TASK: 12
Date: 13/10/25

USE THE TKINTER MODULE FOR UI DESIGN

12-1 DEVELOP A MAZE GAME USING PYGAME.

Aim:-

To design and develop a simple maze game using Python's Pygame library, where the player navigates a square through a maze to reach the goal without colliding with walls.

Algorithm:-

1) Start
2) Import required modules:
   * Import Pygame for game development
   * Import sys to exit the program cleanly.
3) Initialize Pygame using Pygame.init()
4) Set up the game window:
   * Define screen width & height
   * Create the display surface
   * Set the window title
5) Define colours using RGB values for white, Black, blue, green
6) Create player & goal rectangles:
   * player starts at (50, 50)
   * Goal is placed at (550, 350).
7) Create maze walls:
   * Define a list of rectangular wall obstacles.
8) Define function check_collision(rect, walls).
   * check if the player's rectangle collides with any wall
   * Return True if collides

PROGRAM:-

```
import pygame
import sys
pygame.init()
WIDTH, HEIGHT = 600, 400
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Maze Game")

# Colors
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
BLUE = (0, 0, 200)
GREEN = (0, 200, 0)
RED = (200, 0, 0)
# clock
clock = pygame.time.Clock()
# player setup
player_size = 20
player = pygame.Rect(50, 50, player_size, player_size)
# Goal setup
goal = pygame.Rect(550, 350, player_size, player_size)
# Maze walls
walls = [ pygame.Rect(100, 0, 20, 300),
          pygame.Rect(200, 100, 20, 300),
          pygame.Rect(300, 0, 20, 250),
          pygame.Rect(400, 150, 20, 250),
          pygame.Rect(500, 0, 20, 250),
        ]
# Function to check wall collisions
def check_collision(rect, walls):
    for wall in walls:
```

```
        if rect.colliderect(wall):
            return True
    return False

# Game loop
running = True
while running:
    screen.fill(WHITE)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    keys = pygame.key.get_pressed()
    move_x, move_y = 0, 0
    if keys[pygame.K_LEFT]:
        move_x = -3
    if keys[pygame.K_RIGHT]:
        move_x = 3
    if keys[pygame.K_UP]:
        move_y = -3
    if keys[pygame.K_DOWN]:
        move_y = 3

    player.move_ip(move_x, move_y)
    if check_collision(player, walls):
        player.move_ip(-move_x, -move_y)
    if player.colliderect(goal):
        font = pygame.font.SysFont("Arial", 28)
        text = font.render("You win!", True, RED)
        screen.blit(text, (230, 180))
        pygame.display.flip()
        pygame.time.wait(2000)
        running = False
    for wall in walls:
        pygame.draw.rect(screen, BLACK, wall)
    pygame.quit()
    sys.exit()
```

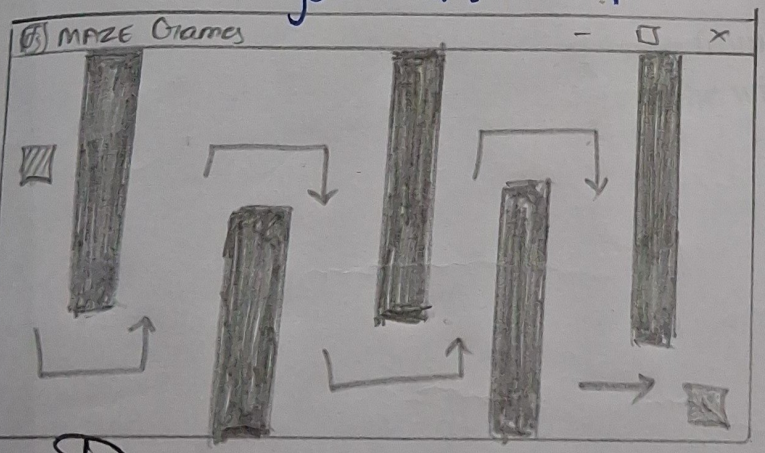
9 Main game loop:
- Fill the background
- Handle Quit events to close game
- Detect keyboard input
- Move the player & check for wall collisions.

10) Draw game objects:
- Draw wall, goal, and player.

11) Update display using pygame.display.flip()
12) Exit the game using pygame.quit() & sys.exit()
13 End

O/P:

- The window displays a maze made of black walls
- The blue square represents the player.
- The green square represents the goal.
- The player moves using arrow keys:
  - ↑ Up
  - ↓ down
  - ← Left
  - → Right
- If the player touches a wall, the move is undone
- When the player reaches the goal, a message "You win!." appears for 2 secs & game exits



EX.NO.	12
PERFORMANCE (5)	5
RESULT AND ANALYSIS	5
VIVA VOCE (5)	5
RECORD 5)	5
TOTAL (20)	5
SIGN	

20

RESULT:-

Thus the Python program has been successfully executed.