Task - 12 Simulate Gaming Concepts Using Pygame

AIM :-

To simulate Gaming Concepts using pygame

Snake Game :

Problem 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 direction when left, right, down, up key is pressed.
4. When the snake by 10
5. If the snake units the window Game over.

ALGORITHM :-

1. Import pyame package and initialize it.
2. Define the window size and title
3. Create a snake class which initialize the snake position, colour, and movement
4. Create a fruit class which initialize 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 and the game

display and draw the snake and fruit.

9. Create a game loop to countinwously update the game display, snake collides with the window

PROGRAM :-

```
# importing libraries
import pygame
import time
import random
  Snake_Speed = 15
window_x = 720
window_y = 480
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)
Pygame.fruit ()
Pygame.display.set_caption ('Geeksfor Geeks Snakes')
game_window = pygame.display.set_mode
    ((window_x, window_y))
    fps = pygame.time.clock()
    Snake_position = [100,500]
Snake_body =[[100,50],
    [90,50]
    [80,50],
    [70,50],
    ]
fruit_position = [random ran orange.
```

```
fruit_Spawn = True
direction = 'RIGHT'
change_to = direction
score : 0
def show_score (choice, colour, fount, size):
score_font = pygame.font.SysFont (font, size)
score_surface = score_font.render
['Score:' + str (score), True, color) Score
Score_rect = score_surface.get.rect ()
game_window.built (score_surface, score_rect)
def game_over ():
wry font = pygame.font.SysFont( times new
                                roman, 50)
game-over_surface = my-font_render (
your score is:
game_over_rect.midtop = (window = x/2, window -4/4)
game_window blit (game_over_surface, game_
over_rect)
pygame.display.flip ()
time.sleep (2)
pygame.quit ()
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 direction == 'UP':
    Snake_Position [1] == 10
if direction = 'DOWN'
    Snake_Position [1] += 10
if direction = 'LEFT'
    Snake = position [0] -=10
if direction = = 'RIGHT';
    Snake_position[0] += 10
Snake_body : insert 10, list [Snake_position))
if snake_position [0] == fruit_Position [0] and
Snake_position [1] == fruit_position [1];
    score += 10
    fruit_Spawn = False
else:
    Snake_body.pop ()
    if not fruit_Spawn:
        fruit_position = [random.randrange[1,
(window = x//10)) * 10, random. randrange (1,(window = 4 // 10))
        fruit_spawn = True
        game_window. fiy(black)
        for Pos in Snake.body.
        pygame. Rect (pos[0], pos[1], 10, 10))
        pygame. display. update ()
        fps. tick (Snake_Speed)
```

Output :-

Score : 0

12(b) write a python program to develop a 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. outline colours for the board and pieces
5. outline the initial state of the board as a list of lists Counting the piece
6. draw the board and start the gran loop

PROGRAM :-

```
import pygame
pygame.init()
screen_size = (640, 640)
screen = pygame.displays.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 - colour = white if (row+col]%.2== 0 else
brown
        pygame.draw.rect (screen, square_colour, square_
rect)
```

```
def draw_pieces(board):
    piece_images={
'r':pygame.image.load('images/rook.pnd'),
'n':pygame.image.load('images/knight.png')
'b':pyame.image.load('images/bishop.png'),
}
for row in range e(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.bit(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']
]
draw_board()
draw_pieces(board)
while.True:
    for event in pygame.event.get():
pygame.quit()
  quit()
pygame.display.update()
```

PERFORMANCE (5)	
RESULT AND ANALYSIS (5)	
VIVA VOCE (5)	
RECORD (5)	
TOTAL (20)	
SIGN WITH DATE	

Result:

Thus the program for pygame is executed and verified successfully.

Output