Index

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No** | **Aim** | **Pg No** | **Sign** |
| 1 | Write a Python Program to perform a translation operation on Rectangle by taking initial co-ordinates from user | 03 |  |
| 2 | Write a Python Program to perform a Scaling operation on Triangle by taking initial co-ordinates from user | 05 |  |
| 3 | Write a Python Program to perform a Reflection operation on Polygon by taking initial co-ordinates from user | 07 |  |
| 4 | Write a Python Program to rotate right angle triangle by 45 degree by taking initial co-ordinates from user | 10 |  |
| 5 | Write a Python Program to perform a shearing operation on Rectangle in positive direction off x-axis by taking initial co-ordinates from user | 12 |  |
| 6 | Write a Python Program to create below shape and perform reflection about parallel to y-axis, followed by translation and scaling operation on it | 14 |  |
| 7 | Implement space invader game in python using pygame module | 16 |  |
| 8 | Implement Sanke game in python using pygame module | 21 |  |
| 9 | Implement 2D UFO Game using unity | 28 |  |
| 10 | Implement 3D Roll Ball Game using unity | 32 |  |

Practical 1:

Write a Python Program to perform a translation operation on Rectangle by taking initial co-ordinates from user

Code:

from tkinter import \*

root = Tk()

C = Canvas(root,bg="cyan",height=700,width=700)

C.create\_text(110,40,text="figure before Translation",fill="black",font=("Arial"))

x1 = int(input("Enter x1 value: "))

y1 = int(input("Enter y1 value: "))

x2 = int(input("Enter x2 value: "))

y2 = int(input("Enter y2 value: "))

x3 = int(input("Enter x3 value: "))

y3 = int(input("Enter y3 value: "))

x4 = int(input("Enter x4 value: "))

y4 = int(input("Enter y4 value: "))

C.create\_polygon(x1,y1,x2,y2,x3,y3,x4,y4,fill="red")

tx=250

ty=250

C.create\_text(405,340,text="figure after Translation",fill="black",font=("Arial"))

C.create\_polygon(x1+tx,y1+ty,x2+tx,y2+ty,x3+tx,y3+ty,x4+tx,y4+ty,fill="red")

C.pack()

mainloop()

Practical 2:

Write a Python Program to perform a Scaling operation on Triangle by taking initial co-ordinates from user

Code:

from tkinter import \*

root = Tk()

C = Canvas(root,bg="cyan",height=700,width=700)

C.create\_text(160,90,text="figure before Scaling",fill="black",font=("Arial"))

x1 = int(input("Enter x1 value: "))

y1 = int(input("Enter y1 value: "))

x2 = int(input("Enter x2 value: "))

y2 = int(input("Enter y2 value: "))

x3 = int(input("Enter x3 value: "))

y3 = int(input("Enter y3 value: "))

C.create\_polygon(x1,y1,x2,y2,x3,y3,fill="red")

sx=3

sy=3

C.create\_text(380,300,text="figure after Scaling",fill="black",font=("Arial"))

C.create\_polygon(x1\*sx,y1\*sy,x2\*sx,y2\*sy,x3\*sx,y3\*sy,fill="red")

C.pack()

mainloop()

Practical 3:

Write a Python Program to perform a Reflection operation on Polygon by taking initial co-ordinates from user

from tkinter import \*

import math

root = Tk()

C = Canvas(root,bg="aqua",height=500,width=700)

C.create\_text(200,30,text="Original Triangle",fill="green")

x1 = int(input("Enter x1 value: "))

y1 = int(input("Enter y1 value: "))

x2 = int(input("Enter x2 value: "))

y2 = int(input("Enter y2 value: "))

x3 = int(input("Enter x3 value: "))

y3 = int(input("Enter y3 value: "))

C.create\_polygon(x1,y1,x2,y2,x3,y3,fill="blue")

a = int(input("Enter arbitary axis: "))

x11 =(-x1 + (2\*a))

x21 =(-x2 + (2\*a))

x31 =(-x3 + (2\*a))

C.create\_polygon(x11,y1,x21,y2,x31,y3,fill="blue")

a = int(input("Enter arbitary axis: "))

y11 =(-y1 + (2\*a))

y21 =(-y2 + (2\*a))

y31 =(-y3 + (2\*a))

C.create\_polygon(x1,y11,x2,y21,x3,y31,fill="blue")

C.pack()

root.mainloop()

Practical 4:

Write a Python Program to rotate right angle triangle by 45 degree by taking initial co-ordinates from user

Code:

import math

from tkinter import \*

root=Tk();

C=Canvas(root,bg="gray",height=1000,width=1000)

C.create\_text(150,40,text="figure Before Rotation",fill="black",font=('Helvicta 15 bold'))

x1 = int(input("Enter x1 : "))

y1 = int(input("Enter y1 : "))

x2 = int(input("Enter x2 : "))

y2 = int(input("Enter y2 : "))

x3 = int(input("Enter x3 : "))

y3 = int(input("Enter y3 : "))

C.create\_polygon(x1,y1,x2,y2,x3,y3,fill="red")

x11=abs((x1\*math.cos(45))-(y1\*math.sin(45)))

y12=abs((y1\*math.sin(45))+(x1\*math.cos(45)))

x21=abs((x2\*math.cos(45))-(y2\*math.sin(45)))

y22=abs((y2\*math.sin(45))+(x2\*math.cos(45)))

x31=abs((x3\*math.cos(45))-(y3\*math.sin(45)))

y32=abs((y3\*math.sin(45))+(x3\*math.cos(45)))

C.create\_polygon(x11,y12,x21,y22,x31,y32,fill="blue")

C.pack();

mainloop()

Practical 5:

Write a Python Program to perform a shearing operation on Rectangle in positive direction off x-axis by taking initial co-ordinates from user

Code:

from tkinter import \*

import math

root = Tk()

C = Canvas(root, bg="aqua", height=500, width=700)

C.create\_text(100,40,text="Figure after shearing",fill="black")

x1 = int(input("Enter x1 value: "))

y1 = int(input("Enter y1 value: "))

x2 = int(input("Enter x2 value: "))

y2 = int(input("Enter y2 value: "))

x3 = int(input("Enter x3 value: "))

y3 = int(input("Enter y3 value: "))

x4 = int(input("Enter x4 value: "))

y4 = int(input("Enter y4 value: "))

C.create\_polygon(x1, y1, x2, y2, x3, y3,x4,y4, fill="blue")

t=math.radians(45)

x11=x1+(y1\*math.tan(t))

x21=x2+(y2\*math.tan(t))

C.create\_polygon(x11, y1, x21, y2, x3, y3,x4,y4, fill="red")

C.pack()

root.mainloop()

Practical 6:

Write a Python Program to create below shape and perform reflection about parallel to y-axis, followed by translation and scaling operation on it

Code:

from tkinter import \*

root = Tk()

c = Canvas(root,bg="grey",height=500,width=1250)

c.create\_text(150,80,text='Before')

x1 = 150

x2 = 150

x3 = 300

x4 = 300

x5 = 150

x6 = 150

x7 = 100

y1 = 100

y2 = 130

y3 = 130

y4 = 170

y5 = 170

y6 = 200

y7 = 150

c.create\_polygon(x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6,x7,y7,fill='Light blue')

a = 350

x1 = 2\*a - x1

x2 = 2\*a - x2

x3 = 2\*a - x3

x4 = 2\*a - x4

x5 = 2\*a - x5

x6 = 2\*a - x6

x7 = 2\*a - x7

c.create\_text(450,80,text='After Reflection')

c.create\_polygon(x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6,x7,y7,fill='Red')

tx=150

ty=80

c.create\_text(650,80,text='After Translation')

c.create\_polygon(x1+tx,y1+ty,x2+tx,y2+ty,x3+tx,y3+ty,x4+tx,y4+ty,x5+tx,y5+ty,x6+tx,y6+ty,x7+tx,y7+ty,fill='Green')

sx=2

sy=2

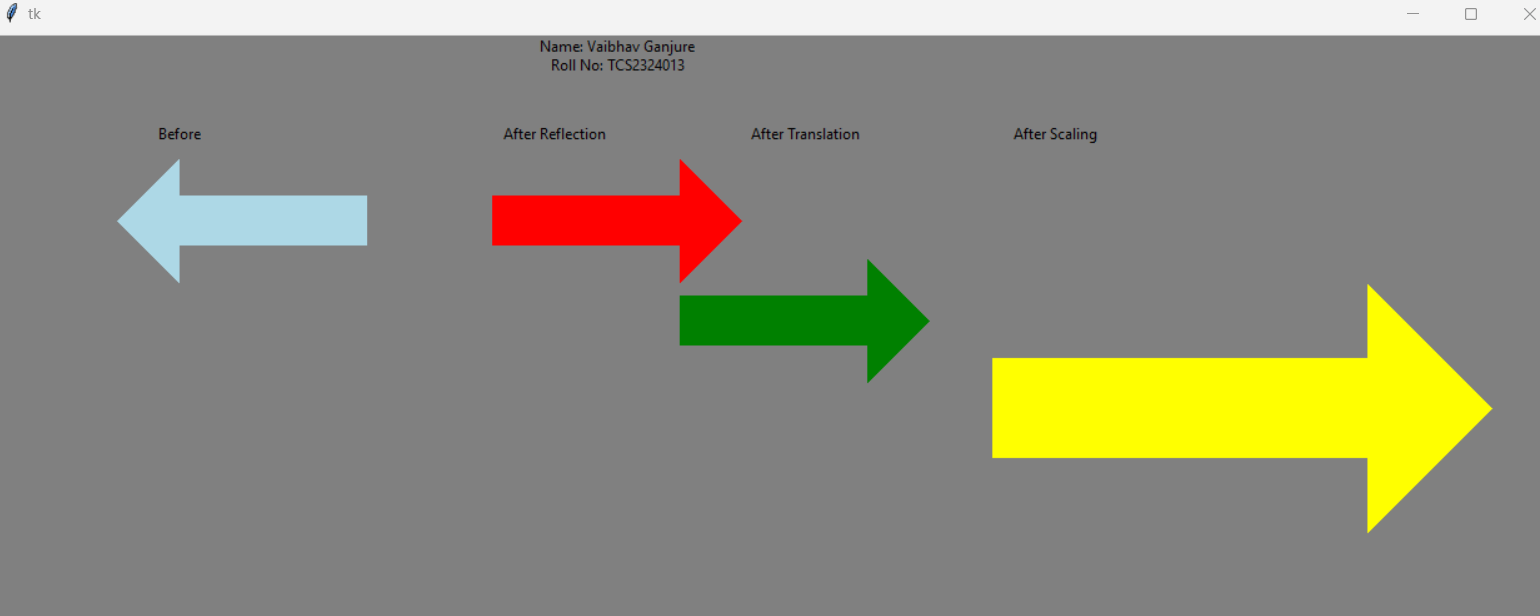
c.create\_text(850,80,text='After Scaling')

c.create\_polygon(x1\*sx, y1\*sy, x2\*sx, y2\*sy,x3\*sx,y3\*sy,x4\*sx,y4\*sy,x5\*sx,y5\*sy,x6\*sx,y6\*sy,x7\*sx,y7\*sy,fill='yellow')

c.pack()

mainloop()

Output:



Practical 7:

Implement space invader game in python using pygame module

Code:

import pygame

import random

import math

from pygame import mixer

pygame.init()

screen = pygame.display.set\_mode((800, 600))

pygame.display.set\_caption("SpaceInvader")

# icon set

icon1 = pygame.image.load("enemy.png")

pygame.display.set\_icon(icon1)

# backgroundSound

mixer.music.load("background.wav")

mixer.music.play(-1)

# player img set

playerimg = pygame.image.load("player.png")

playerX = 370

playerY = 480

playerX\_change = 0

# creating bullet

# ready : You cant see the bullet

# fire : The bullet is currently moving

bulletImg = pygame.image.load('bullet.png')

bulletX = 0

bulletY = 488

bulletX\_change = 0

bulletY\_change = 10

bullet\_state = "ready"

score\_value = 0

# addingfont for displayingcode

font = pygame.font.Font("freesansbold.ttf", 32)

textX = 10

textY = 10

over\_font = pygame.font.Font("freesansbold.ttf", 70)

def game\_over\_text():

score = over\_font.render("Game Over", True, (255, 255, 255))

screen.blit(score, (200, 250))

def show\_score(x, y):

score = font.render("Score: " + str(score\_value), True, (255, 255, 255))

screen.blit(score, (x, y))

# Enemy

enemyimg = []

enemyX = []

enemyY = []

enemyX\_change = []

enemyY\_change = []

no\_of\_enemies = 6

# enemy

for i in range(no\_of\_enemies):

enemyimg.append(pygame.image.load("enemy.png"))

enemyX.append(random.randint(0, 735))

enemyY.append(random.randint(50, 150))

enemyX\_change.append(3)

enemyY\_change.append(40)

# Background Img

background = pygame.image.load("background.png")

# collision function

def isCollision(enemyX, enemyY, bulletX, bulletY):

distance = math.sqrt((math.pow(enemyX - bulletX, 2)) + (math.pow(enemyY - bulletY, 2)))

if distance < 27:

return True

else:

return False

def fire\_bullet(x, y):

global bullet\_state

bullet\_state = "fire"

screen.blit(bulletImg, (x + 16, y + 10))

def player(x, y):

screen.blit(playerimg, (x, y))

def enemy(x, y, i):

screen.blit(enemyimg[i], (x, y))

running = True

# Game Window

while running:

screen.fill((0, 45, 49))

screen.blit(background, (0, 0))

# playerX-=0.1

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

if event.type == pygame.KEYDOWN:

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

playerX\_change = -3

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

playerX\_change = 3

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

if bullet\_state is "ready":

bullet\_sound = mixer.Sound("laser.wav")

bullet\_sound.play()

bulletX = playerX

fire\_bullet(bulletX, bulletY)

if event.type == pygame.KEYUP:

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

playerX\_change = 0

# bullet movement

if bulletY <= 0:

bulletY = 480

bullet\_state = "ready"

if bullet\_state is "fire":

fire\_bullet(bulletX, bulletY)

bulletY -= bulletY\_change

playerX += playerX\_change

# enemy movement

for i in range(no\_of\_enemies):

if enemyY[i] > 480:

for j in range(no\_of\_enemies):

enemyY[i] = 2000

game\_over\_text()

break

enemyX[i] += enemyX\_change[i]

if enemyX[i] <= 0:

enemyX\_change[i] = 0.5

enemyY[i] += enemyY\_change[i]

elif enemyX[i] >= 735:

enemyX\_change[i] = -0.5

enemyY[i] += enemyY\_change[i]

collision = isCollision(enemyX[i], enemyY[i], bulletX, bulletY)

if collision:

bulletY = 480

bullet\_state = "ready"

score\_value += 1

print(score\_value)

enemyX[i] = random.randint(0, 735)

enemyY[i] = random.randint(50, 150)

enemy(enemyX[i], enemyY[i], i)

# Setting the boundary

if playerX <= 0:

playerX = 0

elif playerX >= 736:

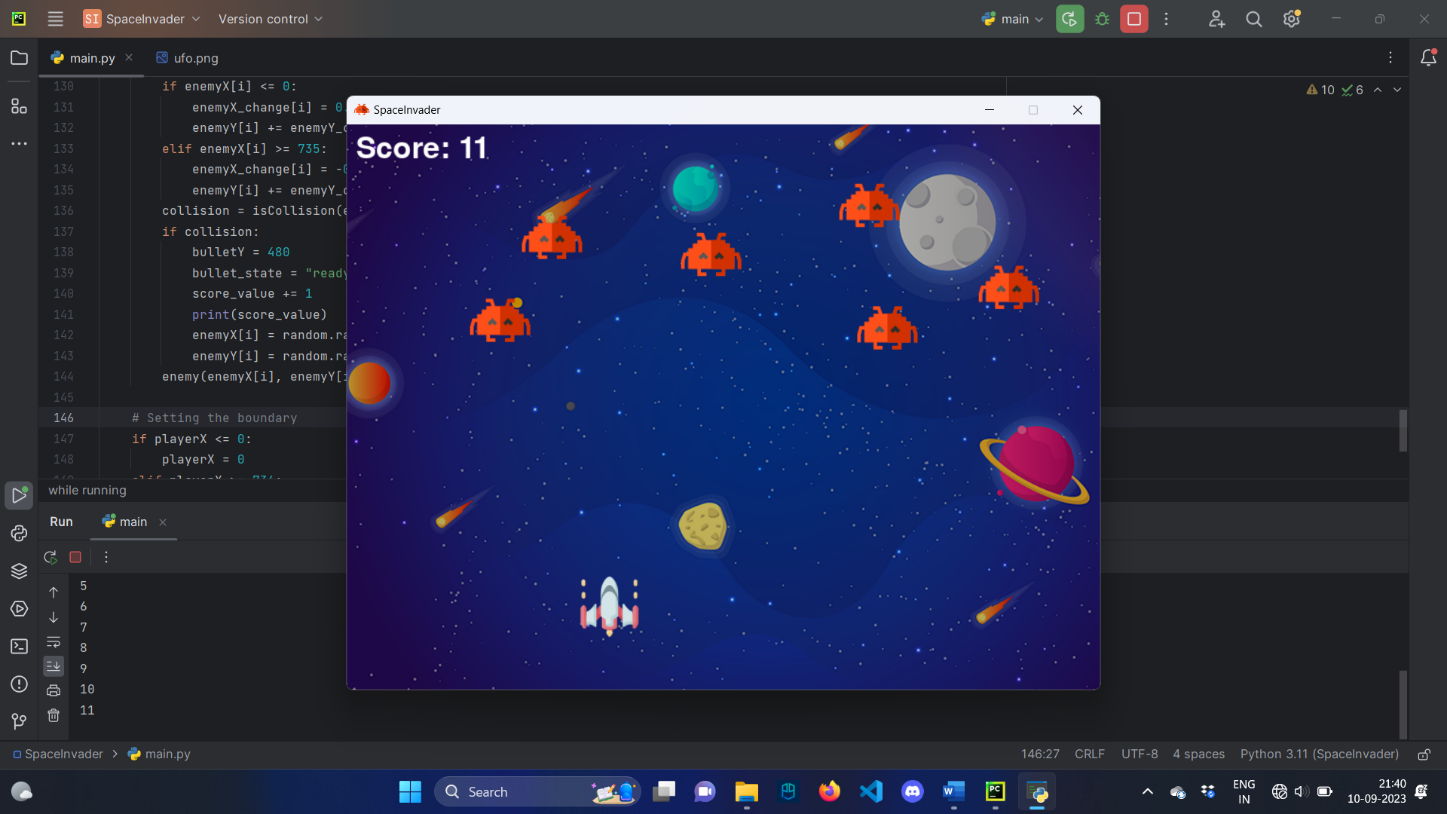
playerX = 736

player(playerX, playerY)

show\_score(textX, textY)

pygame.display.update()

Output:



Practical 8:

Implement Snake game in python using pygame module

Code:

import pygame

import random

import math

pygame.init()

screen = pygame.display.set\_mode((900, 700))

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

icon = pygame.image.load("SnakeHead.png")

pygame.display.set\_icon(icon)

playerimg = pygame.image.load("SnakeHead.png")

length = 1

playerX = [434] \* length

playerY = [350] \* length

playerX\_change = 0

playerY\_change = 0

initial = "down"

current\_direction = "down"

opposite\_direction = {"left": "right", "right": "left", "up": "down", "down": "up"}

auto\_move\_timer = pygame.time.get\_ticks()

auto\_move\_interval = 0.5

def player(x, y, p, length, direction):

bodyX = [x] \* length

bodyY = [y] \* length

for i in range(1, length):

if direction == 'left':

p = pygame.transform.rotate(playerimg, -90)

bodyX[i] = bodyX[i - 1] + 36

bodyY[i] = bodyY[i - 1]

pygame.draw.circle(screen, (42, 72, 8), (bodyX[i] + 40, bodyY[i] + 30), 18)

elif direction == 'right':

p = pygame.transform.rotate(playerimg, 90)

bodyX[i] = bodyX[i - 1] - 36

bodyY[i] = bodyY[i - 1]

pygame.draw.circle(screen, (42, 72, 8), (bodyX[i] +20 , bodyY[i] + 30), 18)

elif direction == 'up':

p = pygame.transform.rotate(playerimg, 180)

bodyX[i] = bodyX[i - 1]

bodyY[i] = bodyY[i - 1] + 36

pygame.draw.circle(screen, (42, 72, 8), (bodyX[i] + 32, bodyY[i] + 42), 18)

elif direction == 'down':

p = playerimg

bodyX[i] = bodyX[i - 1]

bodyY[i] = bodyY[i - 1] - 36

pygame.draw.circle(screen, (42, 72, 8), (bodyX[i] + 30, bodyY[i] + 20), 18)

screen.blit(p, (x, y))

eggimg = []

eggX = []

eggY = []

no\_of\_egg = 2

for i in range(no\_of\_egg):

eggimg.append(pygame.image.load("egg3.png"))

eggX.append(random.randint(0, 865))

eggY.append(random.randint(0, 635))

def egg(x, y, i):

screen.blit(eggimg[i], (x, y))

def tail():

playerX.append(-1)

playerY.append(-1)

gameover = pygame.font.Font("freesansbold.ttf", 64)

def game\_over():

for i in range(0, 2):

eggX[i] = 1000

eggY[i] = 1000

over = gameover.render("GAME OVER", True, (255, 255, 255))

screen.blit(over, (300, 300))

font = pygame.font.Font("freesansbold.ttf", 32)

score\_value = 0

def show\_score():

score = font.render("Score: " + str(score\_value), True, (255, 255, 255))

screen.blit(score, (10, 10))

def isCollision(eggX, eggY, playerX, playerY):

distance = math.sqrt((math.pow(eggX - playerX, 2)) + (math.pow(eggY - playerY, 2)))

return distance

running = True

game\_active = True

while running:

screen.fill((0, 0, 0))

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

if event.type == pygame.KEYDOWN:

if game\_active

if event.key == pygame.K\_LEFT and current\_direction != "right":

playerX\_change = -30

playerY\_change = 0

initial = 'left'

current\_direction = 'left'

elif event.key == pygame.K\_RIGHT and current\_direction != "left":

playerX\_change = 30

playerY\_change = 0

current\_direction = 'right'

elif event.key == pygame.K\_UP and current\_direction != "down":

playerY\_change = -30

playerX\_change = 0

current\_direction = 'up'

elif event.key == pygame.K\_DOWN and current\_direction != "up":

playerY\_change = 30

playerX\_change = 0

current\_direction = 'down'

elif (

(event.key == pygame.K\_LEFT and current\_direction == "right")

or (event.key == pygame.K\_RIGHT and current\_direction == "left")

or (event.key == pygame.K\_UP and current\_direction == "down")

or (event.key == pygame.K\_DOWN and current\_direction == "up")

):

game\_active = False

if game\_active:

current\_time = pygame.time.get\_ticks()

if current\_time - auto\_move\_timer > auto\_move\_interval:

if current\_direction == "left":

playerX\_change = -0.5

playerY\_change = 0

elif current\_direction == "right":

playerX\_change = 0.5

playerY\_change = 0

elif current\_direction == "up":

playerY\_change = -0.5

playerX\_change = 0

elif current\_direction == "down":

playerY\_change = 0.5

playerX\_change = 0

auto\_move\_timer = current\_time

for i in range(length - 1, 0, -1):

playerX[i] = playerX[i - 1]

playerY[i] = playerY[i - 1]

playerX[0] += playerX\_change

playerY[0] += playerY\_change

if playerX[0] <= 0 or playerX[0] >= 847 or playerY[0] <= 0 or playerY[0] >= 640:

game\_active = False

for i in range(30, length):

dcol = isCollision(playerX[0], playerY[0], playerX[i], playerY[i])

if dcol < 1:

game\_active = False

for i in range(no\_of\_egg):

egg(eggX[i], eggY[i], i)

collision = isCollision(eggX[i], eggY[i], playerX[0], playerY[0])

if collision < 27:

length += 1

tail()

score\_value += 1

eggX[i] = random.randint(0, 865)

eggY[i] = random.randint(0, 635)

player(playerX[0], playerY[0], playerimg, length, current\_direction)

show\_score()

if not game\_active:

player(playerX[0], playerY[0], playerimg, length, current\_direction)

game\_over\_text = gameover.render("GAME OVER", True, (255, 255, 255))

screen.blit(game\_over\_text, (300, 300))

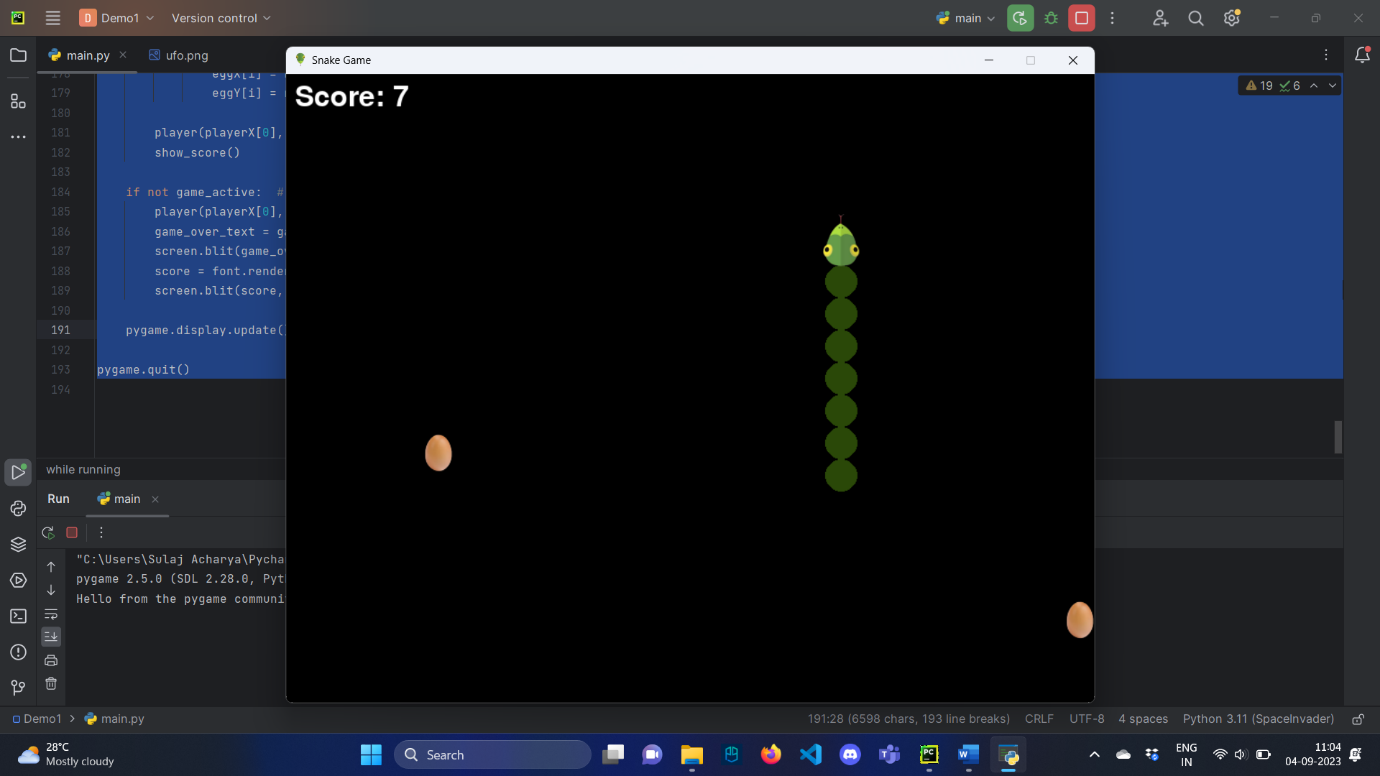
score = font.render("Score: " + str(score\_value), True, (255, 255, 255))

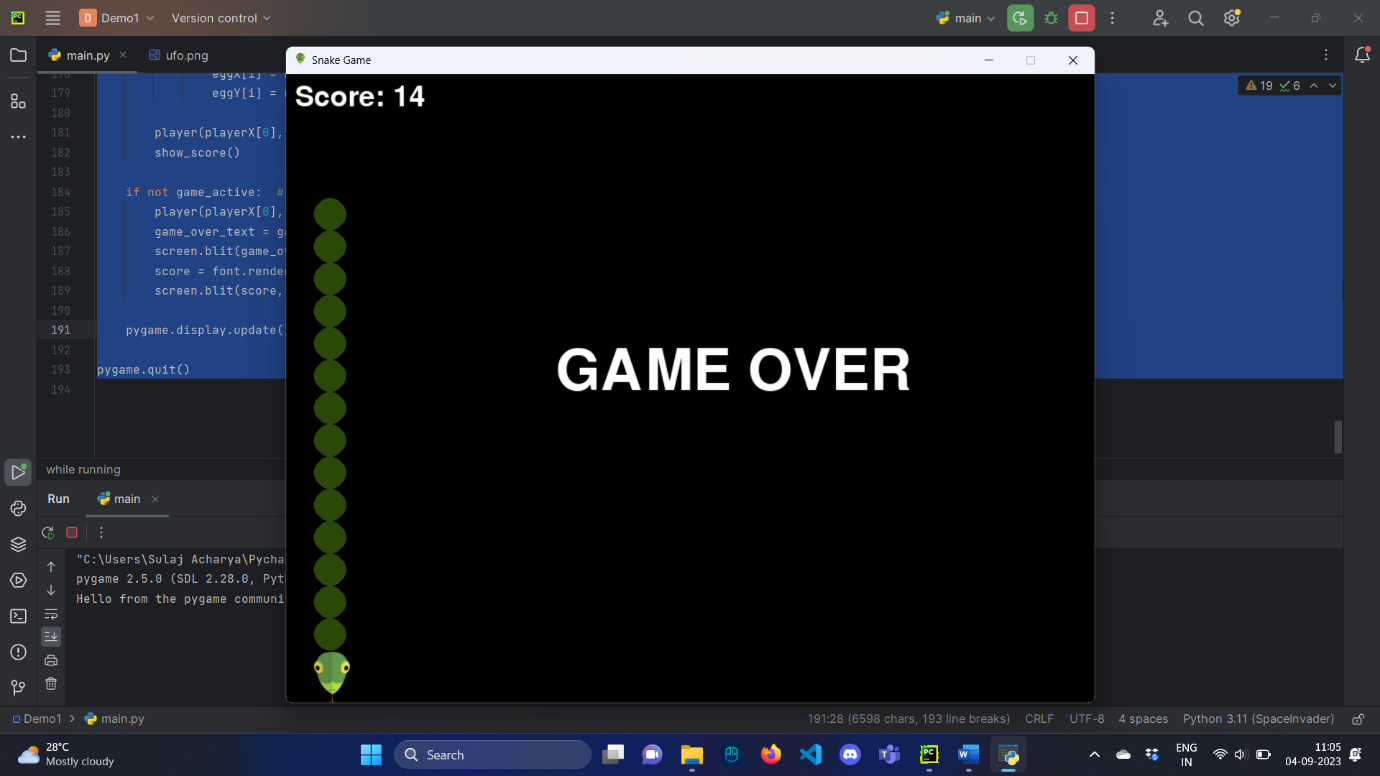
screen.blit(score, (10, 10))

pygame.display.update()

pygame.quit()

Output:





Practical 9:

Implement 2D UFO Game using unity

.PlayerController Script:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class playercontroller : MonoBehaviour

{

private Rigidbody2D rbd;

public float speed;

public Text winText;

public Text countText;

public int count = 0;

// Start is called before the first frame update

void Start()

{

rbd = GetComponent<Rigidbody2D>();

}

// Update is called once per frame

void FixedUpdate()

{

float moveHorizontal = Input.GetAxis("Horizontal");

float moveVertical = Input.GetAxis("Vertical");

Vector2 movement = new Vector2(moveHorizontal, moveVertical);

rbd.AddForce(movement\*speed);

}

void OnTriggerEnter2D(Collider2D other)

{

if (other.tag == "PickUp")

{

other.gameObject.SetActive(false);

count++;

SetCountText();

}

}

void SetCountText()

{

countText.text = "Count" + count.ToString();

if (count == 7)

{

winText.text = "you won";

}

}

}

.CameraController Script:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class cameracontroller : MonoBehaviour

{

public GameObject player;

public Vector3 offset;

// Start is called before the first frame update

void Start()

{

offset = transform.position - player.transform.position;

}

// Update is called once per frame

void LateUpdate()

{

transform.position = player.transform.position + offset;

}

}

Rotator Script:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class rotator : MonoBehaviour

{

// Start is called before the first frame update

void Start()

{

}

// Update is called once per frame

void Update()

{

transform.Rotate(new Vector3(0, 0, 45) \* Time.deltaTime);

}

}

A screenshot of a computer

Description automatically generated

Output:

Practical no 10:

Implement 3D Roll Ball Game using Unity

PlayerController Script:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class controller : MonoBehaviour

{

private Rigidbody rb;

public float speed;

public Text winText;

public Text countText;

public int count;

// Start is called before the first frame update

void Start()

{

count = 0;

winText.text = "";

rb = GetComponent<Rigidbody>();

SetCountText();

}

// Update is called once per frame

void FixedUpdate()

{

float moveHorizontal = Input.GetAxis("Horizontal");

float moveVertical = Input.GetAxis("Vertical");

Vector3 movement=new Vector3(moveHorizontal,0.0f,moveVertical);

rb.AddForce(movement \* speed);

}

void OnTriggerEnter(Collider other)

{

if (other.gameObject.CompareTag("pickup"))

{

other.gameObject.SetActive(false);

count = count + 1;

SetCountText();

}

}

void SetCountText()

{

countText.text = "Score: " + count.ToString();

if (count >= 6){

winText.text = " You Win";

}

}

}

CameraController Script:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CameraController : MonoBehaviour

{

private Vector3 offset;

public GameObject player;

// Start is called before the first frame update

void Start()

{

offset = transform.position - player.transform.position;

}

// Update is called once per frame

void LateUpdate()

{

transform.position = player.transform.position + offset;

}

}

Rotator Script:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Rotator : MonoBehaviour

{

// Start is called before the first frame update

void Start()

{

}

// Update is called once per frame

void Update()

{

transform.Rotate(new Vector3(15, 30, 45) \* Time.deltaTime);

}

}

Output:

