TASK-12: Simulate Gaming Concepts using pygame

13/10/25

Aim: To simulate gaming concepts using pygame.

snakegame:-

Problem: Write a python program to create a snakegame using pygame package

Conditions:-

1. set the window size
2. create a snake
3. make snake to move in directions when left, right, down and up key is pressed
4. when snake hits fruit, increases the score by 10.
5. If snake hits 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 position, colour and movement
4. create a snake class which fruit position and colour
5. Create a function to check if snake collides with fruit and increases
6. create function of check snake collides window and end the game.
7. Create a function to update snake position based to user input
8. Create function update up to game display, snake position and check for collisions
9. End the game if user quits or snake collides with the window.

Program:-

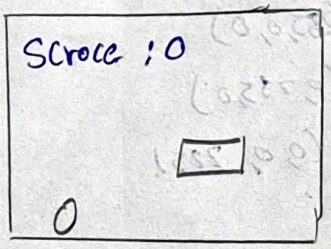
```
#importing libraries.
    import pygame
    import time
    import random
    Snake_Speed = 15
```

```
# defining colours
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)

# initializing pygame.
pygame.init()

# initialise game window
pygame.display.set_Captaion(Greeks for Greeks, Snakes)
game.window = pygame.display.set_mode(window_x, window_y))
# FPS (Frames Per second) controller.
fps = Pygame.time.clock()
# defining snake default position.
snake = pygame.time.clock()
# 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, colour, font, size).
```

Output:



```
# create the displays object.
# score_surface
# score_surface = score_font.render('Score: ' + str(score), True, colour)

# create a rectangular object for text.
# surface object.
score_rect = score_surface.get_rect()

# displaying text
# surface object.
def game_over():

# creating front 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)
# after 2 seconds we will quit program.
time.sleep(2)
# deactivating pygame library.
pygame.quit()
# quit program.
quit()

# main function.
while True:
    # handling key events.
    for event in pygame.event.get():
        if event.key == pygame.KEYDOWN:
        if event.key = pygame.K_UP:
            change_to = 'UP'
        if event.key == pygame.K_DOWN:
            change_to = 'LEFT'
        if event.key == pygame.K_RIGHT: 'RIGHT':
            change_to = 'RIGHT'
```

Problem 2:- Write a python program to Develop a chess board Using pygame.

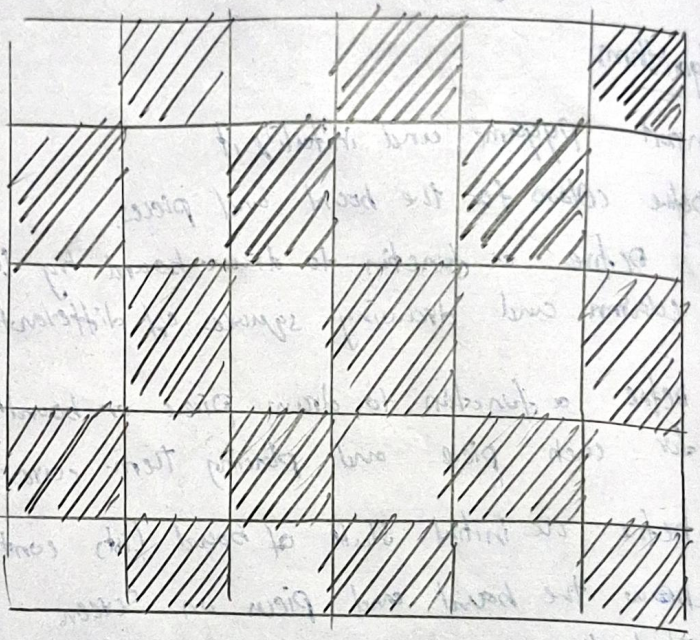
Algorithm:

1. import pygame and initialize it
2. Define colours for the board and pieces.
   Define a function to draw board by looping over rows & column and drawing squares of different colour.
3. Define a function to draw piece on board by loading images for each piece and placing them corresponding square.
4. Define the initial state of board (list) containing the pieces.
5. Draw the board and pieces on screen.
6. 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 colours
    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_colour = white (row+col)%2 == 0 else brown
    square_rect = pygame.Rect(col* 80, row* 80, 80, 80)
```

output



```
# Define function draw the piece.
    def -draw - pieces (board):
      piece - images = {
      'r' = Pygame. image. load (images / rocks . png");
      'n' = pygame, image . load (images / knight . png).
      'b' = pygame image . load ("images / bishop . png);
      'q' = pygame image . load (images / king png).
      k = pygame image load (image, pawn . png).
    }
    for row in range (8):
    for col in range (8):
      if  piece ! =! --?
      piece -image = piece - images (piece)
      piece - rect = pygame . Rect (col*, 80, row* 80, 80, 80)
# Define intial state of board.
    board [.
      ['r' 'n' 'b' 'k' 'b' 'n' 'r']
      ['p' 'p' 'p' 'p' 'p' 'p' 'p']
      [ '.' , '.' , '.' , '.' , '.' , '.']
      [ '.' , '.' , '.' , '.' , '.' , '.')
      [ '.' , '.' , '.' , '.' , '.' , '.')
      [ '.' , '.' , '.' , '.' , '.' , '.')
      ('P' 'P' 'P' 'P' 'P' 'P' 'P' 'P')
      ('R' 'N' 'B' 'Q' 'K' 'B' 'N' 'R')
    ]
# start game loop
    while True:
      for event in pygame . event get().
      if event type == pygame . quit,
        pygame quit ()
        quit ()
      pygame - display update()
```

completed

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

VEL TECH - CSE	
EX NO.	21
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	20
SIGN WITH DATE	