

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI ERODE – 638 060

Snake Game

A Micro Project Report

submitted by

Bharathi T

23ITR015

Chiba Roshan A

23ITR019

Elakya R

23ITR045

PYTHON PROGRAMMING AND FRAMEWORKS ( 22ITT32 )

DEPARTMENT OF INFORMATION TECHNOLOGY

Description:

The Enhanced Snake Game project revamps the classic Snake Game by incorporating modern features like user authentication and high score tracking, aiming to enrich the gaming experience through personalization and replayability. Developed with Python and the Pygame library, this project maintains the original game's simple mechanics while adding user-focused functionalities. Players can create unique profiles, log in securely, and track their high scores, encouraging continued engagement with the game. Upon starting the game, players are welcomed with a login screen that enables registration or authentication, creating a distinct identity for each user. Successfully authenticated players proceed to the main gameplay, where they control a snake that grows with each food item consumed. The core challenge remains intact, as the player must avoid collisions with the walls and the snake's own body, with the difficulty level naturally increasing as the snake's length extends. Each player’s high score is tracked and displayed during gameplay, allowing them to monitor and improve upon their previous best performance. This scoring system is supported by a local database that securely stores user credentials and high scores, adding a sense of accomplishment and motivation to the gameplay. The graphical interface and input handling are managed via Pygame, providing a smooth, interactive experience. This project serves as a demonstration of core programming skills, covering user authentication, database management, and graphical rendering in real-time, all within an entertaining context. Through this blend of classic gameplay and modern features, the Enhanced Snake Game offers a fun yet structured approach to learning foundational software development concepts. The end result is an engaging game that combines nostalgia with a personalized, challenging environment, encouraging users to improve their scores with each session.

|  |  |
| --- | --- |
|  |  |

**CODING:**

1.PYGAME:

import pygame

import random

import time

import sqlite3

pygame.init()

white = (255, 255, 255)

bright\_red = (255, 0, 0)

green = (0, 255, 0)

blue = (0, 100, 255)

black = (0, 0, 0)

yellow = (255, 255, 102)

gray = (169, 169, 169)

dis\_width = 700

dis\_height = 500

dis = pygame.display.set\_mode((dis\_width, dis\_height))

pygame.display.set\_caption('Enhanced Snake Game')

background\_image = pygame.image.load('gamestart.jpg')

background\_image = pygame.transform.scale(background\_image, (dis\_width, dis\_height))

game\_background = pygame.image.load('gamebg.jpg')

game\_background = pygame.transform.scale(game\_background, (dis\_width, dis\_height))

clock = pygame.time.Clock()

font\_style = pygame.font.SysFont("bahnschrift", 25)

score\_font = pygame.font.SysFont("comicsansms", 35)

input\_font = pygame.font.SysFont("bahnschrift", 30)

snake\_block = 10

error\_message = ''

error\_timer = 0

def create\_database():

    conn = sqlite3.connect("snake\_game.db")

    cursor = conn.cursor()

    cursor.execute("""CREATE TABLE IF NOT EXISTS users (

        username TEXT PRIMARY KEY,

        password TEXT,

        high\_score INTEGER DEFAULT 0)""")

    conn.commit()

    conn.close()

def register\_user(username, password):

    conn = sqlite3.connect("snake\_game.db")

    cursor = conn.cursor()

    cursor.execute("INSERT OR IGNORE INTO users (username, password) VALUES (?, ?)", (username, password))

    conn.commit()

    conn.close()

def validate\_user(username, password):

    conn = sqlite3.connect("snake\_game.db")

    cursor = conn.cursor()

    cursor.execute("SELECT \* FROM users WHERE username = ? AND password = ?", (username, password))

    result = cursor.fetchone()

    conn.close()

    return result is not None

def get\_high\_score(username):

    conn = sqlite3.connect("snake\_game.db")

    cursor = conn.cursor()

    cursor.execute("SELECT high\_score FROM users WHERE username = ?", (username,))

    result = cursor.fetchone()

    conn.close()

    return result[0] if result else 0

def update\_high\_score(username, new\_score):

    conn = sqlite3.connect("snake\_game.db")

    cursor = conn.cursor()

    cursor.execute("UPDATE users SET high\_score = ? WHERE username = ? AND high\_score < ?", (new\_score, username, new\_score))

    conn.commit()

    conn.close()

def display\_score(score, high\_score):

    score\_text = score\_font.render(f"Your Score: {score}  |  High Score: {high\_score}", True, bright\_red)

    dis.blit(score\_text, [0, 0])

    pygame.draw.line(dis, black, (0, 50), (dis\_width, 50), 4)

def draw\_snake(snake\_block, snake\_list):

    for x in snake\_list:

        pygame.draw.rect(dis, bright\_red, [x[0], x[1], snake\_block, snake\_block])

def message(msg, color):

    mesg = font\_style.render(msg, True, color)

    dis.blit(mesg, [dis\_width / 6, dis\_height / 3])

def button(msg, x, y, w, h, ic, ac, action=None):

    mouse = pygame.mouse.get\_pos()

    click = pygame.mouse.get\_pressed()

    if x + w > mouse[0] > x and y + h > mouse[1] > y:

        pygame.draw.rect(dis, ac, (x, y, w, h))

        if click[0] == 1 and action is not None:

            action()

    else:

        pygame.draw.rect(dis, ic, (x, y, w, h))

    text\_surface = input\_font.render(msg, True, black)

    dis.blit(text\_surface, (x + (w - text\_surface.get\_width()) // 2, y + (h - text\_surface.get\_height()) // 2))

def welcome\_screen():

    while True:

        dis.blit(background\_image, (0, 0))

        button("Play", 250, 250, 200, 50, green, blue, login\_screen)

        for event in pygame.event.get():

            if event.type == pygame.QUIT:

                pygame.quit()

                quit()

        pygame.display.update()

        clock.tick(30)

def login\_screen():

    global error\_message, error\_timer

    username = ''

    password = ''

    active\_username = False

    active\_password = False

    input\_box\_username = pygame.Rect(250, 200, 400, 50)

    input\_box\_password = pygame.Rect(250, 280, 400, 50)

    while True:

        dis.blit(background\_image, (0, 0))

        username\_label = input\_font.render("Username:", True, yellow)

        password\_label = input\_font.render("Password:", True, yellow)

        dis.blit(username\_label, (50, 210))

        dis.blit(password\_label, (50, 290))

        for event in pygame.event.get():

            if event.type == pygame.QUIT:

                pygame.quit()

                quit()

            if event.type == pygame.MOUSEBUTTONDOWN:

                if input\_box\_username.collidepoint(event.pos):

                    active\_username = True

                    active\_password = False

                elif input\_box\_password.collidepoint(event.pos):

                    active\_password = True

                    active\_username = False

                else:

                    active\_username = False

                    active\_password = False

            if event.type == pygame.KEYDOWN:

                if active\_username:

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

                        username = username[:-1]

                    else:

                        username += event.unicode

                if active\_password:

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

                        password = password[:-1]

                    else:

                        password += event.unicode

        pygame.draw.rect(dis, white if active\_username else gray, input\_box\_username)

        pygame.draw.rect(dis, white if active\_password else gray, input\_box\_password)

        dis.blit(input\_font.render(username, True, black), (input\_box\_username.x + 5, input\_box\_username.y + 5))

        dis.blit(input\_font.render(password, True, black), (input\_box\_password.x + 5, input\_box\_password.y + 5))

        button("Login", 250, 350, 180, 50, green, blue, lambda: attempt\_login(username, password))

        button("Register", 440, 350, 180, 50, green, blue, lambda: attempt\_register(username, password))

        if error\_message:

            error\_surface = font\_style.render(error\_message, True, yellow)

            dis.blit(error\_surface, (dis\_width / 6, 180))

            if time.time() - error\_timer > 2:

                error\_message = ''

        pygame.display.update()

        clock.tick(30)

def attempt\_login(username, password):

    global error\_message, error\_timer

    if username and password:

        if validate\_user(username, password):

            game\_loop(username, 15)

            error\_message = ''

        else:

            error\_message = "Invalid login details."

            error\_timer = time.time()

    else:

        error\_message = "Username and password cannot be empty."

        error\_timer = time.time()

def attempt\_register(username, password):

    global error\_message, error\_timer

    if username and password:

        register\_user(username, password)

        error\_message = "Registration successful!"

    else:

        error\_message = "Username and password cannot be empty."

    error\_timer = time.time()

def game\_loop(player\_name, snake\_speed):

    game\_over = False

    game\_close = False

    x1, y1 = dis\_width / 2, dis\_height / 2

    x1\_change, y1\_change = 0, 0

    snake\_list = []

    length\_of\_snake = 1

    foodx = round(random.randrange(0, dis\_width - snake\_block) / 10.0) \* 10.0

    foody = round(random.randrange(35, dis\_height - snake\_block) / 10.0) \* 10.0

    high\_score = get\_high\_score(player\_name)

    direction = 'STOP'

    while not game\_over:

        while game\_close:

            dis.blit(game\_background, (0, 0))

            message("You Lost! Press Q-Quit or C-Play Again", bright\_red)

            display\_score(length\_of\_snake - 1, high\_score)

            pygame.display.update()

            for event in pygame.event.get():

                if event.type == pygame.KEYDOWN:

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

                        if length\_of\_snake - 1 > high\_score:

                            update\_high\_score(player\_name, length\_of\_snake - 1)

                        game\_over = True

                        game\_close = False

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

                        if length\_of\_snake - 1 > high\_score:

                            update\_high\_score(player\_name, length\_of\_snake - 1)

                        game\_loop(player\_name, snake\_speed)

        for event in pygame.event.get():

            if event.type == pygame.QUIT:

                game\_over = True

            elif event.type == pygame.KEYDOWN:

                if event.key == pygame.K\_LEFT and direction != 'RIGHT':

                    x1\_change, y1\_change = -snake\_block, 0

                    direction = 'LEFT'

                elif event.key == pygame.K\_RIGHT and direction != 'LEFT':

                    x1\_change, y1\_change = snake\_block, 0

                    direction = 'RIGHT'

                elif event.key == pygame.K\_UP and direction != 'DOWN':

                    x1\_change, y1\_change = 0, -snake\_block

                    direction = 'UP'

                elif event.key == pygame.K\_DOWN and direction != 'UP':

                    x1\_change, y1\_change = 0, snake\_block

                    direction = 'DOWN'

        if x1 >= dis\_width or x1 < 0 or y1 >= dis\_height or y1 < 35:

            game\_close = True

        x1 += x1\_change

        y1 += y1\_change

        dis.blit(game\_background, (0, 0))

        pygame.draw.rect(dis, yellow, [foodx, foody, snake\_block, snake\_block])

        snake\_head = [x1, y1]

        snake\_list.append(snake\_head)

        if len(snake\_list) > length\_of\_snake:

            del snake\_list[0]

        for block in snake\_list[:-1]:

            if block == snake\_head:

                game\_close = True

        draw\_snake(snake\_block, snake\_list)

        display\_score(length\_of\_snake - 1, high\_score)

        pygame.draw.line(dis, black, (0, 50), (dis\_width, 50), 4)

        pygame.display.update()

        if x1 == foodx and y1 == foody:

            foodx = round(random.randrange(0, dis\_width - snake\_block) / 10.0) \* 10.0

            foody = round(random.randrange(35, dis\_height - snake\_block) / 10.0) \* 10.0

            length\_of\_snake += 1

        clock.tick(snake\_speed)

    pygame.quit()

    quit()

create\_database()

welcome\_screen()

2.GAME.PY:

import pygame

import random

from data\_db import get\_high\_score, update\_high\_score

pygame.init()

white = (255, 255, 255)

bright\_red = (255, 0, 0)

green = (0, 255, 0)

blue = (0, 100, 255)

black = (0, 0, 0)

yellow = (255, 255, 102)

gray = (169, 169, 169)

dis\_width = 700

dis\_height = 500

dis = pygame.display.set\_mode((dis\_width, dis\_height))

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

game\_background = pygame.image.load('gamebg.jpg')

game\_background = pygame.transform.scale(game\_background, (dis\_width, dis\_height))

font\_style = pygame.font.SysFont("bahnschrift", 25)

score\_font = pygame.font.SysFont("comicsansms", 35)

input\_font = pygame.font.SysFont("bahnschrift", 30)

clock = pygame.time.Clock()

snake\_block = 10

def message(msg, color):

    mesg = font\_style.render(msg, True, color)

    dis.blit(mesg, [dis\_width / 6, dis\_height / 3])

def display\_score(score, high\_score):

    score\_text = score\_font.render(f"Your Score: {score}  |  High Score: {high\_score}", True, bright\_red)

    dis.blit(score\_text, [0, 0])

    pygame.draw.line(dis, black, (0, 50), (dis\_width, 50), 4)

def draw\_snake(snake\_block, snake\_list):

    for x in snake\_list:

        pygame.draw.rect(dis, bright\_red, [x[0], x[1], snake\_block, snake\_block])

def game\_loop(player\_name, snake\_speed):

    game\_over = False

    game\_close = False

    x1 = dis\_width / 2

    y1 = dis\_height / 2

    x1\_change = 0

    y1\_change = 0

    snake\_list = []

    length\_of\_snake = 1

    foodx = round(random.randrange(0, dis\_width - snake\_block) / 10.0) \* 10.0

    foody = round(random.randrange(50, dis\_height - snake\_block) / 10.0) \* 10.0

    high\_score = get\_high\_score(player\_name)

    while not game\_over:

        while game\_close:

            dis.blit(game\_background, (0, 0))

            message("You Lost! Press Q-Quit or C-Play Again", bright\_red)

            display\_score(length\_of\_snake - 1, high\_score)

            pygame.display.update()

            for event in pygame.event.get():

                if event.type == pygame.KEYDOWN:

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

                        if length\_of\_snake - 1 > high\_score:

                            update\_high\_score(player\_name, length\_of\_snake - 1)

                        game\_over = True

                        game\_close = False

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

                        if length\_of\_snake - 1 > high\_score:

                            update\_high\_score(player\_name, length\_of\_snake - 1)

                        game\_loop(player\_name, snake\_speed)

        for event in pygame.event.get():

            if event.type == pygame.QUIT:

                game\_over = True

            elif event.type == pygame.KEYDOWN:

                if event.key == pygame.K\_LEFT and x1\_change == 0:

                    x1\_change = -snake\_block

                    y1\_change = 0

                elif event.key == pygame.K\_RIGHT and x1\_change == 0:

                    x1\_change = snake\_block

                    y1\_change = 0

                elif event.key == pygame.K\_UP and y1\_change == 0:

                    y1\_change = -snake\_block

                    x1\_change = 0

                elif event.key == pygame.K\_DOWN and y1\_change == 0:

                    y1\_change = snake\_block

                    x1\_change = 0

        if x1 >= dis\_width or x1 < 0 or y1 >= dis\_height or y1 < 50:

            game\_close = True

        x1 += x1\_change

        y1 += y1\_change

        dis.blit(game\_background, (0, 0))

        pygame.draw.rect(dis, yellow, [foodx, foody, snake\_block, snake\_block])

        snake\_head = [x1, y1]

        snake\_list.append(snake\_head)

        if len(snake\_list) > length\_of\_snake:

            del snake\_list[0]

        for block in snake\_list[:-1]:

            if block == snake\_head:

                game\_close = True

        draw\_snake(snake\_block, snake\_list)

        display\_score(length\_of\_snake - 1, high\_score)

        pygame.display.update()

        if x1 == foodx and y1 == foody:

            foodx = round(random.randrange(0, dis\_width - snake\_block) / 10.0) \* 10.0

            foody = round(random.randrange(50, dis\_height - snake\_block) / 10.0) \* 10.0

            length\_of\_snake += 1

        clock.tick(snake\_speed)

    pygame.quit()

    quit()

3.WELCOME.PY:

import pygame

import time

from auth import attempt\_login, attempt\_register

from game import game\_loop

from data\_db import get\_high\_score, create\_database, register\_user

pygame.init()

white = (255, 255, 255)

bright\_red = (255, 0, 0)

green = (0, 255, 0)

blue = (0, 100, 255)

black = (0, 0, 0)

yellow = (255, 255, 102)

gray = (169, 169, 169)

dis\_width = 700

dis\_height = 500

dis = pygame.display.set\_mode((dis\_width, dis\_height))

pygame.display.set\_caption('Enhanced Snake Game')

background\_image = pygame.image.load('gamestart.jpg')

background\_image = pygame.transform.scale(background\_image, (dis\_width, dis\_height))

game\_background = pygame.image.load('gamebg.jpg')

game\_background = pygame.transform.scale(game\_background, (dis\_width, dis\_height))

font\_style = pygame.font.SysFont("bahnschrift", 25)

score\_font = pygame.font.SysFont("comicsansms", 35)

input\_font = pygame.font.SysFont("bahnschrift", 30)

clock = pygame.time.Clock()

error\_message = ''

error\_timer = 0

def render\_error\_message(msg, color):

    mesg = font\_style.render(msg, True, color)

    text\_width = mesg.get\_width()

    dis.blit(mesg, [dis\_width - text\_width - 20, 20])

def button(msg, x, y, w, h, ic, ac, action=None):

    mouse = pygame.mouse.get\_pos()

    click = pygame.mouse.get\_pressed()

    if x + w > mouse[0] > x and y + h > mouse[1] > y:

        pygame.draw.rect(dis, ac, (x, y, w, h))

        if click[0] == 1 and action is not None:

            action()

    else:

        pygame.draw.rect(dis, ic, (x, y, w, h))

    text\_surface = input\_font.render(msg, True, black)

    dis.blit(text\_surface, (x + (w - text\_surface.get\_width()) // 2, y + (h - text\_surface.get\_height()) // 2))

def welcome\_screen():

    global error\_message, error\_timer

    while True:

        dis.blit(background\_image, (0, 0))

        button("Play", 250, 250, 200, 50, green, blue, login\_screen)

        if error\_message:

            render\_error\_message(error\_message, bright\_red if 'Invalid' in error\_message else blue)

            if time.time() - error\_timer > 2:

                error\_message = ''

        for event in pygame.event.get():

            if event.type == pygame.QUIT:

                pygame.quit()

                quit()

        pygame.display.update()

        clock.tick(30)

def login\_screen():

    global error\_message, error\_timer

    username = ''

    password = ''

    active\_username = False

    active\_password = False

    input\_box\_username = pygame.Rect(250, 200, 400, 50)

    input\_box\_password = pygame.Rect(250, 280, 400, 50)

    while True:

        dis.blit(background\_image, (0, 0))

        username\_label = input\_font.render("Username:", True, yellow)

        password\_label = input\_font.render("Password:", True, yellow)

        dis.blit(username\_label, (50, 210))

        dis.blit(password\_label, (50, 290))

        for event in pygame.event.get():

            if event.type == pygame.QUIT:

                pygame.quit()

                quit()

            if event.type == pygame.MOUSEBUTTONDOWN:

                if input\_box\_username.collidepoint(event.pos):

                    active\_username = True

                    active\_password = False

                elif input\_box\_password.collidepoint(event.pos):

                    active\_password = True

                    active\_username = False

                else:

                    active\_username = False

                    active\_password = False

            if event.type == pygame.KEYDOWN:

                if active\_username:

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

                        username = username[:-1]

                    else:

                        username += event.unicode

                if active\_password:

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

                        password = password[:-1]

                    else:

                        password += event.unicode

        pygame.draw.rect(dis, white if active\_username else gray, input\_box\_username)

        pygame.draw.rect(dis, white if active\_password else gray, input\_box\_password)

        dis.blit(input\_font.render(username, True, black), (input\_box\_username.x + 5, input\_box\_username.y + 5))

        dis.blit(input\_font.render(password, True, black), (input\_box\_password.x + 5, input\_box\_password.y + 5))

        button("Login", 250, 350, 180, 50, green, blue, lambda: attempt\_login\_action(username, password))

        button("Register", 440, 350, 180, 50, green, blue, lambda: attempt\_register\_action(username, password))

        if error\_message:

            render\_error\_message(error\_message, bright\_red if 'Invalid' in error\_message else blue)

            if time.time() - error\_timer > 2:

                error\_message = ''

        pygame.display.update()

        clock.tick(30)

def attempt\_login\_action(username, password):

    global error\_message, error\_timer

    if not username or not password:

        error\_message = "Username and password cannot be empty!"

        error\_timer = time.time()

    elif attempt\_login(username, password):

        game\_loop(username, 15)

    else:

        error\_message = "Invalid login details."

        error\_timer = time.time()

def attempt\_register\_action(username, password):

    global error\_message, error\_timer

    if not username or not password:

        error\_message = "Username and password cannot be empty!"

        error\_timer = time.time()

    else:

        register\_user(username, password)

        error\_message = "Registration successful!"

        error\_timer = time.time()

welcome\_screen()

4.DATA.DB\_PY:

import sqlite3

def create\_database():

    connection = sqlite3.connect("snake\_game.db")

    cursor = connection.cursor()

    cursor.execute('''CREATE TABLE IF NOT EXISTS users (

                        id INTEGER PRIMARY KEY AUTOINCREMENT,

                        username TEXT UNIQUE,

                        password TEXT,

                        high\_score INTEGER DEFAULT 0

                      )''')

    connection.commit()

    connection.close()

def register\_user(username, password):

    try:

        connection = sqlite3.connect("snake\_game.db")

        cursor = connection.cursor()

        cursor.execute("INSERT INTO users (username, password) VALUES (?, ?)", (username, password))

        connection.commit()

    except sqlite3.IntegrityError:

        print("Username already exists.")

    finally:

        connection.close()

def validate\_user(username, password):

    connection = sqlite3.connect("snake\_game.db")

    cursor = connection.cursor()

    cursor.execute("SELECT \* FROM users WHERE username = ? AND password = ?", (username, password))

    user = cursor.fetchone()

    connection.close()

    return user is not None

def get\_high\_score(username):

    connection = sqlite3.connect("snake\_game.db")

    cursor = connection.cursor()

    cursor.execute("SELECT high\_score FROM users WHERE username = ?", (username,))

    high\_score = cursor.fetchone()

    connection.close()

    return high\_score[0] if high\_score else 0

def update\_high\_score(username, new\_score):

    connection = sqlite3.connect("snake\_game.db")

    cursor = connection.cursor()

  cursor.execute("UPDATE users SET high\_score = ? WHERE username = ? AND high\_score < ?", (new\_score, username, new\_score))

    connection.commit()

    connection.close()

create\_database()

5.AUTH.PY:

import sqlite3

def create\_connection():

    conn = sqlite3.connect('snake\_game.db')

    return conn

def create\_table():

    conn = create\_connection()

    cursor = conn.cursor()

    cursor.execute('''CREATE TABLE IF NOT EXISTS users (

                        id INTEGER PRIMARY KEY AUTOINCREMENT,

                        username TEXT UNIQUE NOT NULL,

                        password TEXT NOT NULL)''')

    conn.commit()

    conn.close()

def register\_user(username, password):

    conn = create\_connection()

    cursor = conn.cursor()

    try:

        cursor.execute("INSERT INTO users (username, password) VALUES (?, ?)", (username, password))

        conn.commit()

        return True

    except sqlite3.IntegrityError:

        return False

    finally:

        conn.close()

def validate\_user(username, password):

    conn = create\_connection()

    cursor = conn.cursor()

    cursor.execute("SELECT password FROM users WHERE username = ?", (username,))

    stored\_password = cursor.fetchone()

    conn.close()

    if stored\_password:

        return stored\_password[0] == password

    return False

def attempt\_login(username, password):

    return validate\_user(username, password)

def attempt\_register(username, password):

    return register\_user(username, password)

def get\_high\_score():

    conn = create\_connection()

    cursor = conn.cursor()

    cursor.execute("SELECT score FROM highscores ORDER BY score DESC LIMIT 1")

    score = cursor.fetchone()

    conn.close()

    return score[0] if score else 0

def save\_high\_score(username, score):

    conn = create\_connection()

    cursor = conn.cursor()

    cursor.execute("INSERT INTO highscores (username, score) VALUES (?, ?)", (username, score))

    conn.commit()

    conn.close()

create\_table()

OUTPUT:







