Task 12

Simulate Gaming Concepts using pygame.

Aim To Simulate Gaming Concepts using Pygame.

Snake Game:-

Problem: Write a python 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 directions 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 initalize it.
2. Define the window size and title
3. Create a snake class which initialize

the Snake position, color, and movement.

4. Create a fruit class which initialize the fruit position and color.

5. Create a function to check if the Snake collides with the fruit and increase the score.

6. Create a function to check if the Snake collides with the windows and end the game.

7. Create a function to update the Snake position based on the user input

8. Create a function to update the game display and draw the Snake and fruit.

9. Create a game loop to continuously update the game display, snake position, and check for collisions

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)
# initialising 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) controller
fps = pygame.time.Clock()
# defining snake default position
```

```
snake_position = [100, 50]
# defining snake default position
snake_body = [[100, 50], [90, 50],
              [80, 50], [70, 50]]
#fruit position
fruit_position = [random.randrange
(1, (window_x//10))*10, random.randrang
    (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.
```

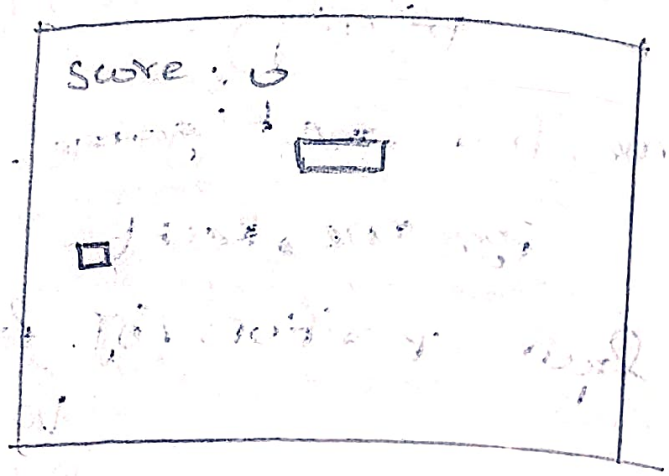
```
game_over_rect = game_over_surface.get
                 get_rect()

# Setting position of the text
  game_over_rect.midtop = (window_x/2,
                           window_y/4)

# blit will draw the text on screen
  game_over_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
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:
        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))
pygame.draw.rect(game_window, white,
        pygame.Rect(
    fruit_position[0], fruit_position[1]
                10,10))

# Game over condition

    game_over
    if snake_position[0] < 0 or snake[0]
        window_x-10:
        game_over()

if snake_position[1] < 0 or snake_position
    [1]=>block[1]:
        game_over()

# displaying score continously
    show_score(1,white,'times new
            roman',20)

# Refresh game screen
    pygame.display.update()
```

output:-



# frame per second / Refresh Rate

fps = frame_rate / snake_speed

Result:- Thus, the python program to simulate learning concepts using ~~and~~ successfully executed.

Task 12-1

Write a python program to Develop a chess board using Pygame

Aim:- To use the python program to develop a chess board using pygame

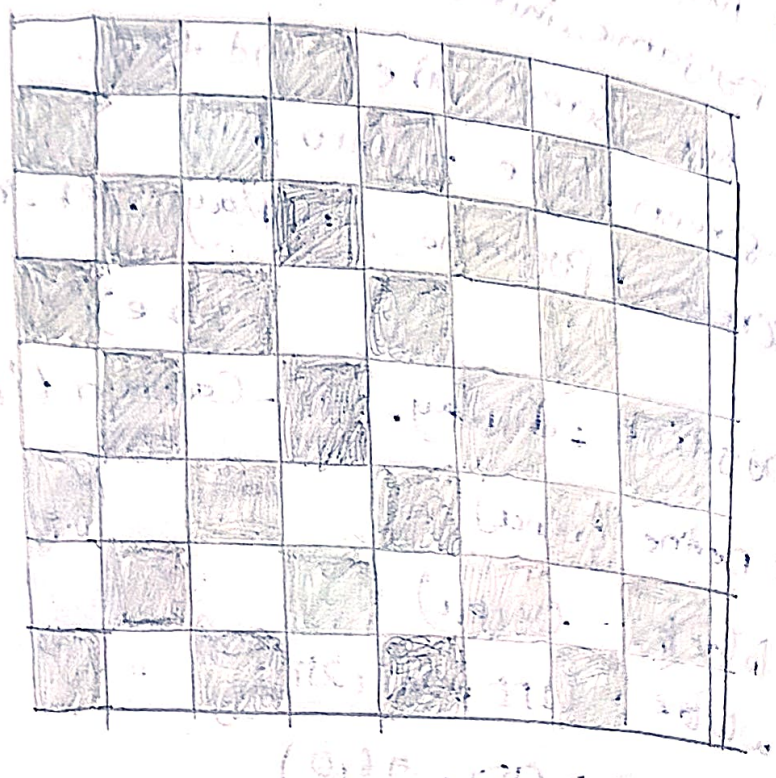
Algorithm

1. Import Pygame and initialize it.
2. Set screen size and title.
3. Define colors for the board and pieces.
   Define a function to draw the board by looping over rows and columns and drawing squares of different colors.
4. Define a function to draw the pieces on board by loading images for each piece and placing them on the corresponding square.
5. Define the initial state of the board as a list of lists containing the pieces.
6. Draw the board and pieces on the screen.
7. 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, color,
                  square_rect)
```

Output



# define function to draw the pieces

```
def draw_pieces(board):
    piece_images = {
    'r': pygame.image.load('image/rook.png'),
    'n': pygame.image.load('image/knight.png'),
    'b': pygame.image.load('image/bishop.png'),
    'q': pygame.image.load('image/queen.png'),
    'k': pygame.image.load('image/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_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'],
['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()
```

| VEL TECH - CSE | |
|---|---|
| EX NO. | 12 |
| PERFORMANCE (5) | 5 |
| RESULT AND ANALYSIS (3) | 3 |
| VIVA VOCE (3) | 3 |
| RECORD (4) | 4. |
| TOTAL (15) | |
| SIGN WITH DATE | |

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