Task 12.Simulate Gaming concepts using PygameCO5-K5

Aim: To Simulate Gaming concepts using Pygame

SnakeGame:

1.Write a python program to create a snakeGame 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
Sample Output:

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 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
10. End the game if the user quits or the snake collides with the window

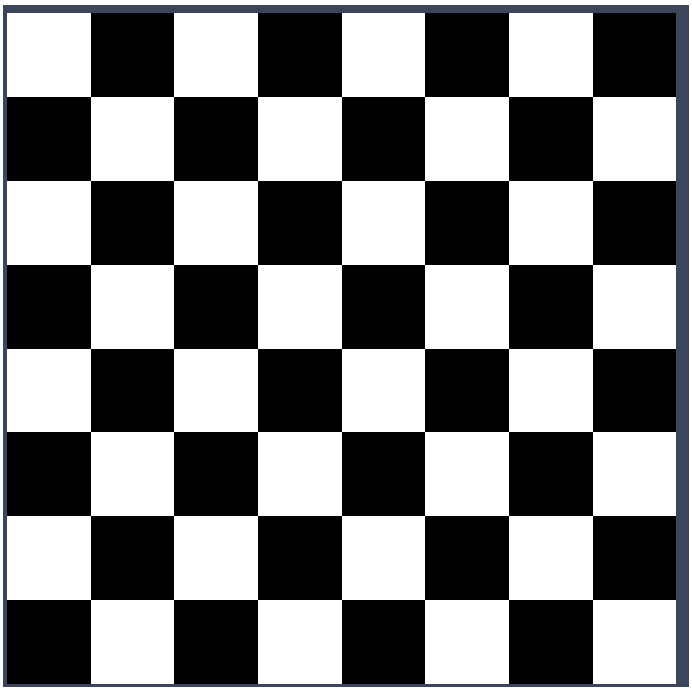
Program:

```
importpygame
import random

# Initialize Pygame package
```

```
pygame.init()

# Define window size and title
WINDOW_WIDTH = 500
WINDOW_HEIGHT = 500
WINDOW_TITLE = "Snake Game"
window = pygame.display.set_mode((WINDOW_WIDTH, WINDOW_HEIGHT))
pygame.display.set_caption(WINDOW_TITLE)

# Definecolors
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
RED = (255, 0, 0)
GREEN = (0, 255, 0)

# Define Snake class
class Snake:
def __init__(self):
self.x = WINDOW_WIDTH/2
self.y = WINDOW_HEIGHT/2
self.color = GREEN
self.direction = "right"
self.speed = 10
self.size = 10
self.body = [(self.x, self.y)]

def move(self):
ifself.direction == "right":
self.x += self.speed
elifself.direction == "left":
self.x -= self.speed
elifself.direction == "down":
self.y += self.speed
elifself.direction == "up":
self.y -= self.speed

self.body.insert(0, (self.x, self.y))
self.body.pop()

def draw(self):
for x, y in self.body:
pygame.draw.rect(window, self.color, [x, y, self.size, self.size])

# Define Fruit class
class Fruit:
def __init__(self):
self.x = random.randint(0, WINDOW_WIDTH-self.size)
self.y = random.randint(0, WINDOW_HEIGHT-self.size)
self.color = RED
self.size = 10
```

```
def draw(self):
pygame.draw.rect(window, self.color, [self.x, self.y, self.size, self.size])

# Define function to check collision with fruit
defcheck_fruit_collision(snake, fruit):
ifsnake.x == fruit.x and snake.y == fruit.y:
fruit.x = random.randint(0, WINDOW_WIDTH-fruit.size)
fruit.y = random.randint(0, WINDOW_HEIGHT-fruit.size)
snake.body.append((snake.x, snake.y))
return True
return False

# Define function to check collision with window
defcheck_window_collision(snake):
ifsnake.x< 0 or snake.x> WINDOW_WIDTH-snake.size or snake.y< 0 or snake.y>
WINDOW_HEIGHT-snake.size:
return True
return False

# Initialize Snake and Fruit objects
snake = Snake()
fruit = Fruit
```

Write a python program to Develop a chess board using pygame.
Sample output:



Algorithm:

1. Import pygame and initialize it.
2. Set screen size and title.
3. Define colors for the board and pieces.

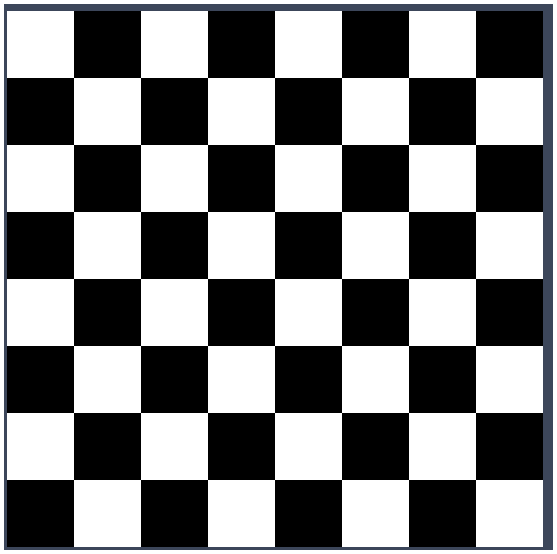
Define a function to draw the board by looping over rows and columns and drawing squares of different colors.

4. Define a function to draw the pieces on the board by loading images for each piece and placing them on the corresponding square.
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:

```
importpygame

# 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')

# Definecolors
black = (0, 0, 0)
white = (255, 255, 255)
brown = (153, 76, 0)

# Define function to draw the board
defdraw_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
defdraw_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 and pieces
draw_board()
draw_pieces(board)

# Start game loop
while True:
for event in pygame.event.get():
ifevent.type == pygame.QUIT:
pygame.quit()
quit()
pygame.display.update()
```



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