TASK-12 Simulate Gaming concepts using Pygame

Aim: To simulate Gaming concepts using Pygame

Algorithm:

1. Import Pygame Package and initialize it
2. Define the window size and title
3. create a snake class which initializes the snake Postion color, and movement.
4. create a fruit class which initializes 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 fruit and 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 continously update the game display. snake Position, and check for collisions.
10. End the game if the user quit or the snake collides with the window.

output

score : 0

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()
# Intialise game window
pygame.display.set_caption('Geeks Geeks snake')
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 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 object
    score_rect = score_surface.get_rect()
# game over function
def game_over():
    # creating font object my_font
    my_font = pygame.font.SysFont('times new roman', 50)
# creating a text = my_font.render(
```

```
'Your Score is : ' + str(score), True, red)
# 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 even.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

  # Moving the snake
  if direction == 'UP'
      snake_Position[1] -= 10
  # snake body growing mechanism
  # if fruit 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

    score += 10
    fruit_spawn = False
    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(back)
      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 conditions
      if snake_position[0] <0 or snake_position[1]
      > window_y-10;
        game over()
# 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,'time new romon',20)
# Refersh game screen
      Pygame.display.updates()
```

Frame Per Second / Refresh Rate

FPS . tick (snake_speed).

Result: Thus the c-Program Gaming concepts using Pygame is executed verified successfully.

Task 12.2 To develop a chess board using Pygame.

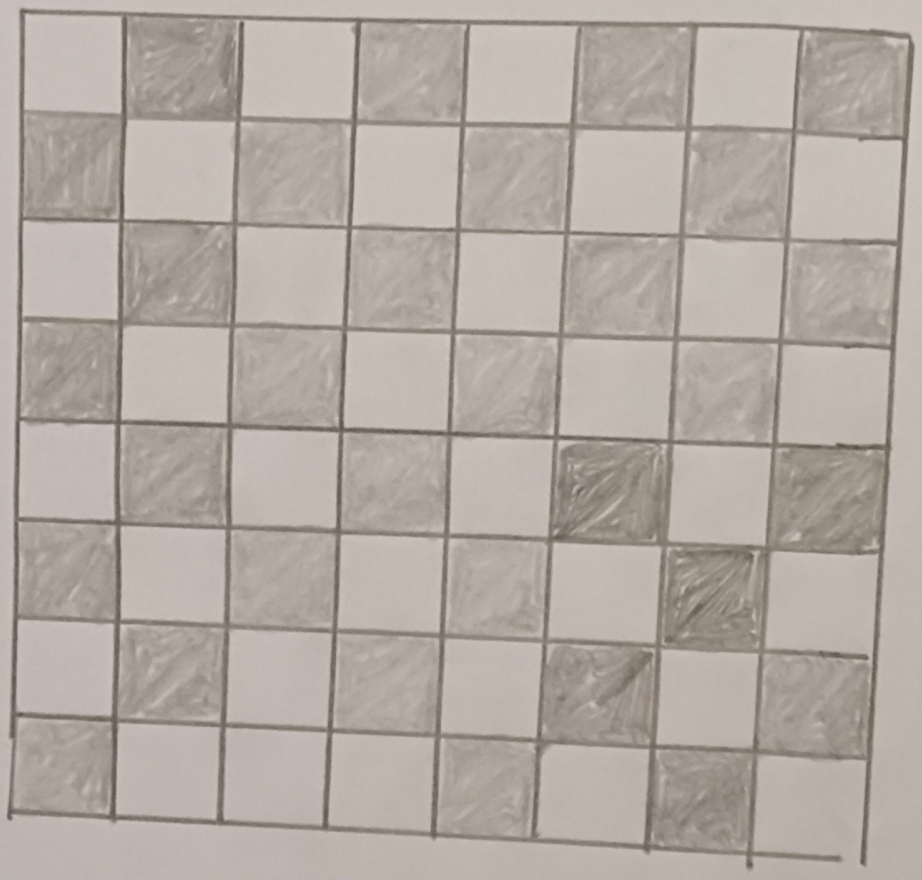
Aim: To develop a chess board using Pygame

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 the board by loading images for each Piece and Placing them on the corresponding square of different colors
5. Define the initial state of the board as a list of lists containing the Pieces.
6. Draw the bard 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 = (150,76,0)
# 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_image = {
    'r': Pygame.image.load('images/rook.png'),
    'n': Pygame.image.load('images/knight.Png'),
    'b': Pygame.image.load('images/queen.Png'),
    'q': Pygame.image.load('images/king.Png'),
    'k': Pygame.image.load('images/king.Png'),
    'p': Pygam.image.load('images/Pawn.Png')
    }
    for row in range(8):
        for col in range(8):
            Piece = board[row][col]
            if Piece != ' '
            Piece_image = Price_image[Piece]
            Piece_rect = Pygame.Rect(col*80 row*80,80,80)
            screen.bit(Piece_Image, Piece_rect).
```

```
# Define initial state of the board
board = [
['r','n','b','k','n','r']
['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 - CS	
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.