Task 12 : Simulate Gaming concepts using Pygame

Aim: To simulate Gaming concepts using Pygame

Snake Game:

Problem 1. Write a python 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 initialize it
2. Define the window size and title
3. Create a snake class which initializes the snake position, 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 window and end the game
7. Create a function to update the game display and draw the snake and fruit
8. End the game if the user quits or the snake collides with the window

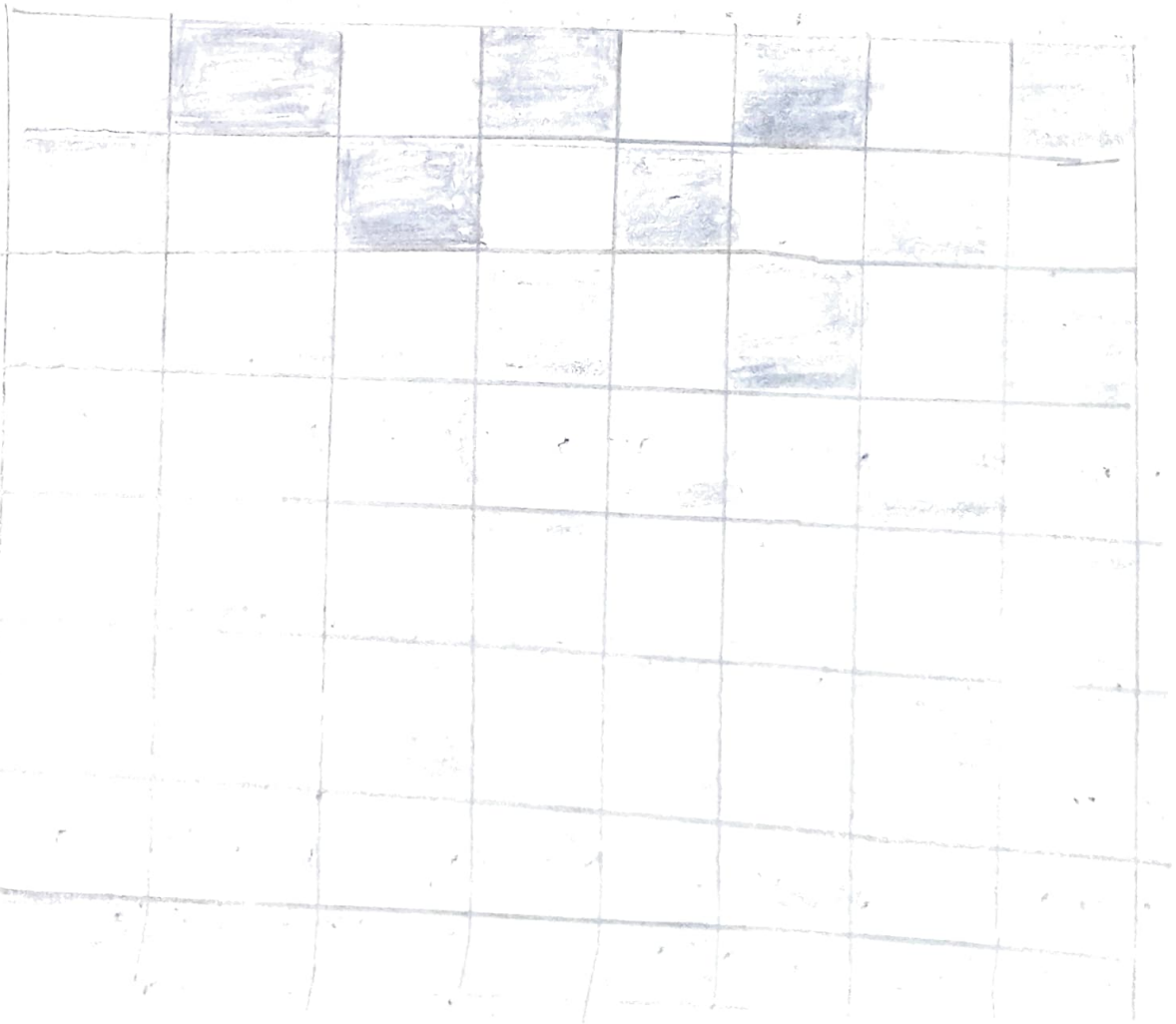
Program:

```
# importing libraries
import pygame
import time
import random

snake_speed = 15
# window size
window_x = 720
window_y = 480
```

output

```
# defone 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()

# defending 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_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):
```

Sample output :-



```
# creating font object score_font
score_font = pygame.font.SysFont('Font_size')

# create the display surface object
# score_surface
score_surface = score_font.render

# 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_rect()

# 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()
```

```
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 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:
    Py game . draw . rect (game - window, gree
                    Py game . Rect (pos[0], pos[1], 1
                    10))

    Py game . draw . rect (game - window, white,
                    Py game . Rect(
                    fruit - position[0], fruit - position
                    [1], 10, 10))

    # Game over conditions
if snake - position[0] < 0 or snake - position
                    [0] > window_x - 10,

        game - over()

if snake - position [1] < 0 or snake - positi
                    [1] == block[1]

    * displaying score continuously
show - score (1, white, 'times new roman',

# Refresh game screen
    Pygame . display . update()

# Frame per second / Refresh Rate
fps. tick (snake - speed)
```

Problem 2. Write a python program to Develop chess board using .pygame

Algorithm =

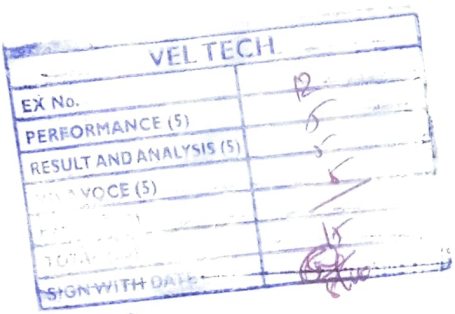
1. import pygame and initialize it.
2. Set Screen size and title
3. Define colors for the board and pieces.
4. Define a function to draw the pieces on the board by loading images for each piece and placing them on the corresponding square.
6. Draw the board and pieces on the Screen
7. start the game 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)
def draw_board():
    for row 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/rook
                               .png')
```

```
'n': pygame.image.load('images/knight.png'),
'b': pygame.image.load('images/bishop.png'),
'q': pygame.image.load('images/queen.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)

# Define initial state of the board
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()
```



Result:- Thus the program for pygame is executed and verified success fully.