Date: 22-10-25 Simulate gaming concepts using pygame.

TASK 12

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. Creat a snake
3. make the snake to move in the directions when left, right down and up key is pressed
4. When 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 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 collide with the window and end the game.
6. Create a function to update the game display and draw the snake and fruit
7. Create a game loop to continously update the game display, Snake. Motion and Clear for collision

program:

```
# importing libraries
import pygame
import time
import random
snake_speed = 15
# window size
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.Color(0,255,0)
blue = pygame.Color(0,0,255)

pygame.init()

pygame = display.set_caption

game_window = pygame.display.set_mode
# FPS
FPS = pygame.time.Clock()
# define snake default position
snake_position = [100,50]
snake_body = [[100,50],[90,50],[80,50],[70,50]]
# default position
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):
# creating font object score_font
score_font = pygame.font.SysFont(font, size)
game_over_react = game_over_surface.get_rect()
# setting position of the text
game_over_rect.midtop = (window_x/2, window_y/4)
# blit will draw text on screen
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'
# moving 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.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
    not fruit_spawn:
    fruit_position = (random.randrange
        (window_x//10)) * 10,
    random.randrange(1, (window_y//10)) * 10
```

SCRIPT

```
fruit_spawn = True
game_window = pile(black)
for pain snake.body:
    pygame.draw.rect(game_window, green,
        pygame.Rect(pos[0], pos[1], (or[0]))
pygame.draw.rect(game_window, white, pygame.Rect(
    fruit_position[0], fruit_position[1], 10,10))
if snake_position[0] == or snake[0]
    window_x-10:
    game_over()
if snake_position[0] == or snake_Position
    [1] == block[1]:
        game_over()
show_score('white', times new roman: 20)
# Refresh game screen
pygame.display.update()
# frame per second
# S.tick(snake_speed)
```

Result:

Thus, the python program to simulate gaming concept 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 Checks board using pygame.

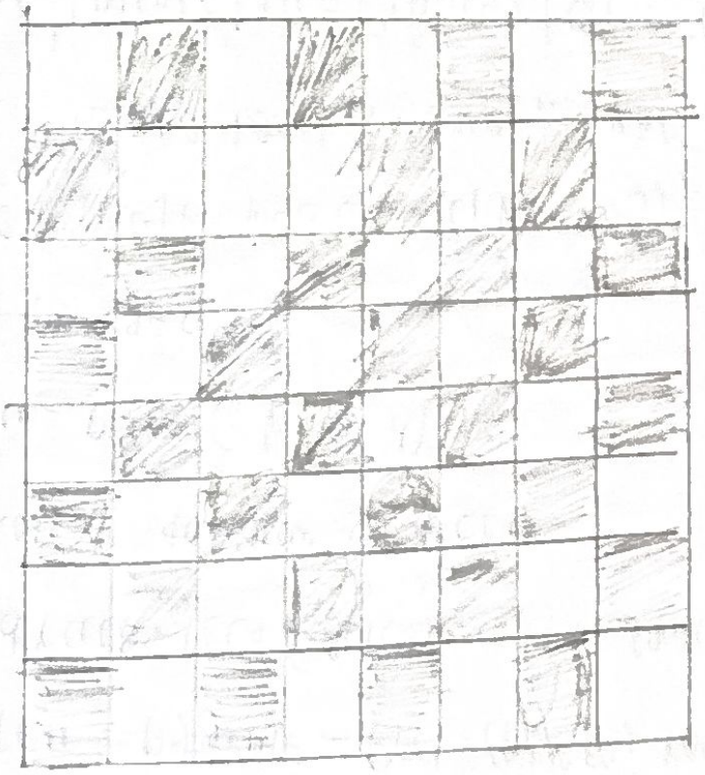
Algorithm:

1. import pygame and initialize
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 colums 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.
5. Draw the pieces on the screen
6. Start the game loop

program:

```
import pygame
# initialize program
    pygame.init()
    screen_size = (
    screen = pygame.display.set_mode(screen_size)
    pygame.display.set_caption("Chess Board")
# Define colors
    black = (0,0,0)
    white = (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_rect = pygame.Rect(col*80, row*80, 80, 80)
            pygame.draw.rect(screen, color, square_rect)
# Define function to draw the pieces
    def draw_pieces(board):
        pieces_images = {
    'r': pygame.image.load('images/bishop.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': 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']
]
# draw board and pieces
    draw_board()
    draw_pieces(board)
    while True:
    # for event in pygame.weight():
            if event.type==pygame.QUIT:
            pygame.quit()
                quit()
    pygame.display.update()
```

completed

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	15

Result:

Thus, the program for pygame is executed and verified successfully