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()
```