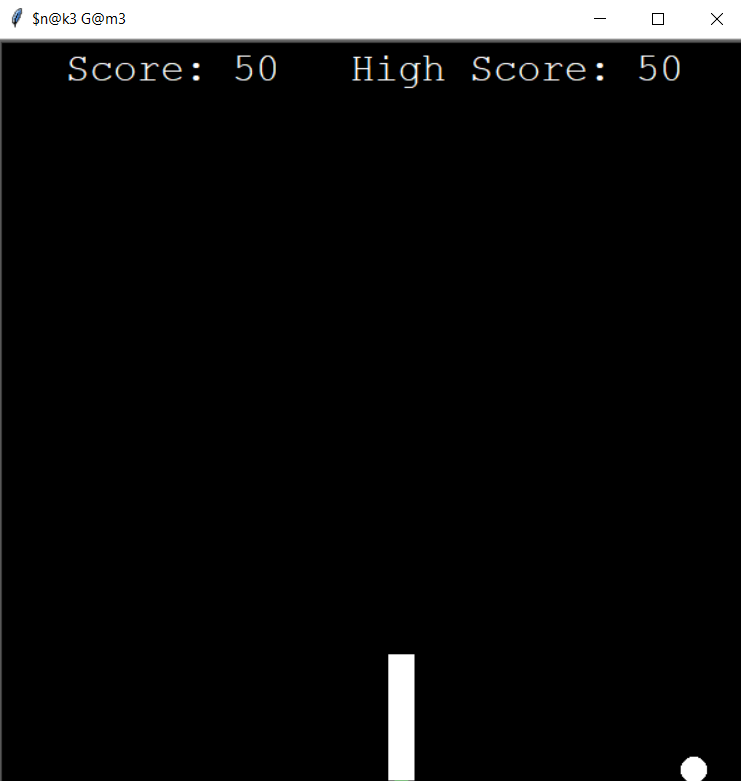
pen.clear()

pen.write("Score: {} High Score: {}".format(score, high\_score), align="center", font=("Courier", 24, "normal"))

time.sleep(delay)

v.mainloop()

1. **Output:**



1. **References:**

* <https://interactivepython.org/runestone/static/CS152f17/PythonTurtle/OurFirstTurtleProgram.html>
* <https://www.youtube.com/watch?v=BP7KMlbvtOo&list=PLlEgNdBJEO-n8k9SR49AshB9j7b5Iw7hZ>