Task - 12

Simulate Gaming concept using Pygame

Aim: To simulate Gaming concept using Pygame

Snake game:

Write a python program to create a snake game using pygame package.

Algorithm

1. Import pygame package and initialize it
2. Define the window size
3. Create a snake class which initializes the snake position and movement
4. Create a function to check if the snake collides with the fruit
5. Create a game loop to continously update the game display.
6. End the game if the user quits

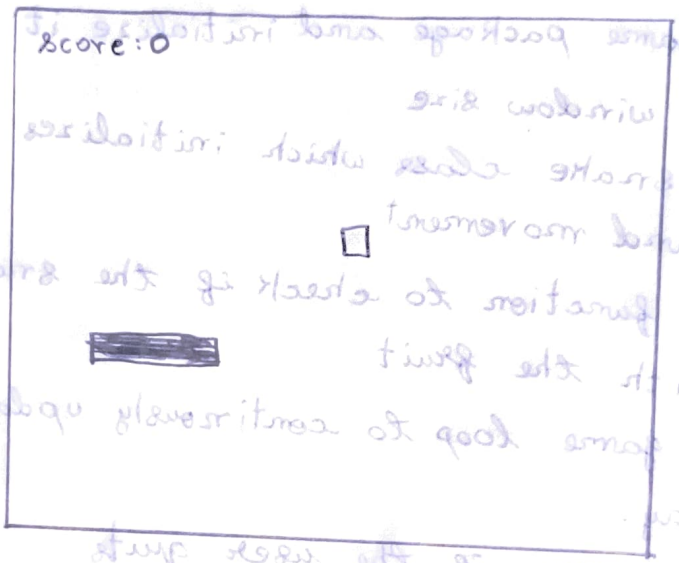
Program

```
import pygame
import time
import random
snake_speed = 15
window_x = 720
window_y = 480
black = pygame.color(0,0,0)
white = pygame.color(255,255,255)
red = pygame.color(255,0,0)
green = pygame.color(0,255,0)
blue = pygame.color(0,0,255)
```

```
Pygame.init()
Pygame.display.set_caption('GeeksforGeeks snakes')
game_window = pygame.display.set_mode((window_x, window_y))
fps = pygame.time.Clock()
snake_body = [[100, 50],
              [90, 50],
              [80, 50],
              [70, 50]
              ]
fruit_position = [random.randrange(1, (window_x//10))*10,
                  random.randrange(1, (window_y//10))*10]

fruit_position = True
direction = 'RIGHT'
change_to = direction
score = 0
def show_score(choice, color, font, size):
    score_font = pygame.font.SysFont(font, size)
    score_surface = score_font.render('Score : '+str(score),
                                      True, color)
    score_rect = score_surface.get_rect()
    game_window.blit(score_surface, score_rect)

def game_over():
    my_font = pygame.font.SysFont('times new roman', 50)
    game_over = my_font.render(
        'Your Score is : '+str(score), True, red)
    game_over_rect = game_over_surface.get_rect()
    game_over_rect.midtop = (window_x/2, window_y/4)
    game_window.blit(game_over_surface, game
                                        _over)

    pygame.display.flip()
    time.sleep(2)
    pygame.quit()
```

```
quit()
while true:
  for event in pygame.event.get():
    if event.type == pygame.KeyDown:
      if event.key == pygame.K_UP:
        change_to = 'UP'
      if event.key == pygame.K_DOWN:
        change_to = 'DOWN'
      if event.key == Pygame.K_LEFT:
        change_to = 'LEFT'
      if event.key == Pygame.K_RIGHT:
        change_to = 'RIGHT'
  if change_to == 'UP' and direction == 'DOWN':
      direction = 'UP'
  if change_to == 'DOWN' and direction != 'UP':
      direction = 'DOWN'
  if change_to == 'LEFT' and direction != 'RIGHT':
      direction = 'LEFT'
  if change_to == 'RIGHT' and direction != 'LEFT':
      direction = 'RIGHT'
  if direction == 'UP':
      snake_position[1] == 10
  if direction == 'DOWN'
      snake_position[1] += 10
  if direction == 'LEFT':
      snake_position[0] -= 10
  if direction == 'RIGHT':
      snake_position[0] += 10
  snake_body.insert(0, list(snake_position))
  if snake_position[0] == fruit_position[0]
      == fruit_position[1]:
      score += 10
```

Output:



```
else:
    snake_body.pop()
if not fruit_spawn:
    fruit_position = [random.randrange(1,(window_x//10))
        *10, random.randrange(1,(window_y//10))*10]
fruit_spawn = True
game_window.fill(black)
for pos in snake_body
    Pygame draw.reet(game_window, green)
        Pygame.reet(pos[0], pos[1], 10,10)]
Pygame.draw.reet(game_window, white, Pygame.reet)
    fruit_position[0], fruit position[1], 10,10))
if snake_position[1]<or snake_position[1] > window
                                    -y-10;
    game_over()
for block in snak.body[1:]:
    if snake.position[0] == block[0] and snake_position
                                    [1] = block[1]:
        game_over()
show_score(1, white, 'times new roman', 20)
Pygame.display.update()
fps.click(snake_speed)
```

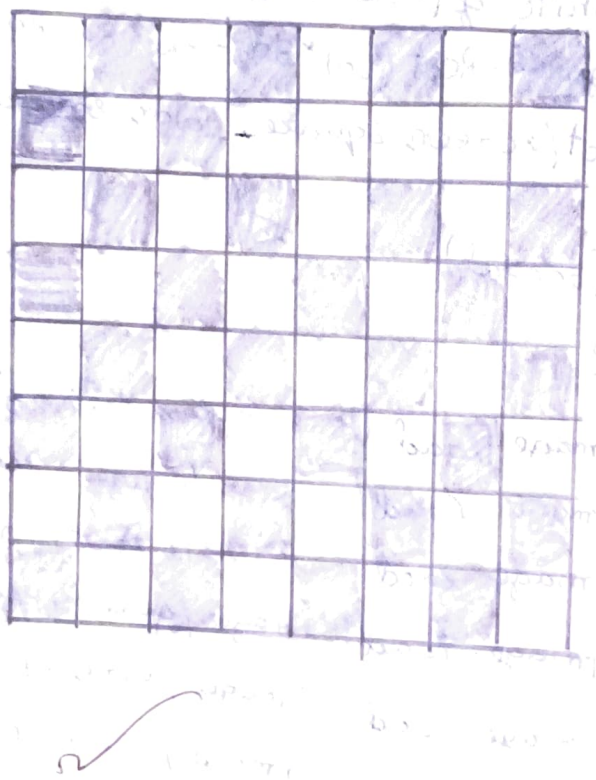
2. Write a programe to Develop a chess board using Pygame.

Program

```
import pygame
Pygame.init()
screen_size = (640,640)
screen = Pygame.Display.set_mode(screen_size)
```

```
black = (0, 0, 0)
white = (255, 255, 255)
brown = (153, 76, 0)

def draw_board():
    for row in range(8):
        for col in range(8):
            square_color = white if (row + col) % 2 == 0 else brown
            square_rect = pygame.Rect(col * 80, row * 80, 80, 80)
            pygame.draw.rect(screen, square_color, square_rect)

def draw_pieces(board):
    piece_images = {
        'r': pygame.image.load('images/rook.png'),
        'n': pygame.image.load('images/knight.png'),
        'b': pygame.image.load('images/bishop.png'),
        'q': pygame.image.load('images/queen.png'),
        'k': pygame.image.load('images/king.png'),
        'p': pygame.image.load('image/pawn.png'),
    }
    for row in range(8):
        for col in range(8):
            piece = board[row][col]
            if piece != '':
                piece_img = piece_images[piece]
                piece_rect = pygame.rect(col * 80, row * 80, 80, 80)
                screen.blit(piece_image, piece_rect)
```

output



```
board = [
    ['r', 'n', 'b', 'q', 'k', 'b', 'n', 'r'],
    ['p', 'p', 'p', 'p', 'p', 'p', 'p', 'p'],
    [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '],
    [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '],
    [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '],
    [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '],
    ['P', 'P', 'P', 'P', 'P', 'P', 'P', 'P'],
    ['R', 'N', 'B', 'Q', 'K', 'B', 'N', 'R']
]
draw_board()
draw_pieces(board)
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
pygame.display.update()
```

Result: Thus the program for pygame is executed and verified successfully.

VELTECH	
	12
FORMANCE (5)	5
LEARDATION/VISIS (5)	5
VOICE (5)	5
	20
	8

13/10