Task-12: Simulate gaming concepts using Pygame.

Aim: To simulate gaming concepts using Pygame.

snake game:

1. Write a Program to create a snake game using Pygame Package.

conditions:

1. set the window size
2. create a snake
3. make the snake to move in the direction when left, right, down and up key is pressed.
4. when the snake hits the fruit. Increase the score by 10.
5. If the snake hits the window. Game over.

Algorithm:

1. Import Pygame package and initialize it.
2. Define the window size and title.
3. Create a snake class which initializes the snake Position, colour and movement.
4. Create a fruit. Position and colour.
5. create a function to check if the snake collides with fruit and increase the score.
6. create a function to check Position based on user input.
7. Create a game loop to continuously update the game display, snake Position, and check for collisions.
8. End the game if the user quits or snake collides with the window.

Program:

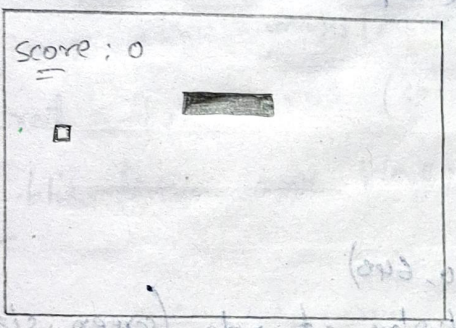
```
Import Pygame
Import time.
Import random
snake_speed = 15
window_x = 720
window_y = 480.
black = Pygame.colour(0,0,0)
white = pygame.colour(255,255,255)
red = pygame.colour(255,0,0)
green = Pygame.colour(0,255,0)
blue = Pygame.colour(0,0,255)
Pygame.init()
Pygame.display.set_caption('Greeks for Geeks.snakes')
games_window = Pygame.display.set_mode(window_x,
                                       window_y))
fps = Pygame.time.clock()
snake_Position = [100,50]
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_spawn = True.
direction = 'right'
change_to = direction
score = 0
def show_score (choice, color, font, size)
score_font = Pygame.font.sys_font(font, size)
score_surface = score_font.render('score:' +str(score),
                                  True, colour)
```

```
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_surface = 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, rect)
pygame.display.flip()
Time sleep(2).
pygame.quit()
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 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] 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 = rect
   [pos[0], pos[1], 10, 10)).
```

```
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().
For blocks in snake_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.tick(snake_speed).
```

output:



2. write a Python program to develop a chess board using Pygame.

Algorithm:

1. Import pygame and initialize it
2. Set screen size and title.
3. Define colours for board and pieces.
4. Define a function to draw the pieces on the board by loading images for each piece.
5. Define initial state of board as a list of lists containing pieces.
6. Draw the board and pieces on the screen
7. Start the game loop.

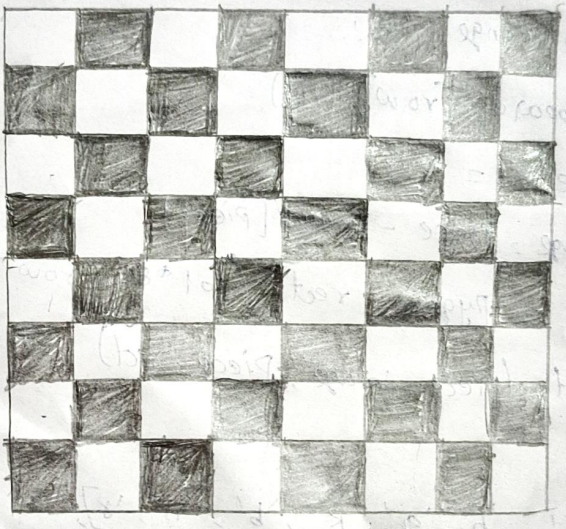
Program:

```
Import pg game.
 Pygame int ()
screen_size = (640, 640)
screen = Pygame.display set_mode (screen.size)
 Pygame.display.set.caption ('chess Board')
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_reot = Pygame.rect (col*80, row*80, 80, 80)
Pygame.draw.rect (screen - 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/king.png'),
'p': pygame.image.load('images/pawn.png')
}
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)
board [
['r', 'n', 'b', 'q', 'k', 'b', 'n', 'r'],
['p', 'p', 'p', 'p', 'p', 'p', 'p', 'p'],
['.', '.', '.', '.', '.', '.', '.', '.'],
['.', '.', '.', '.', '.', '.', '.', '.'],
['P', 'N', 'B', 'Q', 'K', 'B', 'N', 'P']
]
draw = board()
draw_Pieces(board).
```

```
while True:
for event in pygame.event.get():
if event.type = pygame.QUIT:
 pygame.quit()
 quit().
pygame.display.update().
```

output:



VEL TECH - CSE	
EX NO.	12
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
TOTAL (20)	
SIGN WITH DATE	15

Completed

Result:

Thus the Program for Pygame is executed and verified successfully.