

TASK 121 Simulate Gaming concepts using Pygame

AIM: to simulate gaming concepts using Pygame

Snake game:

Problem 1: write a Python Program to create a snake game using Pygame package.

Conditions:

1. Set the window size
2. create a snake
3. make the snakes to move in the directions when left, right down and up key is pressed.
4. when the snake hit 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 and color.
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 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.

Sample output:

score:10

Output:

score:0

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	



Program:

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

snake_speed = 15

# window size
window_x = 710
window_y = 480

# defining colors

black = pygame.Color(0, 0, 0)
black
white = pygame.Color(255, 255, 255)
red = pygame.Color(225, 0, 0)
green = pygame.Color(0, 255, 0)
blue = pygame.Color(0, 0, 225)

# initialise game window
pygame.display.set_caption('1 blocks for green 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_position = [100, 50]

# 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 - shown = 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):

creating font object score_font

score_font = pygame.font.SysFont (font, size)

create the display surface object

score_surface

score_surface = score_font.render ('score: ' + str (score), True, color)

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)

blt will draw the text on screen

game_window.blit(game_over_surface, game_over_rect)
pygame.display.flip()

after 2 seconds we will quit the program
time.sleep(2)

deactivating Pygame library

pygame.quit()

quit the program

quit()

main function

while True:

handling key events

for event in pygame.event.get():

if event.type == pygame.KEY_DOWN:

if event.key == pygame.K_UP:

change_to = 'up'

if event.key == pygame.K_DOWN:

~~direction~~

change_to = 'DOWN'

if event.key == pygame.K_LEFT:

change_to = 'LEFT'

if event.key == pygame.K_RIGHT:

change_to = 'RIGHT'

moving the snake

if direction == 'up':

snake_position[1] -= 10

if direction == 'down':

snake_position[0] += 10

if direction == 'RIGHT':

snake_position[0] += 10

snake body growing mechanisms

if fruit and snake collide then scores

will be incremented by 10.

snake_body.insert(0, list(snake_position))

else:

snake_body.pop()

if not fruit_spawn:

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))

Game over conditions

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()

touching the snake body

for block in snake_body[1:]:

if snake_position[0] == block[0] and snake_position[1] == block[1]:

game_over

Refresh game screen

pygame.display.update()

frame per second (Refresh rate)

fps.tick(snake_speed)

Problem 2 - Write a Python program to develop a chess board using Py game

Algorithm:

1. Import Py game and initialize it
2. Set screen size and title
3. Define colors for the board and pieces.
Define a function to draw the boards by looping over rows and columns and drawing squares to 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 as a list of list containing the pieces
6. Draw the board and pieces on the screen.
7. Start the game loop.

Program:

```
import pygame
# initialize Py game
pygame.init()

# Set the screen size and title
screen_size = (640, 640)
screen = pygame.display.set_mode(screen_size)
pygame.display.set_caption('Chess Board')

# Define colors
black = (0, 0, 0)
white = (255, 255, 255)
brown = (153, 98, 0)

# Define function to draw the board.
```

sample output:

A 10x10 grid of squares, alternating between white and dark gray shading. The grid is slightly tilted and has some faint, illegible text visible in the background.

square - color = white if (row+col) % 2 == 0 else black

square - rect = pygame.Rect(col * 80, row * 80, 80, 80)

~~pygame.Rect = pygame.Rect (col * 80, row~~

pygame.draw.rect (screen, square-color, square-rect)

define function to draw the pieces

'r' = pygame.image.load('images/king.png')

'n' = pygame.image.load('images/knight.png')

'b' = pygame.image.load('images/^{bishop}~~queen~~.png')

'q' = 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'],

~~['r', 'n', 'b', 'q', 'b', 'n', 'r']~~

]

[illegible]

C

draw board and pieces

draw_board()

draw_pieces(board)

start game loop

while True:

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

pygame.display.update()

VEL TECH - CSE	
EX NO.	12
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
TOTAL (20)	15
SIGN WITH DATE	

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