|  |  |
| --- | --- |
| **TASK 12** | **Simulate Gaming concepts using Pygame** |

**Problem 1:**

Mr.John wants to design a breakout game Game. Help Mr.John to design the Game**.**

**PROGRAM**import pygame

import random

pygame.init() **# initializes the Pygame library**

white = (255, 255, 255) **# sets the colors**

black = (0, 0, 0)

blue = (0, 0, 255)

red = (255, 0, 0)

green = (0, 255, 0)

window\_width = 600 **# defines the window dimensions**

window\_height = 500

brick\_width = 60

brick\_height = 20

fps = 60

window = pygame.display.**set\_mode**((window\_width, window\_height)) **# The game window is created**

pygame.display.**set\_caption**("Breakout Game") **# to set the title to the game window**

paddle = pygame.Rect(window\_width // 2 - 50, window\_height - 30, 100, 10) **# paddle is a rectangle positioned near the bottom of the screen (x axis, y axis, width,height)**

paddle\_speed = **7 # the paddle will move 7 pixels in the respective direction.**

ball = pygame.Rect(window\_width // 2 - 10, window\_height // 2 - 10, 20, 20)

ball\_speed\_x = 5 \* random.choice((1, -1**)) # The ball starts moving at a speed of 5 pixels in both the x and y directions.**

ball\_speed\_y = -5

brick\_rows = 5 **# Grid of Bricks# has 5 rows and 10 columns of bricks.**

brick\_columns = 10

bricks = []

for row in range(brick\_rows):

brick\_row = []

for col in range(brick\_columns):

brick = pygame.Rect(col \* (brick\_width + 10) + 35, row \* (brick\_height + 5) + 35, brick\_width, brick\_height)

brick\_row.append(brick**) # Each brick is added to the bricks list.**

bricks.append(brick\_row)

def draw\_bricks(): **# Function to Draw Bricks**

for row in bricks:

for brick in row:

pygame.draw.rect(window, blue, brick)

def move\_ball(): **# Ball Movement and Collision Detection**

global ball\_speed\_x, ball\_speed\_y

ball.x += ball\_speed\_x

ball.y += ball\_speed\_y

if ball.left <= 0 or ball.right >= window\_width:

ball\_speed\_x \*= -1

if ball.top <= 0:

ball\_speed\_y \*= -1

if ball.colliderect(paddle):

ball\_speed\_y \*= -1

for row in bricks:

for brick in row:

if ball.colliderect(brick):

row.remove(brick)

ball\_speed\_y \*= -1

break

if ball.bottom >= window\_height:

return False

return True

def display\_text(text, size, color, x, y): **#Display Text "Game Over!" on the Screen**

font = pygame.font.SysFont("comicsansms", size)

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

window.blit(text\_surface, (x, y)) **#used to draw one surface onto another.**

def breakout\_game():

running = True

clock = pygame.time.Clock()

while running:

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

keys = pygame.key.get\_pressed()

if keys[pygame.K\_LEFT] and paddle.left > 0:

paddle.x -= paddle\_speed

if keys[pygame.K\_RIGHT] and paddle.right < window\_width:

paddle.x += paddle\_speed

if not move\_ball():

display\_text("Game Over!", 50, red, window\_width // 2 - 130, window\_height // 2)

pygame.display.flip()

pygame.time.delay(2000)

running = False

if all(len(row) == 0 for row in bricks):

display\_text("You Win!", 50, green, window\_width // 2 - 130, window\_height // 2)

pygame.display.flip()

pygame.time.delay(2000)

running = False

window.fill(black)

pygame.draw.rect(window, white, paddle)

pygame.draw.ellipse(window, red, ball)

draw\_bricks()

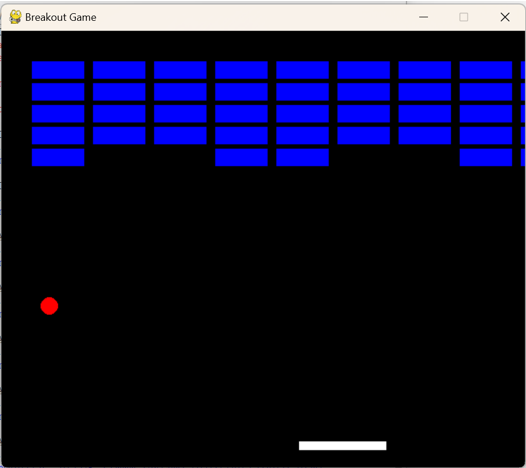
pygame.display.update()

clock.tick(fps)

pygame.quit()

breakout\_game()

**OUTPUT**



**Problem 2:**

**Mr.Nelson wants to design a Sudoku Game. Help Mr.Nelson to design the Game.**

**Program:**

import pygame

import time

pygame.init()

WIDTH, HEIGHT = 540, 600

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

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

WHITE = (255, 255, 255)

BLACK = (0, 0, 0)

GREY = (128, 128, 128)

BLUE = (0, 0, 255)

font = pygame.font.SysFont("comicsans", 40) **# font for rendering numbers in the grid is set up**

small\_font = pygame.font.SysFont("comicsans", 20)

board = [ #**Sudoku Board Initialization**

[5, 3, 0, 0, 7, 0, 0, 0, 0],

[6, 0, 0, 1, 9, 5, 0, 0, 0],

[0, 9, 8, 0, 0, 0, 0, 6, 0],

[8, 0, 0, 0, 6, 0, 0, 0, 3],

[4, 0, 0, 8, 0, 3, 0, 0, 1],

[7, 0, 0, 0, 2, 0, 0, 0, 6],

[0, 6, 0, 0, 0, 0, 2, 8, 0],

[0, 0, 0, 4, 1, 9, 0, 0, 5],

[0, 0, 0, 0, 8, 0, 0, 7, 9]

]

def draw\_grid(): #**Drawing the Grid**:

gap = WIDTH // 9 # Gap between the lines

for i in range(10):

if i % 3 == 0:

thick = 4

else:

thick = 1

pygame.draw.line(win, BLACK, (0, i \* gap), (WIDTH, i \* gap), thick)

pygame.draw.line(win, BLACK, (i \* gap, 0), (i \* gap, WIDTH), thick)

def draw\_numbers(board): # **Drawing Numbers**:

gap = WIDTH // 9

for i in range(9):

for j in range(9):

if board[i][j] != 0:

text = font.render(str(board[i][j]), True, BLACK)

win.blit(text, (j \* gap + 15, i \* gap + 15))

def draw\_selected(row, col): #**Highlighting Selected Cell**:

gap = WIDTH // 9

pygame.draw.rect(win, BLUE, (col \* gap, row \* gap, gap, gap), 4)

def main(): **#** **The game loop is started, and once exited, Pygame cleans up and exits properly.**

key = None

row = -1

col = -1

run = True

while run:

for event in pygame.event.get():

if event.type == pygame.QUIT:

run = False

if event.type == pygame.KEYDOWN:

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

key = 1

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

key = 2

if event.key == pygame.K\_3:

key = 3

if event.key == pygame.K\_4:

key = 4

if event.key == pygame.K\_5:

key = 5

if event.key == pygame.K\_6:

key = 6

if event.key == pygame.K\_7:

key = 7

if event.key == pygame.K\_8:

key = 8

if event.key == pygame.K\_9:

key = 9

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

board[row][col] = 0

key = None

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

# Confirm number input (for future logic to validate)

key = None

if event.type == pygame.MOUSEBUTTONDOWN:

pos = pygame.mouse.get\_pos()

row = pos[1] // (WIDTH // 9)

col = pos[0] // (WIDTH // 9)

win.fill(WHITE)

draw\_grid()

draw\_numbers(board)

if row >= 0 and col >= 0:

draw\_selected(row, col)

if key is not None and row >= 0 and col >= 0:

board[row][col] = key

pygame.display.update()

main()

pygame.quit()

**OUTPUT**

