

**PYTHON PROJECT REPORT**

**Snake Game with Python Project**



**BACHELOR OF TECHNOLOGY ( CSE )**

PRAKASH CHANDRA KUMAR (100554)

**Under the guidance of**

**Ms. Neny Pandel (Assistant professor)**

**Project Name: - Snake Game**

**GYAN VIHAR SCHOOL OF ENGINEERING AND TECHNOLOGY**

DEPARTMENT OF COMPUTER SCIENCE

**SURESH GYAN VIHAR UNIVERSITY**

Approved by NAAC with ‘A’ Grade, and now NIRF Ranked University

**JAIPUR, RAJASTHAN 302017**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Contents** | **Page No.** |
| **1** | **ABSTRACT** | **3** |
| **2** | **Introduction** | **4** |
| **3** | **TABLE OF CONTENTS** | **6** |
| **4** | **source code of the program** | **25** |
| **5** | **output** | **26** |
| **6** | **CONCLUSION** | **27** |
| **7** | **REFERENCES** | **27** |
| **8** |  |  |
| **9** |  |  |

**ABSTRACT**

Our topic for the project-based learning (PBL) for Python Engineering is Snake Game . The Nokia inspired Snake game implemented on 8051 platform. The game is built on AT89S51 Microcontroller. The game uses an 8\*8 LED dot matrix display and five way keys for user interface. A dedicated delay settings key provides one touch access to the delay settings mode where the speed of the Snake can be adjusted.

In our project we are implementing the snake game by using microcontroller. In this we are using switches one is Start for starting of the game and remaining switches for snake movements left, right, up and down these switches used like as a buttons in mobile. We are going to develop coding by using Embedded C according to that coding by using dot matrix the snake and egg will be display on 8\*8 dot-matrix and by using switches the game will be operate according to our requirements.



INTRODUCTION

Snake game is a classic computer action game, in which we control a snake to move and collect food in a map. In this paper we

are interested in developing a controller with artificial intelligence (AI) using movement rating functions and evolutionary

algorithms (EA). Before we elaborate how we design our controller, we first describe the snake game we implemented and define

the objective.

The game environment is a 30×30 map. The snake starts with length one. At each

time step the snake moves one step, and it can go straight, turn left, or turn right.

There are two kinds of food in our game, apples and bananas. Eating an apple

increases the length of the snake by one, whereas eating a banana decreases the

length by two. There is always one apple and at most one banana in the map. One

banana appears after every 30 apples are eaten. We can play the original snake game

with a simple strategy: we move the snake up and down from the right to the left

without touching the last row (the row at the bottom); when the snake is close to the

leftmost boundary, go back to the rightmost boundary along the last row. In this way,

the snake keeps alive until it occupies the whole environment.

The objective of our game is to maximize the score. The above-mentioned simple strategy may keep the snake alive, but

without moving toward the apples efficiently it cannot get a high score. Thus, we need to design a more intelligent controller, which

is the topic of this paper. The rest of this paper is organized as follows. Section II reviews related work about computer game design

using EAs. The main component of our AI controller is a set of rating functions, which will be described in Section III. We

aggregate these functions through weighted sum, and Section IV presents how we set weights through an EA. Section V gives the

experiments and results. Conclusions are made in Section VI

****

## 

## **TABLE OF CONTENTS**

## **What is Snake Game In Python Code?**

The **Snake Game In Python Code** is a fully functional desktop application project that covers all of the elements that IT students and computer-related courses will require for their college projects or for leisure time purposes. I’m sure you’ve all played the Snake game, and you’ve all vowed never to lose. We all loved looking for shortcuts as kids to avoid seeing the “Game Over” warning, but I’m sure you’d want to make this “Snake” dance to your beats as a techie. This is what I’m going to show you all in this article. This **Code For Snake Game In Python** is useful for learning new skills and practicing Python Game development. This project is quite useful, and the concept and logic of the project are simple to grasp. The source code is open source and free to use. Simply scroll down and click the download option.

## **What is the main goal of the Snake Game?**

This **Source Code For Snake Game In Python** is one of the most popular arcade games of all time. The player’s main aim in this game is to capture as many fruits as possible without striking the wall or himself. While learning Python or Pygame, making a snake game might be a fun challenge. It’s one of the most beginner-friendly projects out there, and it’s one that every new programmer should take on as a challenge. Learning to make a video game is a fascinating and enjoyable experience.

* **What is the logic behind Snake Game In Python?**

This Make Snake Game In Python has a very simple logic which every node in the snake will be represented by an object that tracks X and Y coordinates. You’ll use the same logic, but each node will be moved to the preceding node’s position before the head is moved to the new place. There will be no need to depict anything using numbers.

* **About the Project : Snake Game In Python Code**

The **Coding For Snake Game In Python** is a simple desktop files as well as a Python script. The graphics of the game are smooth, and the controls are simple.

The player controls a long, thin snake-like creature that moves around on a bordered plane, picking up thus apple while avoiding hitting its own tail or the playing area’s edges.

This **Snake Game In Python Source Code** includes a tutorial and a code development guide. This is a simple and basic-level little project for learning purposes. Snake game is open source, so you can download the zip file and change it to fit your needs. You can also customize this project to meet your needs and application made using the Python programming language. We may also create highly fascinating games with the Python programming language. The snake game is one of them. The project system file comprises resource create a fantastic advanced-level project.

**SOURCE CODE of the program**

import pygame

import sys

import os

import random

import math

pygame.init()

pygame.display.set\_caption("Snake Game")

pygame.font.init()

random.seed()

SPEED = 0.36

SNAKE\_SIZE = 9

APPLE\_SIZE = SNAKE\_SIZE

SEPARATION = 10

SCREEN\_HEIGHT = 600

SCREEN\_WIDTH = 800

FPS = 25

KEY = {"UP": 1, "DOWN": 2, "LEFT": 3, "RIGHT": 4}

screen = pygame.display.set\_mode((SCREEN\_WIDTH, SCREEN\_HEIGHT), pygame.HWSURFACE)

score\_font = pygame.font.Font(None, 38)

score\_numb\_font = pygame.font.Font(None, 28)

game\_over\_font = pygame.font.Font(None, 46)

play\_again\_font = score\_numb\_font

score\_msg = score\_font.render("Score : ", 1, pygame.Color("green"))

score\_msg\_size = score\_font.size("Score")

background\_color = pygame.Color(0, 0, 0)  # we will fill background color as black

black = pygame.Color(0, 0, 0)

gameClock = pygame.time.Clock()

def checkCollision(posA, As, posB, Bs):

    if (posA.x < posB.x + Bs and posA.x + As > posB.x and posA.y < posB.y + Bs and posA.y + As > posB.y):

        return True

    return False

def checkLimits(snake):

    if (snake.x > SCREEN\_WIDTH):

        snake.x = SNAKE\_SIZE

    if (snake.x < 0):

        snake.x = SCREEN\_WIDTH - SNAKE\_SIZE

    if (snake.y > SCREEN\_HEIGHT):

        snake.y = SNAKE\_SIZE

    if (snake.y < 0):

        snake.y = SCREEN\_HEIGHT - SNAKE\_SIZE

class Apple:

    def \_\_init\_\_(self, x, y, state):

        self.x = x

        self.y = y

        self.state = state

        self.color = pygame.color.Color("orange")  # color of food

    def draw(self, screen):

        pygame.draw.rect(screen, self.color, (self.x, self.y, APPLE\_SIZE, APPLE\_SIZE), 0)

class segment:

    def \_\_init\_\_(self, x, y):

        self.x = x

        self.y = y

        self.direction = KEY["UP"]

        self.color = "white"

class snake:

    def \_\_init\_\_(self, x, y):

        self.x = x

        self.y = y

        self.direction = KEY["UP"]

        self.stack = []

        self.stack.append(self)

        blackBox = segment(self.x, self.y + SEPARATION)

        blackBox.direction = KEY["UP"]

        blackBox.color = "NULL"

        self.stack.append(blackBox)

    def move(self):

        last\_element = len(self.stack) - 1

        while (last\_element != 0):

            self.stack[last\_element].direction = self.stack[last\_element - 1].direction

            self.stack[last\_element].x = self.stack[last\_element - 1].x

            self.stack[last\_element].y = self.stack[last\_element - 1].y

            last\_element -= 1

        if (len(self.stack) < 2):

            last\_segment = self

        else:

            last\_segment = self.stack.pop(last\_element)

        last\_segment.direction = self.stack[0].direction

        if (self.stack[0].direction == KEY["UP"]):

            last\_segment.y = self.stack[0].y - (SPEED \* FPS)

        elif (self.stack[0].direction == KEY["DOWN"]):

            last\_segment.y = self.stack[0].y + (SPEED \* FPS)

        elif (self.stack[0].direction == KEY["LEFT"]):

            last\_segment.x = self.stack[0].x - (SPEED \* FPS)

        elif (self.stack[0].direction == KEY["RIGHT"]):

            last\_segment.x = self.stack[0].x + (SPEED \* FPS)

        self.stack.insert(0, last\_segment)

    def getHead(self):

        return (self.stack[0])

    def grow(self):

        last\_element = len(self.stack) - 1

        self.stack[last\_element].direction = self.stack[last\_element].direction

        if (self.stack[last\_element].direction == KEY["UP"]):

            newSegment = segment(self.stack[last\_element].x, self.stack[last\_element].y - SNAKE\_SIZE)

            blackBox = segment(newSegment.x, newSegment.y - SEPARATION)

        elif (self.stack[last\_element].direction == KEY["DOWN"]):

            newSegment = segment(self.stack[last\_element].x, self.stack[last\_element].y + SNAKE\_SIZE)

            blackBox = segment(newSegment.x, newSegment.y + SEPARATION)

        elif (self.stack[last\_element].direction == KEY["LEFT"]):

            newSegment = segment(self.stack[last\_element].x - SNAKE\_SIZE, self.stack[last\_element].y)

            blackBox = segment(newSegment.x - SEPARATION, newSegment.y)

        elif (self.stack[last\_element].direction == KEY["RIGHT"]):

            newSegment = segment(self.stack[last\_element].x + SNAKE\_SIZE, self.stack[last\_element].y)

            blackBox = segment(newSegment.x + SEPARATION, newSegment.y)

        blackBox.color = "NULL"

        self.stack.append(newSegment)

        self.stack.append(blackBox)

    def iterateSegments(self, delta):

        pass

    def setDirection(self, direction):

        if (self.direction == KEY["RIGHT"] and direction == KEY["LEFT"] or self.direction == KEY["LEFT"] and

                direction == KEY["RIGHT"]):

            pass

        elif (self.direction == KEY["UP"] and direction == KEY["DOWN"] or self.direction == KEY["UP"] and

              direction == KEY["DOWN"]):

            pass

        else:

            self.direction = direction

    def get\_rect(self):

        rect = (self.x, self.y)

        return rect

    def getX(self):

        return self.x

    def getY(self):

        return self.y

    def setX(self, x):

        self.x = x

    def setY(self, y):

        self.y = y

    def checkCrashing(self):

        counter = 1

        while (counter < len(self.stack) - 1):

            if (checkCollision(self.stack[0], SNAKE\_SIZE, self.stack[counter], SNAKE\_SIZE) and

                    self.stack[counter].color != "NULL"):

                return True

            counter += 1

        return False

    def draw(self, screen):

        pygame.draw.rect(screen, pygame.color.Color("green"), (self.stack[0].x, self.stack[0].y,

                                                               SNAKE\_SIZE, SNAKE\_SIZE), 0)

        counter = 1

        while (counter < len(self.stack)):

            if (self.stack[counter].color == "NULL"):

                counter += 1

                continue

            pygame.draw.rect(screen, pygame.color.Color("yellow"), (self.stack[counter].x,

                                                                    self.stack[counter].y, SNAKE\_SIZE, SNAKE\_SIZE), 0)

            counter += 1

def getKey():

    for event in pygame.event.get():

        if event.type == pygame.KEYDOWN:

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

                return KEY["UP"]

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

                return KEY["DOWN"]

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

                return KEY["LEFT"]

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

                return KEY["RIGHT"]

            elif event.key == pygame.K\_ESCAPE:

                return "exit"

            elif event.key == pygame.K\_y:

                return "yes"

            elif event.key == pygame.K\_n:

                return "no"

        if event.type == pygame.QUIT:

            sys.exit(0)

def endGame():

    message = game\_over\_font.render("Gsme Over", 1, pygame.Color("white"))

    message\_play\_again = play\_again\_font.render("Play Again ? (Y/N)", 1, pygame.Color("green"))

    screen.blit(message, (320, 240))

    screen.blit(message\_play\_again, (320 + 12, 240 + 40))

    pygame.display.flip()

    pygame.display.update()

    mKey = getKey()

    while (mKey != "exit"):

        if (mKey == "yes"):

            main()

        elif (mKey == "no"):

            break

        mKey = getKey()

        gameClock.tick(FPS)

    sys.exit(0)

def drawScore(score):

    score\_numb = score\_numb\_font.render(str(score), 1, pygame.Color("red"))

    screen.blit(score\_msg, (SCREEN\_WIDTH - score\_msg\_size[0] - 60, 10))

    screen.blit(score\_numb, (SCREEN\_WIDTH - 45, 14))

def drawGameTime(gameTime):

    game\_time = score\_font.render("Time:", 1, pygame.Color("white"))

    game\_time\_numb = score\_numb\_font.render(str(gameTime / 1000), 1, pygame.Color("white"))

    screen.blit(game\_time, (30, 10))

    screen.blit(game\_time\_numb, (105, 14))

def exitScreen():

    pass

def respawnApple(apples, index, sx, sy):

    radius = math.sqrt((SCREEN\_WIDTH / 2 \* SCREEN\_WIDTH / 2 + SCREEN\_HEIGHT / 2 \* SCREEN\_HEIGHT / 2)) / 2

    angle = 999

    while (angle > radius):

        angle = random.uniform(0, 800) \* math.pi \* 2

        x = SCREEN\_WIDTH / 2 + radius \* math.cos(angle)

        y = SCREEN\_HEIGHT / 2 + radius \* math.sin(angle)

        if (x == sx and y == sy):

            continue

    newApple = Apple(x, y, 1)

    apples[index] = newApple

def respawnApples(apples, quantity, sx, sy):

    counter = 0

    del apples[:]

    radius = math.sqrt((SCREEN\_WIDTH / 2 \* SCREEN\_WIDTH / 2 + SCREEN\_HEIGHT / 2 \* SCREEN\_HEIGHT / 2)) / 2

    angle = 999

    while (counter < quantity):

        while (angle > radius):

            angle = random.uniform(0, 800) \* math.pi \* 2

            x = SCREEN\_WIDTH / 2 + radius \* math.cos(angle)

            y = SCREEN\_HEIGHT / 2 + radius \* math.sin(angle)

            if ((x - APPLE\_SIZE == sx or x + APPLE\_SIZE == sx) and (y - APPLE\_SIZE == sy or y + APPLE\_SIZE == sy)

                    or radius - angle <= 10):

                continue

        apples.append(Apple(x, y, 1))

        angle = 999

        counter += 1

def main():

    score = 0

    mySnake = snake(SCREEN\_WIDTH / 2, SCREEN\_HEIGHT / 2)

    mySnake.setDirection(KEY["UP"])

    mySnake.move()

    start\_segments = 3

    while (start\_segments > 0):

        mySnake.grow()

        mySnake.move()

        start\_segments -= 1

    max\_apples = 1

    eaten\_apple = False

    apples = [Apple(random.randint(60, SCREEN\_WIDTH), random.randint(60, SCREEN\_HEIGHT), 1)]

    respawnApples(apples, max\_apples, mySnake.x, mySnake.y)

    startTime = pygame.time.get\_ticks()

    endgame = 0

    while (endgame != 1):

        gameClock.tick(FPS)

        keyPress = getKey()

        if keyPress == "exit":

            endgame = 1

        checkLimits(mySnake)

        if (mySnake.checkCrashing() == True):

            endGame()

        for myApple in apples:

            if (myApple.state == 1):

                if (checkCollision(mySnake.getHead(), SNAKE\_SIZE, myApple, APPLE\_SIZE) == True):

                    mySnake.grow()

                    myApple.state = 0

                    score += 10

                    eaten\_apple = True

        if (keyPress):

            mySnake.setDirection(keyPress)

        mySnake.move()

        if (eaten\_apple == True):

            eaten\_apple = False

            respawnApple(apples, 0, mySnake.getHead().x, mySnake.getHead().y)

        screen.fill(background\_color)

        for myApple in apples:

            if (myApple.state == 1):

                myApple.draw(screen)

        mySnake.draw(screen)

        drawScore(score)

        gameTime = pygame.time.get\_ticks() - startTime

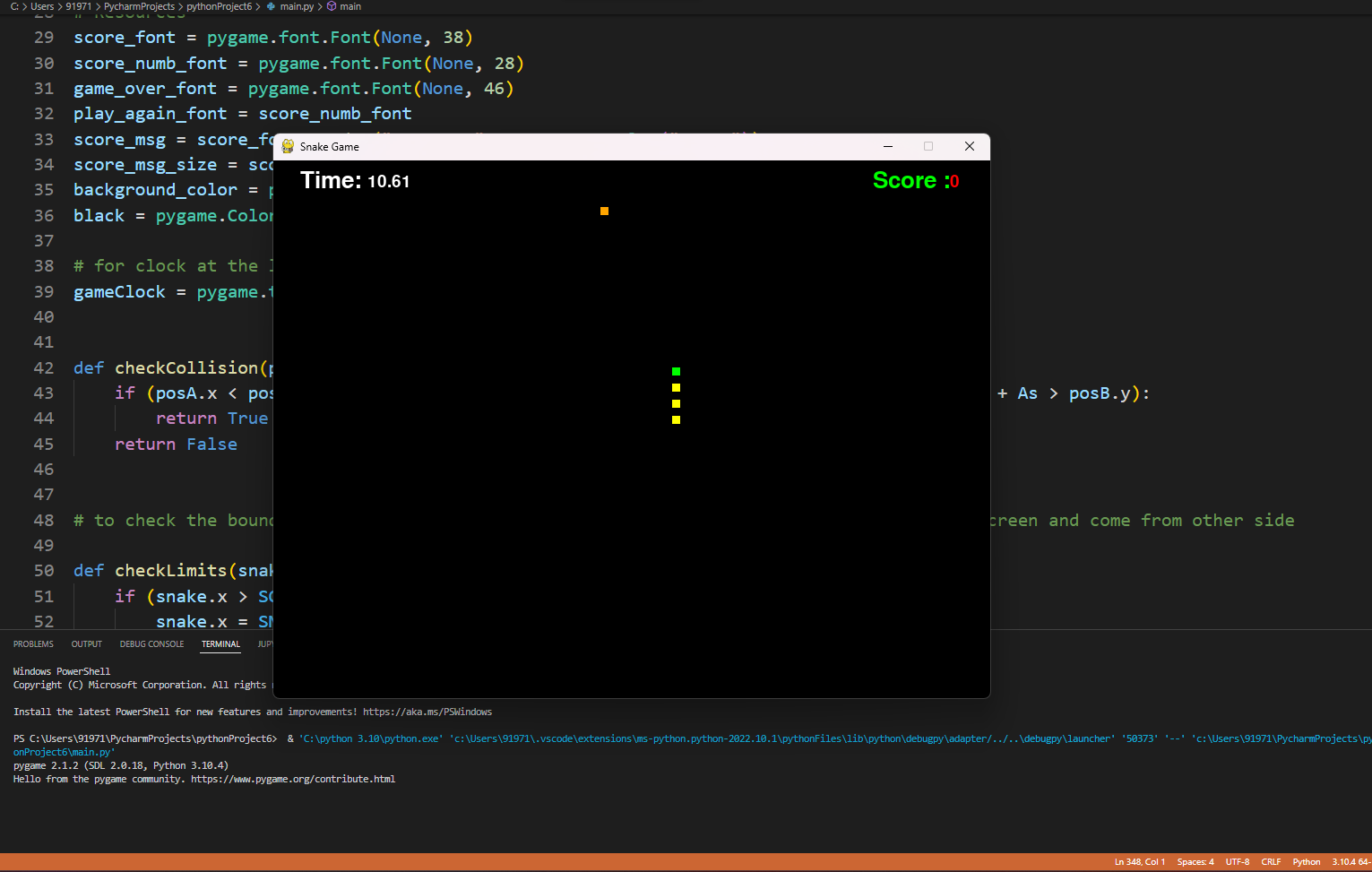
        drawGameTime(gameTime)

        pygame.display.flip()

        pygame.display.update()

main()

**OUTPUT**

****

**CONCLUSION**

Hence Successfully completed and executed the project “ SNAKE GAME ”

**REFERENCES**

[**https://www.slideshare.net/MuhammadAziz102/final-project-report-snake-game-in-python**](https://www.slideshare.net/MuhammadAziz102/final-project-report-snake-game-in-python)

[**https://www.youtube.com/watch?v=BKwnOOs0ml8**](https://www.youtube.com/watch?v=BKwnOOs0ml8)

[**https://www.youtube.com/watch?v=2ZDnw6ifdSI**](https://www.youtube.com/watch?v=2ZDnw6ifdSI)

[**https://en.wikipedia.org/wiki/Snake\_(video\_game\_genre)**](https://en.wikipedia.org/wiki/Snake_(video_game_genre))

[**https://www.geeksforgeeks.org/design-snake-game/**](https://www.geeksforgeeks.org/design-snake-game/)

