Task-12

Simulate Gaming concepts using pygame

Aim :-

To simulate Gaming concepts using pygame.

Algorithm :-

-> Import pygame package and initialize it
-> Define the window size and title
-> create a snake class which initializes the snake position, color, and movement.
-> create a fruit class which initializes the fruit position and color.
-> create a function to check if the snake collides with the fruit and end the game.
-> create a function to update the game display and and draw the snake and fruit.
-> create a game loop to continuously update the game display, snake position, and check of collisions.
-> End the game if the user quits the snake collides with the window.

Program :-

```
#importing libraries
import pygame
import time
import random
snake_speed = 15
#window size
window_x = 720
window_y = 480
#defining colors
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)
# initializing pygame
  pygame.init()
#initialise game window
pygame.display.set_caption('GeeksforGeeks Snakes')
game_window = pygame.display.set_mode([window_x,
                                       window_y])
# fps (frames per second) controler
fps = pygame.time.clock()
# defining snake default position
 snake_position = [100,50]
# defining first 4 blocks of snake body
    snake_body = [[100,50],
                  [90,50],
                  [80,50],
                  [70,50]
                  ]
  # fruit position
    fruit_position = [random.randrange(1,(window_x//10))*10,
                      random.randrange[1,(window_y//10)]*10]
        fruit_spawn = True
    #setting default snake direction towards
    # right
     direction = 'RIGHT'
     change_to = direction
     #initial score
       score = 0
     # displaying score function
     def show_score(choice, color, font, size):
     # creating font object score_font
     score_font = pygame.font.sysfont(font, size)
     #create the display surface object
     # score_surface
     score_surface = score_font.render['score:'+str(score),
                                       True, color]
```

```
# create a rectangular object for the text
# surface object
score - rect - score- surface.get - rect[]
# displaying text
 game- window.blit(score-surface, score-rect)
# game over function
 def game -over[]:
    # creating font object my-font
    my-font = pygame.font.sysfont('times new roman', 50)
   # creating a text surface on which text
    # will be drawn
    game - over - surface = my-font - render(
        'your score is: '+ str(score), true, red)
    # create a rectangular object for the text
    # surface object.
    game-over-rect = game-over - surface.get-rect[]
    # setting position of the text
    game -over -rect- midtop = (window- x/2, window-y/4)
   #blit will draw the text on screen
   game -window.blit(game-over-surface, game-over-rect)
   pygame.display.flip[]
    #after 2 seconds we will quit the program
    time.sleep[2]
    # deactivating pygame library
     pygame.quit[]
    #quit the program
      quit()
    #main function
   while true:
      #handling key events
      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 two keys pressed simultaneously
    # we don't want snake to move into two
    # directions simultaneously
    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'
    # Moving the snake
    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 growing mechanism
    # if fruits and snakes collide then scores
    # will be incremented by 10
```

output

score :0

```
snake_body.insert(0,list(snake_position))
if snake_position[0]==fruit_position[0] and snake_position[1]
                                    ==fruit_position[1]:
    score+=10
    fruit_spawn=false
  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.rect(game_window,green, pygame.
          draw.rect[pos[0],pos[1],10,10]]
  pygame.draw.rect[game_window,white,pygame.Rect(
    fruit_position[0],fruit_position[1],10,10)]
    # Game over conditions
  if snake_position[0]<0 or snake_position[0]>
                                    window_x-10:
      game_over()
  if snake_position[1]<0 or snake_position[1]>window
                                    _y-10:
      game_over()
  ## displaying score continuously
    # touching the snake body
    for block in snake_body[1:]:
  if snake_position[0]==block[0] and snake_position
    [1]==block[1]:
      game_over[]
# displaying score continuously
  show_score(1,white,'times new roman',20]
  # refresh game screen
  pygame.display.update()
  # frame per second/ Refresh Rate
    fps.tick[snake_speed]
```

Task-12.2

Aim:- To write a python program to develop a chess board using Pygame.

Algorithm:-

-> Import pygame and initialize it.
-> Set screen size and title
-> Define colors for the board and pieces.
-> Define the initial state of the board as a list of containing the pieces.
-> Draw the board and pieces on the screen.
-> Start the game loop.

Program:-

```
import pygame
# initialize pygame
  pygame.init()
 # Set screen size and title
 screen_size = [640,640]
 screen = pygame.display.set_mode(screen_size)
 pygame.display.set_caption('Chess Board')
   # Define colors
   black = [0,0,0]
   white = [255,255,255]
   brown = [153,76,0]
   #Define function to draw the board
    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)
     # Define function to draw the pieces
     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('images/pawn.png')
```

Output:-

```
}
for row in range(8):
    for col in range(8):
        piece = board[row][col]
        if piece != '.':
            piece_image = piece_images[piece]
            piece_rect = pygame.Rect(col*80, row*80, 80, 80)
            screen.blit(piece_image, piece_rect)

# Define initial state of the board.
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 and pieces
draw_board()
draw_pieces(board)
# start game loop
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            quit()
    pygame.display.update()
```

completed

VEL TECH	
EX NO.	2
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
TOTAL (20)	
SIGN WITH DATE	15

Result:

Thus the program for pygame is executed and verified successfully.