|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Description** | **Page No.** |
| **PART – A** | | |
| 1 | Write a Python program to initialize Pygame and create a window. |  |
| 2 | Write a Python program to change the screen background color in Pygame. |  |
| 3 | Write a Python program to set and change the window size and title in Pygame. |  |
| 4 | Write a Python program to handle keyboard and mouse events in Pygame. |  |
| 5 | Write a Python program to draw basic shapes (lines, rectangles, circles) in Pygame. |  |
| 6 | Write a Python program to load and display an image using Pygame. |  |
| 7 | Write a Python program to load and customize the cursor in Pygame. |  |
| 8 | Write a Python program to move an image using numeric keypads and the mouse in Pygame. |  |
| **PART – B** | | |
| 1 | Write a Python program to use text as buttons with event handling and display image in the same window after clicking the button in Pygame. |  |
| 2 | Write a program for a Brick Breaker game in Python using Pygame. |  |
| 3 | Write a Python program to load an image on a surface and perform transformations in Pygame. |  |
| 4 | Write a Python program to integrate PyOpenGL for 3D rendering and transformations in Pygame. |  |
| 5 | Write a Python program to develop pong game in Pygame. |  |
| 6 | Write a Python program to develop flappy game in Pygame. |  |
| 7 | Write a Python program to develop tic tac toe game in Pygame. |  |
| 8 | Write a Python program to develop a snake game in Pygame. |  |

**INDEX**

**PART – A**

**1. Write a Python program to initialize Pygame and create a window.**

import pygame

# Initialize Pygame

pygame.init()

# Set up the display

screen = pygame.display.set\_mode((800,640))

pygame.display.set\_caption("My Game Window")

# Main loop

running = True

while running:

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

pygame.display.flip()

# Quit Pygame

pygame.quit()

**2. Write a Python program to change the screen background color in Pygame.**

# Importing the library

import pygame

import sys

import random

# Initializing Pygame

pygame.init()

# Initializing surface

surface = pygame.display.set\_mode((400, 300))

pygame.display.set\_caption('Change Background Color')

# Function to generate a random color

def get\_random\_color():

return (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))

# Function to draw the button

def draw\_button(surface, rect, color, text):

pygame.draw.rect(surface, color, rect)

font = pygame.font.SysFont(None, 24)

text\_surface = font.render(text, True, (0, 0, 0))

text\_rect = text\_surface.get\_rect(center=rect.center)

surface.blit(text\_surface, text\_rect)

# Initializing RGB Color

color = get\_random\_color()

button\_color = (200, 200, 200)

button\_rect = pygame.Rect(150, 130, 100, 40)

# Main loop to keep the window open

running = True

while running:

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

elif event.type == pygame.MOUSEBUTTONDOWN:

if button\_rect.collidepoint(event.pos):

color = get\_random\_color()

# Changing surface color

surface.fill(color)

draw\_button(surface, button\_rect, button\_color, 'Change Color')

pygame.display.flip()

# Quit Pygame

pygame.quit()

sys.exit()

**3. Write a Python program to set and change the window size and title in Pygame.**

# Import the Pygame library

import pygame

# Initialize Pygame

pygame.init()

# Set the initial window size

window\_size = (800, 600)

screen = pygame.display.set\_mode(window\_size)

# Set the initial window title

pygame.display.set\_caption("Initial Title")

# Function to change the window size

def change\_window\_size(new\_size):

global screen

screen = pygame.display.set\_mode(new\_size)

pygame.display.flip()

# Function to change the window title

def change\_window\_title(new\_title):

pygame.display.set\_caption(new\_title)

# Main loop

running = True

while running:

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

elif event.type == pygame.KEYDOWN:

if event.key == pygame.K\_1:

change\_window\_size((640, 480))

elif event.key == pygame.K\_2:

change\_window\_size((1024, 768))

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

change\_window\_title("New Title")

# Fill the screen with a color

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

pygame.display.flip()

# Quit Pygame

pygame.quit()

**4. Write a Python program to handle keyboard and mouse events in Pygame.**

# Import the Pygame library

import pygame

# Initialize Pygame

pygame.init()

# Set up display

window\_size = (800, 600)

screen = pygame.display.set\_mode(window\_size)

pygame.display.set\_caption("Event Handling in Pygame")

# Main loop

running = True

while running:

# Handle events

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

elif event.type == pygame.KEYDOWN:

print(f"Key {pygame.key.name(event.key)} pressed")

elif event.type == pygame.KEYUP:

print(f"Key {pygame.key.name(event.key)} released")

elif event.type == pygame.MOUSEBUTTONDOWN:

if event.button == 1: # Left mouse button

print(f"Left mouse button clicked at {event.pos}")

elif event.button == 2: # Middle mouse button

print(f"Middle mouse button clicked at {event.pos}")

elif event.button == 3: # Right mouse button

print(f"Right mouse button clicked at {event.pos}")

elif event.type == pygame.MOUSEBUTTONUP:

print(f"Mouse button released at {event.pos}")

elif event.type == pygame.MOUSEMOTION:

print(f"Mouse moved to {event.pos}")

# Fill the screen with a color

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

pygame.display.flip()

# Quit Pygame

pygame.quit()

**5. Write a Python program to draw basic shapes (lines, rectangles, circles) in Pygame.**

# Import the Pygame library

import pygame

# Initialize Pygame

pygame.init()

# Set up display

window\_size = (800, 600)

screen = pygame.display.set\_mode(window\_size)

pygame.display.set\_caption("Drawing Shapes in Pygame")

# Colors

WHITE = (255, 255, 255)

BLACK = (0, 0, 0)

RED = (255, 0, 0)

GREEN = (0, 255, 0)

BLUE = (0, 0, 255)

# Main loop

running = True

while running:

# Handle events

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

# Fill the screen with white

screen.fill(WHITE)

# Draw a red line

pygame.draw.line(screen, RED, (100, 100), (700, 100), 5)

# Draw a green rectangle

pygame.draw.rect(screen, GREEN, (150, 200, 500, 100))

# Draw a blue circle

pygame.draw.circle(screen, BLUE, (400, 400), 75)

# Update the display

pygame.display.flip()

# Quit Pygame

pygame.quit()

**6. Write a Python program to load and display an image using Pygame.**

# Import the Pygame library

import pygame

# Initialize Pygame

pygame.init()

# Set up display

window\_size = (800, 600)

screen = pygame.display.set\_mode(window\_size)

pygame.display.set\_caption("Display Image in Pygame")

# Load an image

image\_path = 'Pygame\_logo.gif' # Replace with the path to your image file

image = pygame.image.load(image\_path)

# Main loop

running = True

while running:

# Handle events

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

# Fill the screen with white

screen.fill((255, 255, 255))

# Display the image

screen.blit(image, (100, 100)) # Draw the image at (100, 100)

# Update the display

pygame.display.flip()

# Quit Pygame

pygame.quit()

**7. Write a Python program to load and customize the cursor in Pygame.**

import pygame

import os

# Initialize Pygame

pygame.init()

# Screen settings

WIDTH, HEIGHT = 800, 600

screen = pygame.display.set\_mode((WIDTH, HEIGHT))

pygame.display.set\_caption('Custom Cursor Example')

# Load custom cursor image

cursor\_image = pygame.image.load('Pygame\_logo.gif')

cursor\_image = pygame.transform.scale(cursor\_image, (32, 32)) # Scale cursor image to desired size

cursor\_rect = cursor\_image.get\_rect()

# Hide the default cursor

pygame.mouse.set\_visible(False)

# Main loop

running = True

while running:

screen.fill((255, 255, 255)) # Fill the screen with white

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

# Get mouse position and update cursor rect position

cursor\_rect.center = pygame.mouse.get\_pos()

# Draw custom cursor

screen.blit(cursor\_image, cursor\_rect.topleft)

# Update the display

pygame.display.flip()

pygame.quit()

**8. Write a Python program to move an image using numeric keypads and the mouse in Pygame.**

import pygame

# Initialize Pygame

pygame.init()

# Set up display

window\_size = (800, 600)

screen = pygame.display.set\_mode(window\_size)

pygame.display.set\_caption("Move Image with Keypad and Mouse")

# Load image

image = pygame.image.load('Pygame\_logo.gif')

image\_rect = image.get\_rect()

image\_rect.topleft = (100, 100) # Initial position

# Main loop

running = True

while running:

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

elif event.type == pygame.MOUSEBUTTONDOWN:

if event.button == 1: # Left mouse button

image\_rect.center = event.pos

# Get keys pressed

keys = pygame.key.get\_pressed()

# Move image with numeric keypad

if keys[pygame.K\_KP4]: # Keypad 4 - Move left

image\_rect.x -= 5

if keys[pygame.K\_KP6]: # Keypad 6 - Move right

image\_rect.x += 5

if keys[pygame.K\_KP8]: # Keypad 8 - Move up

image\_rect.y -= 5

if keys[pygame.K\_KP2]: # Keypad 2 - Move down

image\_rect.y += 5

# Fill the screen with white

screen.fill((255, 255, 255))

# Draw the image

screen.blit(image, image\_rect)

# Update the display

pygame.display.flip()

# Quit Pygame

pygame.quit()

**PART – B**

**1. Write a Python program to use text as buttons with event handling and display image in the same window after clicking the button in Pygame.**

import pygame

import sys

# Initialize Pygame

pygame.init()

# Set window size and title

screen\_width, screen\_height = 800, 600

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

pygame.display.set\_caption("Text Button Event Handling")

# Define colors

WHITE = (255, 255, 255)

BLACK = (0, 0, 0)

GRAY = (200, 200, 200)

# Load image

image = pygame.image.load('Pygame\_logo.gif') # Replace 'your\_image.png' with the path to your image

image\_rect = image.get\_rect(center=(screen\_width // 2, screen\_height // 2))

# Define font

font = pygame.font.SysFont(None, 40)

# Button class

class Button:

def \_\_init\_\_(self, text, x, y, width, height):

self.text = text

self.rect = pygame.Rect(x, y, width, height)

self.color = GRAY

def draw(self, screen):

pygame.draw.rect(screen, self.color, self.rect)

text\_surf = font.render(self.text, True, BLACK)

text\_rect = text\_surf.get\_rect(center=self.rect.center)

screen.blit(text\_surf, text\_rect)

def is\_hovered(self, mouse\_pos):

return self.rect.collidepoint(mouse\_pos)

# Create buttons

button1 = Button('Show Image', 100, 50, 200, 50)

# Main loop

running = True

show\_image = False

while running:

screen.fill(WHITE)

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

elif event.type == pygame.MOUSEBUTTONDOWN:

if button1.is\_hovered(event.pos):

show\_image = True

print("Button clicked: Show Image") # Print statement for event handling

# Draw buttons

button1.draw(screen)

# Display image if button is clicked

if show\_image:

screen.blit(image, image\_rect)

pygame.display.flip()

# Quit Pygame

pygame.quit()

sys.exit()

**2. Write a program for a Brick Breaker game in Python using Pygame.**

import pygame

import random

pygame.init()

# Dimensions of the screen

WIDTH, HEIGHT = 600, 500

# Colors

BLACK = (0, 0, 0)

WHITE = (255, 255, 255)

GREEN = (0, 255, 0)

RED = (255, 0, 0)

font = pygame.font.Font('freesansbold.ttf', 15)

screen = pygame.display.set\_mode((WIDTH, HEIGHT))

pygame.display.set\_caption("Block Breaker")

# to control the frame rate

clock = pygame.time.Clock()

FPS = 30

# Striker class

class Striker:

def \_\_init\_\_(self, posx, posy, width, height, speed, color):

self.posx, self.posy = posx, posy

self.width, self.height = width, height

self.speed = speed

self.color = color

# The rect variable is used to handle the placement

# and the collisions of the object

self.strikerRect = pygame.Rect(

self.posx, self.posy, self.width, self.height)

self.striker = pygame.draw.rect(screen,

self.color, self.strikerRect)

# Used to render the object on the screen

def display(self):

self.striker = pygame.draw.rect(screen,

self.color, self.strikerRect)

# Used to update the state of the object

def update(self, xFac):

self.posx += self.speed\*xFac

# Restricting the striker to be in between the

# left and right edges of the screen

if self.posx <= 0:

self.posx = 0

elif self.posx+self.width >= WIDTH:

self.posx = WIDTH-self.width

self.strikerRect = pygame.Rect(

self.posx, self.posy, self.width, self.height)

# Returns the rect of the object

def getRect(self):

return self.strikerRect

# Block Class

class Block:

def \_\_init\_\_(self, posx, posy, width, height, color):

self.posx, self.posy = posx, posy

self.width, self.height = width, height

self.color = color

self.damage = 100

# The white blocks have the health of 200. So,

# the ball must hit it twice to break

if color == WHITE:

self.health = 200

else:

self.health = 100

# The rect variable is used to handle the placement

# and the collisions of the object

self.blockRect = pygame.Rect(

self.posx, self.posy, self.width, self.height)

self.block = pygame.draw.rect(screen, self.color,

self.blockRect)

# Used to render the object on the screen if and only

# if its health is greater than 0

def display(self):

if self.health > 0:

self.brick = pygame.draw.rect(screen,

self.color, self.blockRect)

# Used to decrease the health of the block

def hit(self):

self.health -= self.damage

# Used to get the rect of the object

def getRect(self):

return self.blockRect

# Used to get the health of the object

def getHealth(self):

return self.health

# Ball Class

class Ball:

def \_\_init\_\_(self, posx, posy, radius, speed, color):

self.posx, self.posy = posx, posy

self.radius = radius

self.speed = speed

self.color = color

self.xFac, self.yFac = 1, 1

self.ball = pygame.draw.circle(

screen, self.color, (self.posx,

self.posy), self.radius)

# Used to display the object on the screen

def display(self):

self.ball = pygame.draw.circle(

screen, self.color, (self.posx,

self.posy), self.radius)

# Used to update the state of the object

def update(self):

self.posx += self.xFac\*self.speed

self.posy += self.yFac\*self.speed

# Reflecting the ball if it touches

# either of the vertical edges

if self.posx <= 0 or self.posx >= WIDTH:

self.xFac \*= -1

# Reflection from the top most edge of the screen

if self.posy <= 0:

self.yFac \*= -1

# If the ball touches the bottom most edge of

# the screen, True value is returned

if self.posy >= HEIGHT:

return True

return False

# Resets the position of the ball

def reset(self):

self.posx = 0

self.posy = HEIGHT

self.xFac, self.yFac = 1, -1

# Used to change the direction along Y axis

def hit(self):

self.yFac \*= -1

# Returns the rect of the ball. In this case,

# it is the ball itself

def getRect(self):

return self.ball

# Helper Functions

# Function used to check collisions between any two entities

def collisionChecker(rect, ball):

if pygame.Rect.colliderect(rect, ball):

return True

return False

# Function used to populate the blocks

def populateBlocks(blockWidth, blockHeight,

horizontalGap, verticalGap):

listOfBlocks = []

for i in range(0, WIDTH, blockWidth+horizontalGap):

for j in range(0, HEIGHT//2, blockHeight+verticalGap):

listOfBlocks.append(

Block(i, j, blockWidth, blockHeight,

random.choice([WHITE, GREEN])))

return listOfBlocks

# Once all the lives are over, this function waits until

# exit or space bar is pressed and does the corresponding action

def gameOver():

gameOver = True

while gameOver:

# Event handling

for event in pygame.event.get():

if event.type == pygame.QUIT:

return False

if event.type == pygame.KEYDOWN:

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

return True

# Game Manager

def main():

running = True

lives = 3

score = 0

scoreText = font.render("score", True, WHITE)

scoreTextRect = scoreText.get\_rect()

scoreTextRect.center = (20, HEIGHT-10)

livesText = font.render("Lives", True, WHITE)

livesTextRect = livesText.get\_rect()

livesTextRect.center = (120, HEIGHT-10)

striker = Striker(0, HEIGHT-50, 100, 20, 10, WHITE)

strikerXFac = 0

ball = Ball(0, HEIGHT-150, 7, 5, WHITE)

blockWidth, blockHeight = 40, 15

horizontalGap, verticalGap = 20, 20

listOfBlocks = populateBlocks(

blockWidth, blockHeight, horizontalGap, verticalGap)

# Game loop

while running:

screen.fill(BLACK)

screen.blit(scoreText, scoreTextRect)

screen.blit(livesText, livesTextRect)

scoreText = font.render("Score : " + str(score), True, WHITE)

livesText = font.render("Lives : " + str(lives), True, WHITE)

# If all the blocks are destroyed, then we repopulate them

if not listOfBlocks:

listOfBlocks = populateBlocks(

blockWidth, blockHeight, horizontalGap, verticalGap)

# All the lives are over. So, the gameOver() function is called

if lives <= 0:

running = gameOver()

while listOfBlocks:

listOfBlocks.pop(0)

lives = 3

score = 0

listOfBlocks = populateBlocks(

blockWidth, blockHeight, horizontalGap, verticalGap)

# Event handling

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

if event.type == pygame.KEYDOWN:

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

strikerXFac = -1

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

strikerXFac = 1

if event.type == pygame.KEYUP:

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

strikerXFac = 0

# Collision check

if(collisionChecker(striker.getRect(),

ball.getRect())):

ball.hit()

for block in listOfBlocks:

if(collisionChecker(block.getRect(), ball.getRect())):

ball.hit()

block.hit()

if block.getHealth() <= 0:

listOfBlocks.pop(listOfBlocks.index(block))

score += 5

# Update

striker.update(strikerXFac)

lifeLost = ball.update()

if lifeLost:

lives -= 1

ball.reset()

print(lives)

# Display

striker.display()

ball.display()

for block in listOfBlocks:

block.display()

pygame.display.update()

clock.tick(FPS)

if \_\_name\_\_ == "\_\_main\_\_":

main()

pygame.quit()

**3. Write a Python program to load an image on a surface and perform transformations in Pygame.**

import pygame

import sys

# Initialize Pygame

pygame.init()

# Set up the display

screen\_width, screen\_height = 800, 600

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

pygame.display.set\_caption("Image Transformations in Pygame")

# Load an image

image = pygame.image.load('Pygame\_logo.gif') # Replace 'your\_image.png' with the path to your image

image\_rect = image.get\_rect(center=(screen\_width // 2, screen\_height // 2))

# Transformation functions

def scale\_image(image, scale\_factor):

width = int(image.get\_width() \* scale\_factor)

height = int(image.get\_height() \* scale\_factor)

return pygame.transform.scale(image, (width, height))

def rotate\_image(image, angle):

return pygame.transform.rotate(image, angle)

def flip\_image(image, x\_bool, y\_bool):

return pygame.transform.flip(image, x\_bool, y\_bool)

# Main loop

running = True

scale\_factor = 1.0

angle = 0

flip\_x, flip\_y = False, False

while running:

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

elif event.type == pygame.KEYDOWN:

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

scale\_factor += 0.1

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

scale\_factor = max(0.1, scale\_factor - 0.1)

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

angle -= 10

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

angle += 10

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

flip\_x = not flip\_x

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

flip\_y = not flip\_y

# Clear the screen

screen.fill((255, 255, 255))

# Apply transformations

transformed\_image = scale\_image(image, scale\_factor)

transformed\_image = rotate\_image(transformed\_image, angle)

transformed\_image = flip\_image(transformed\_image, flip\_x, flip\_y)

transformed\_image\_rect = transformed\_image.get\_rect(center=(screen\_width // 2, screen\_height // 2))

# Draw the transformed image

screen.blit(transformed\_image, transformed\_image\_rect)

# Update the display

pygame.display.flip()

# Quit Pygame

pygame.quit()

sys.exit()

**4. Write a Python program to integrate PyOpenGL for 3D rendering and transformations in Pygame.**

import pygame

from pygame.locals import \*

from OpenGL.GL import \*

from OpenGL.GLUT import \*

from OpenGL.GLU import \*

import numpy as np

# Initialize Pygame and PyOpenGL

pygame.init()

display = (800, 600)

pygame.display.set\_mode(display, DOUBLEBUF | OPENGL)

gluPerspective(45, (display[0] / display[1]), 0.1, 50.0)

glTranslatef(0.0, 0.0, -5)

# Define a simple cube

vertices = (

(1, -1, -1),

(1, 1, -1),

(-1, 1, -1),

(-1, -1, -1),

(1, -1, 1),

(1, 1, 1),

(-1, -1, 1),

(-1, 1, 1)

)

edges = (

(0, 1),

(1, 2),

(2, 3),

(3, 0),

(4, 5),

(5, 6),

(6, 7),

(7, 4),

(0, 4),

(1, 5),

(2, 6),

(3, 7)

)

def draw\_cube():

glBegin(GL\_LINES)

for edge in edges:

for vertex in edge:

glVertex3fv(vertices[vertex])

glEnd()

# Main loop

running = True

while running:

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

if event.type == pygame.KEYDOWN:

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

glRotatef(5, 0, 1, 0)

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

glRotatef(-5, 0, 1, 0)

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

glRotatef(5, 1, 0, 0)

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

glRotatef(-5, 1, 0, 0)

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)

draw\_cube()

pygame.display.flip()

pygame.time.wait(10)

pygame.quit()

**5. Write a Python program to develop pong game in Pygame.**

import pygame

import sys

# Initialize Pygame

pygame.init()

# Screen settings

WIDTH, HEIGHT = 800, 600

screen = pygame.display.set\_mode((WIDTH, HEIGHT))

pygame.display.set\_caption('Pong')

# Colors

WHITE = (255, 255, 255)

BLACK = (0, 0, 0)

# Paddle settings

PADDLE\_WIDTH, PADDLE\_HEIGHT = 10, 100

paddle\_speed = 5

# Ball settings

BALL\_SIZE = 20

ball\_speed\_x = 5

ball\_speed\_y = 5

# Initialize paddles and ball

player1 = pygame.Rect(50, HEIGHT // 2 - PADDLE\_HEIGHT // 2, PADDLE\_WIDTH, PADDLE\_HEIGHT)

player2 = pygame.Rect(WIDTH - 50 - PADDLE\_WIDTH, HEIGHT // 2 - PADDLE\_HEIGHT // 2, PADDLE\_WIDTH, PADDLE\_HEIGHT)

ball = pygame.Rect(WIDTH // 2 - BALL\_SIZE // 2, HEIGHT // 2 - BALL\_SIZE // 2, BALL\_SIZE, BALL\_SIZE)

# Game loop

running = True

while running:

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

# Get keys pressed

keys = pygame.key.get\_pressed()

# Player 1 controls

if keys[pygame.K\_w] and player1.top > 0:

player1.y -= paddle\_speed

if keys[pygame.K\_s] and player1.bottom < HEIGHT:

player1.y += paddle\_speed

# Player 2 controls

if keys[pygame.K\_UP] and player2.top > 0:

player2.y -= paddle\_speed

if keys[pygame.K\_DOWN] and player2.bottom < HEIGHT:

player2.y += paddle\_speed

# Ball movement

ball.x += ball\_speed\_x

ball.y += ball\_speed\_y

# Ball collision with top/bottom

if ball.top <= 0 or ball.bottom >= HEIGHT:

ball\_speed\_y \*= -1

# Ball collision with paddles

if ball.colliderect(player1) or ball.colliderect(player2):

ball\_speed\_x \*= -1

# Ball goes out of bounds

if ball.left <= 0 or ball.right >= WIDTH:

ball.x, ball.y = WIDTH // 2 - BALL\_SIZE // 2, HEIGHT // 2 - BALL\_SIZE // 2

ball\_speed\_x \*= -1

# Drawing

screen.fill(BLACK)

pygame.draw.rect(screen, WHITE, player1)

pygame.draw.rect(screen, WHITE, player2)

pygame.draw.ellipse(screen, WHITE, ball)

pygame.draw.aaline(screen, WHITE, (WIDTH // 2, 0), (WIDTH // 2, HEIGHT))

# Update display

pygame.display.flip()

pygame.time.Clock().tick(60)

pygame.quit()

sys.exit()

**6. Write a Python program to develop flappy game in Pygame.**

import pygame

import random

import sys

# Initialize Pygame

pygame.init()

# Screen settings

WIDTH, HEIGHT = 400, 600

screen = pygame.display.set\_mode((WIDTH, HEIGHT))

pygame.display.set\_caption('Flappy Bird')

# Colors

WHITE = (255, 255, 255)

BLACK = (0, 0, 0)

# Bird settings

BIRD\_WIDTH, BIRD\_HEIGHT = 40, 40

bird = pygame.Rect(WIDTH // 4, HEIGHT // 2, BIRD\_WIDTH, BIRD\_HEIGHT)

bird\_speed = 0

gravity = 0.5

jump = -10

# Pipe settings

PIPE\_WIDTH = 70

PIPE\_HEIGHT = 400

pipe\_gap = 150

pipe\_speed = 3

pipe\_frequency = 1500 # milliseconds

last\_pipe = pygame.time.get\_ticks() - pipe\_frequency

pipes = []

# Game loop

running = True

while running:

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

if event.type == pygame.KEYDOWN:

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

bird\_speed = jump

# Bird movement

bird\_speed += gravity

bird.y += bird\_speed

# Generate pipes

time\_now = pygame.time.get\_ticks()

if time\_now - last\_pipe > pipe\_frequency:

pipe\_top = pygame.Rect(WIDTH, 0, PIPE\_WIDTH, random.randint(100, HEIGHT - pipe\_gap - 100))

pipe\_bottom = pygame.Rect(WIDTH, pipe\_top.height + pipe\_gap, PIPE\_WIDTH, HEIGHT - pipe\_top.height - pipe\_gap)

pipes.append(pipe\_top)

pipes.append(pipe\_bottom)

last\_pipe = time\_now

# Move pipes

for pipe in pipes:

pipe.x -= pipe\_speed

# Remove off-screen pipes

pipes = [pipe for pipe in pipes if pipe.x + PIPE\_WIDTH > 0]

# Check for collisions

for pipe in pipes:

if bird.colliderect(pipe):

running = False

if bird.top <= 0 or bird.bottom >= HEIGHT:

running = False

# Drawing

screen.fill(WHITE)

pygame.draw.rect(screen, BLACK, bird)

for pipe in pipes:

pygame.draw.rect(screen, BLACK, pipe)

# Update display

pygame.display.flip()

pygame.time.Clock().tick(60)

pygame.quit()

sys.exit()

**7. Write a Python program to develop tic tac toe game in Pygame.**

import pygame

import sys

# Initialize Pygame

pygame.init()

# Screen settings

WIDTH, HEIGHT = 300, 300

LINE\_WIDTH = 15

WIN\_LINE\_WIDTH = 15

BOARD\_ROWS = 3

BOARD\_COLS = 3

SQUARE\_SIZE = WIDTH // BOARD\_COLS

CIRCLE\_RADIUS = SQUARE\_SIZE // 3

CIRCLE\_WIDTH = 15

CROSS\_WIDTH = 25

SPACE = SQUARE\_SIZE // 4

# Colors

BG\_COLOR = (28, 170, 156)

LINE\_COLOR = (23, 145, 135)

CIRCLE\_COLOR = (239, 231, 200)

CROSS\_COLOR = (84, 84, 84)

# Screen

screen = pygame.display.set\_mode((WIDTH, HEIGHT))

pygame.display.set\_caption('Tic Tac Toe')

screen.fill(BG\_COLOR)

# Board

board = [[0 for \_ in range(BOARD\_COLS)] for \_ in range(BOARD\_ROWS)]

def draw\_lines():

# Horizontal

pygame.draw.line(screen, LINE\_COLOR, (0, SQUARE\_SIZE), (WIDTH, SQUARE\_SIZE), LINE\_WIDTH)

pygame.draw.line(screen, LINE\_COLOR, (0, 2 \* SQUARE\_SIZE), (WIDTH, 2 \* SQUARE\_SIZE), LINE\_WIDTH)

# Vertical

pygame.draw.line(screen, LINE\_COLOR, (SQUARE\_SIZE, 0), (SQUARE\_SIZE, HEIGHT), LINE\_WIDTH)

pygame.draw.line(screen, LINE\_COLOR, (2 \* SQUARE\_SIZE, 0), (2 \* SQUARE\_SIZE, HEIGHT), LINE\_WIDTH)

def draw\_figures():

for row in range(BOARD\_ROWS):

for col in range(BOARD\_COLS):

if board[row][col] == 1:

pygame.draw.circle(screen, CIRCLE\_COLOR, (int(col \* SQUARE\_SIZE + SQUARE\_SIZE // 2), int(row \* SQUARE\_SIZE + SQUARE\_SIZE // 2)), CIRCLE\_RADIUS, CIRCLE\_WIDTH)

elif board[row][col] == 2:

pygame.draw.line(screen, CROSS\_COLOR, (col \* SQUARE\_SIZE + SPACE, row \* SQUARE\_SIZE + SQUARE\_SIZE - SPACE), (col \* SQUARE\_SIZE + SQUARE\_SIZE - SPACE, row \* SQUARE\_SIZE + SPACE), CROSS\_WIDTH)

pygame.draw.line(screen, CROSS\_COLOR, (col \* SQUARE\_SIZE + SPACE, row \* SQUARE\_SIZE + SPACE), (col \* SQUARE\_SIZE + SQUARE\_SIZE - SPACE, row \* SQUARE\_SIZE + SQUARE\_SIZE - SPACE), CROSS\_WIDTH)

def mark\_square(row, col, player):

board[row][col] = player

def available\_square(row, col):

return board[row][col] == 0

def is\_board\_full():

for row in range(BOARD\_ROWS):

for col in range(BOARD\_COLS):

if board[row][col] == 0:

return False

return True

def check\_win(player):

# Vertical win

for col in range(BOARD\_COLS):

if board[0][col] == player and board[1][col] == player and board[2][col] == player:

draw\_vertical\_winning\_line(col, player)

return True

# Horizontal win

for row in range(BOARD\_ROWS):

if board[row][0] == player and board[row][1] == player and board[row][2] == player:

draw\_horizontal\_winning\_line(row, player)

return True

# Ascending diagonal win

if board[2][0] == player and board[1][1] == player and board[0][2] == player:

draw\_asc\_diagonal(player)

return True

# Descending diagonal win

if board[0][0] == player and board[1][1] == player and board[2][2] == player:

draw\_desc\_diagonal(player)

return True

return False

def draw\_vertical\_winning\_line(col, player):

posX = col \* SQUARE\_SIZE + SQUARE\_SIZE // 2

if player == 1:

color = CIRCLE\_COLOR

elif player == 2:

color = CROSS\_COLOR

pygame.draw.line(screen, color, (posX, 15), (posX, HEIGHT - 15), WIN\_LINE\_WIDTH)

def draw\_horizontal\_winning\_line(row, player):

posY = row \* SQUARE\_SIZE + SQUARE\_SIZE // 2

if player == 1:

color = CIRCLE\_COLOR

elif player == 2:

color = CROSS\_COLOR

pygame.draw.line(screen, color, (15, posY), (WIDTH - 15, posY), WIN\_LINE\_WIDTH)

def draw\_asc\_diagonal(player):

if player == 1:

color = CIRCLE\_COLOR

elif player == 2:

color = CROSS\_COLOR

pygame.draw.line(screen, color, (15, HEIGHT - 15), (WIDTH - 15, 15), WIN\_LINE\_WIDTH)

def draw\_desc\_diagonal(player):

if player == 1:

color = CIRCLE\_COLOR

elif player == 2:

color = CROSS\_COLOR

pygame.draw.line(screen, color, (15, 15), (WIDTH - 15, HEIGHT - 15), WIN\_LINE\_WIDTH)

def restart():

screen.fill(BG\_COLOR)

draw\_lines()

for row in range(BOARD\_ROWS):

for col in range(BOARD\_COLS):

board[row][col] = 0

draw\_lines()

player = 1

game\_over = False

# Main loop

while True:

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

sys.exit()

if event.type == pygame.MOUSEBUTTONDOWN and not game\_over:

mouseX = event.pos[0]

mouseY = event.pos[1]

clicked\_row = mouseY // SQUARE\_SIZE

clicked\_col = mouseX // SQUARE\_SIZE

if available\_square(clicked\_row, clicked\_col):

mark\_square(clicked\_row, clicked\_col, player)

if check\_win(player):

game\_over = True

player = player % 2 + 1

draw\_figures()

if event.type == pygame.KEYDOWN:

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

restart()

game\_over = False

player = 1

pygame.display.update()

**8. Write a Python program to develop a snake game in Pygame.**

import pygame

import time

import random

pygame.init()

# Screen dimensions

width = 800

height = 600

# Colors

white = (255, 255, 255)

yellow = (255, 255, 102)

black = (0, 0, 0)

red = (213, 50, 80)

green = (0, 255, 0)

blue = (50, 153, 213)

# Initialize game window globally

dis = pygame.display.set\_mode((width, height), pygame.RESIZABLE)

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

clock = pygame.time.Clock()

snake\_block = 10

snake\_speed = 15

font\_style = pygame.font.SysFont(None, 50)

score\_font = pygame.font.SysFont(None, 35)

def your\_score(score):

value = score\_font.render("Your Score: " + str(score), True, white)

dis.blit(value, [0, 0])

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

for x in snake\_list:

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

def message(msg, color):

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

dis.blit(mesg, [width / 6, height / 3])

def draw\_button(text, rect, color, hover\_color):

mouse\_pos = pygame.mouse.get\_pos()

mouse\_click = pygame.mouse.get\_pressed()

if rect.collidepoint(mouse\_pos):

pygame.draw.rect(dis, hover\_color, rect)

if mouse\_click[0] == 1:

return True

else:

pygame.draw.rect(dis, color, rect)

text\_surface = font\_style.render(text, True, white)

dis.blit(text\_surface, (rect.x + (rect.width - text\_surface.get\_width()) // 2, rect.y + (rect.height - text\_surface.get\_height()) // 2))

return False

def gameLoop():

global width, height, dis # Declare dis as global to modify it within the function

game\_over = False

game\_close = False

x1 = width / 2

y1 = height / 2

x1\_change = 0

y1\_change = 0

snake\_List = []

Length\_of\_snake = 1

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

foody = round(random.randrange(0, height - snake\_block) / 10.0) \* 10.0

while not game\_over:

while game\_close:

dis.fill(blue)

message("You Lost!", red)

if draw\_button("Try Again", pygame.Rect(width / 3, height / 2, 200, 50), green, yellow):

gameLoop()

pygame.display.update()

for event in pygame.event.get():

if event.type == pygame.QUIT:

game\_over = True

game\_close = False

for event in pygame.event.get():

if event.type == pygame.QUIT:

game\_over = True

if event.type == pygame.KEYDOWN:

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

x1\_change = -snake\_block

y1\_change = 0

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

x1\_change = snake\_block

y1\_change = 0

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

y1\_change = -snake\_block

x1\_change = 0

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

y1\_change = snake\_block

x1\_change = 0

if event.type == pygame.VIDEORESIZE:

width, height = event.w, event.h

dis = pygame.display.set\_mode((width, height), pygame.RESIZABLE)

if x1 >= width or x1 < 0 or y1 >= height or y1 < 0:

game\_close = True

x1 += x1\_change

y1 += y1\_change

dis.fill(blue)

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

snake\_Head = []

snake\_Head.append(x1)

snake\_Head.append(y1)

snake\_List.append(snake\_Head)

if len(snake\_List) > Length\_of\_snake:

del snake\_List[0]

for x in snake\_List[:-1]:

if x == snake\_Head:

game\_close = True

our\_snake(snake\_block, snake\_List)

your\_score(Length\_of\_snake - 1)

pygame.display.update()

if x1 == foodx and y1 == foody:

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

foody = round(random.randrange(0, height - snake\_block) / 10.0) \* 10.0

Length\_of\_snake += 1

clock.tick(snake\_speed)

pygame.quit()

quit()

gameLoop()