

5. To simulate game concepts using Pygame

Pygame

Aim: To simulate game concept using Pygame

problem 1. write a program to create a snake game

conditions:

1. set the window size

2. create a snake

3. make the snake to move in the directions

4. when the snake hits

5. if the snake hits the window frame

6. if the snake hits itself

Algorithm

1. Import pygame package and initialize

2. Define the window size and title

3. Create a snake class which initializes the snake position

4. Create a fruit class which initializes

5. Create a function to check if the snake collides with the fruit

6. Create a function to check if the snake collides with itself

Program

```
import pygame
```

```
import time
```

```
import random
```

```
snake_speed = 15
```

```
window_width = 720
```

```
window_height = 480
```

```
black = pygame.color(0, 0, 0)
```

white =

red = P

green

blue

pygar

pygame

game -

SPS = P

snake

snak

fruit

fruit

Bite

char

scor

deR

scor

scor

scor

ga

python game package contains
 snake.py which is the main file.
 snake.py has 3 classes which are
 Snake, Food and Score.
 Snake class has methods like
 move(), eat(), grow() etc.
 Food class has a method called
 get_random_pos().
 Score class has a method called
 add_score().
 In the main file, we have
 a while loop which runs until
 the game is over. Inside the loop,
 we have the following code:

```

    if collision_with_boundaries or collision_with_body:
        game_over()
    if collision_with_food:
        eat()
        grow()
        update_score()
    move()
  
```

 The collision detection is done
 by checking if the head of the snake
 is at the same position as the food.
 If it is, then the eat() method is
 called which increases the length of
 the snake and the grow() method
 is called which places a new food
 item at a random position.
 The update_score() method is called
 which increments the score by 1.
 The move() method is called which
 moves the snake in the direction
 specified by the user's input.
 The game_over() method is called
 when the snake collides with
 its own body or the boundaries
 of the screen. It prints "Game Over!"
 and then exits the program.
 The main file also imports
 the pygame module and initializes
 the display.
 The display is set to 800x600 pixels
 and the background color is white.
 The snake is represented by a list
 of segments and each segment is
 a rectangle of size 20x20 pixels.
 The food is represented by a single
 rectangle of size 20x20 pixels.
 The score is represented by a
 single integer value.
 The game loop runs until the user
 quits the program.

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('snake game')

game_window = pygame. Display. set_mode

fps = pygame. time. clock()

snake_position = [100, 50]

snake_body = [[100, 50],
[90, 50],
[80, 50]
[70, 50]]

fruit_position = [random.randrange(1, window_size // 10) * 10,
random.randrange(1, window_size // 10) * 10]

fruit_spawn = True

direction = 'RIGHT'

change_to = direction

score = 0

def show_score(choice, color, font, size):

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

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

score_rect = score_surface.get_rect()

game_window.blit(score_surface, score_rect)

def game_over():

my_font = pygame.Font.SysFont('times new roman', 30)

game_over_surface = my_font.render('your score is: ' + str(score), True, red)

game_over_rect = game_over_surface.get_rect()

game_over_rect = game_over_surface.get_rect()

game_over_rect.midtop = (window_x/2, window_y)

game_window.blit(game_over_surface, game_over_rect)

time.sleep(2)

pygame.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'

If change-to == 'RIGHT' and direction != 'LEFT':
 direction = 'RIGHT'

If direction == 'UP':

 snake-position[i] - = 10

If direction == 'DOWN':

 snake-position[i] + = 10

If direction == 'LEFT':

 snake-position[0] - = 10

If direction == 'RIGHT':

 snake-position[0] + = 10

snake-body.insert(0, list(snake-position))

score + = 10

fruit-spawn = False

else snake-body.pop()

If not fruit-spawn:

 fruit-position = {random.randrange(1, (window-1)/10)}

fruit-spawn = True

game-window.fill(black)

for pos in snake-body:

 Pygame.draw.rect(game-window, green, Pygame

 Rect(pos[0], pos[1])

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

Show-score(), white, 'times new roman', 20

Pygame.display.update()

(snake-speed)

Chloro-Dinitro-Benzoic Acid (1)

Chloro-Dinitro-Benzoic Acid (2)

Dinitro-Benzoic Acid - Dinitro-Benzoic Acid

Group (Bar 1) 100 g

Bar 1	Bar 2	Bar 3	Bar 4	Bar 5	Bar 6	Bar 7	Bar 8	Bar 9	Bar 10	Bar 11	Bar 12	Bar 13	Bar 14	Bar 15	Bar 16	Bar 17	Bar 18	Bar 19	Bar 20
Chloro-Dinitro-Benzoic Acid	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Dinitro-Benzoic Acid	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Chloro-Dinitro-Benzoic Acid	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Dinitro-Benzoic Acid	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Chloro-Dinitro-Benzoic Acid	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Dinitro-Benzoic Acid	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Chloro-Dinitro-Benzoic Acid	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Dinitro-Benzoic Acid	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Chloro-Dinitro-Benzoic Acid	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Dinitro-Benzoic Acid	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Chloro-Dinitro-Benzoic Acid	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Dinitro-Benzoic Acid	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Chloro-Dinitro-Benzoic Acid	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Dinitro-Benzoic Acid	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Chloro-Dinitro-Benzoic Acid	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Dinitro-Benzoic Acid	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Chloro-Dinitro-Benzoic Acid	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Dinitro-Benzoic Acid	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Disposal: Dissolve in water, pour into a sink.

Disposal: Dissolve in water, pour into a sink.

Disposal: Dissolve in water, pour into a sink.

Storage: Store in a dry place.

Storage: Store in a dry place.

problem 2. write a Python program to develop a chess board using Pygame

Algorithm

1. Import Pygame and initialize
2. Set screen size and title
3. Define colors for the board and pieces
4. Draw the board and piece on the screen
5. 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):
```

```
        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)
```

```
def draw_piece(board):
    piece_images = [
        pygame.image.load('images/rock.png'),
        " " (images/knight.png),
        " " (images/bishop.png),
        " " (images/king.png),
        " " (images/queen.png)
    ]
```

```
for row in range(8):
    for col in range(8):
        piece = board[row][col]
        if piece == '':
            piece_image = piece_images[0]
        else:
            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],
    [P, P, P, P, P, P, P, P]
]
```

```
[R, N, B, Q, K, B, N, R],
[P, P, P, P, P, P, P, P],
[P, P, P, P, P, P, P, P]
```

```
draw_board()
draw_pieces(board)
```

```
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
```

Completed

VRL TECH	
EX NO.	✓
PERFORMANCE (5)	✓
RESULT AND ANALYSIS (5)	✓
VIVA VOCE (5)	✓
RECORD (5)	✓
TOTAL (20)	✓
WITH DATE	✓

Result: Hence writing a Python program
to develop a chess Board using
Python program is done successfully