Task12. Simulate Gaming Concepts using Pygame

Aim:

To simulate Gaming Concepts using Pygame.

12.1: write a Python Program to create a SnakeGame using Pygame Package.

Algorithm:

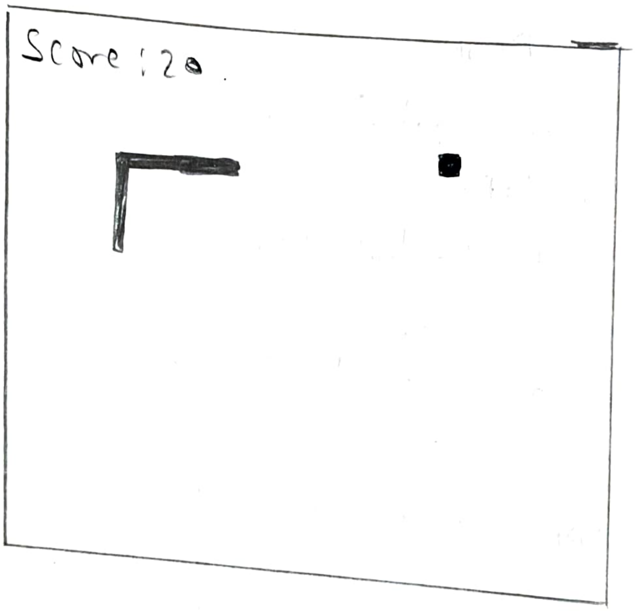
1. Import Pygame package and initialize it.
2. Define the window size and title.
3. Create a snake class which initializes the snake position, color, and movement. Create a fruit class which initializes the fruit position and colour.
4. Create a function to check if the snake ~~conditions~~ collides with the window and end the game.
5. Create a function to update the snake position based on user input.
6. Create a function to update the game display and draw the snake and fruit.
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 the snake collides with the window

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~~ 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_Position=[100,50]
Snake_body=[[100,50],[90,50],[80,50],[70,50]]
fruit_position=[random.randrange(1,(window_x//10))*10, randon.rand
range(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.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_surface=my_font.render(~~'Your~~ '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()
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
```


Score : 20.

```
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))
Pygame.draw.rect(game_window, white, Pygame.Rect(fruit_Position
[0], fruit_position[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 block 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)
```

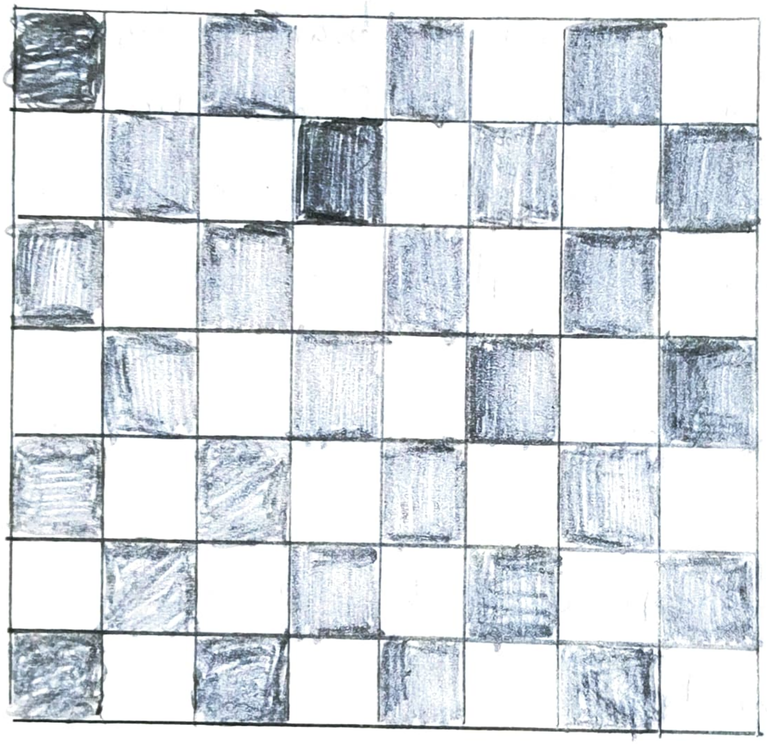
12.2| Write a Python Program to develop a chess board using Pygame.

Algorithm:

1. Import Pygame and initialize it.
2. Set Screen size and title.
3. Define colors for the board and pieces.
   Define a function to draw ~~Pieces~~ 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 squares.
5. Define the initial state of the board of a list of lists containing the pieces.
6. Draw the board and pieces on the screen.
7. Start the game loop.

Program:

```
import pygame
Pygame.init()
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)
#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)
def draw_pieces(board):
    Piece_images={`r':Pygame.image.load(`images/ruck.png'),
                 `n':Pygame.image.load(`images/knight.png'),
                 `b':Pygame.image.loud(`images/bishop.Png'),
                 `q':Pygame.image.loud(`images/queen.Png'),
                 `k':Pygame.image.loud(`images/King.Png'),
                 `p':Pygame.image.loud(`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', '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()
         quit()
  pygame.display.update()
```

VELTECH	
EX No.	12
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	20
SIGN WITH DATE	[illegible] 15/10

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