A

Project on

“SNAKE GAME USING PYTHON”

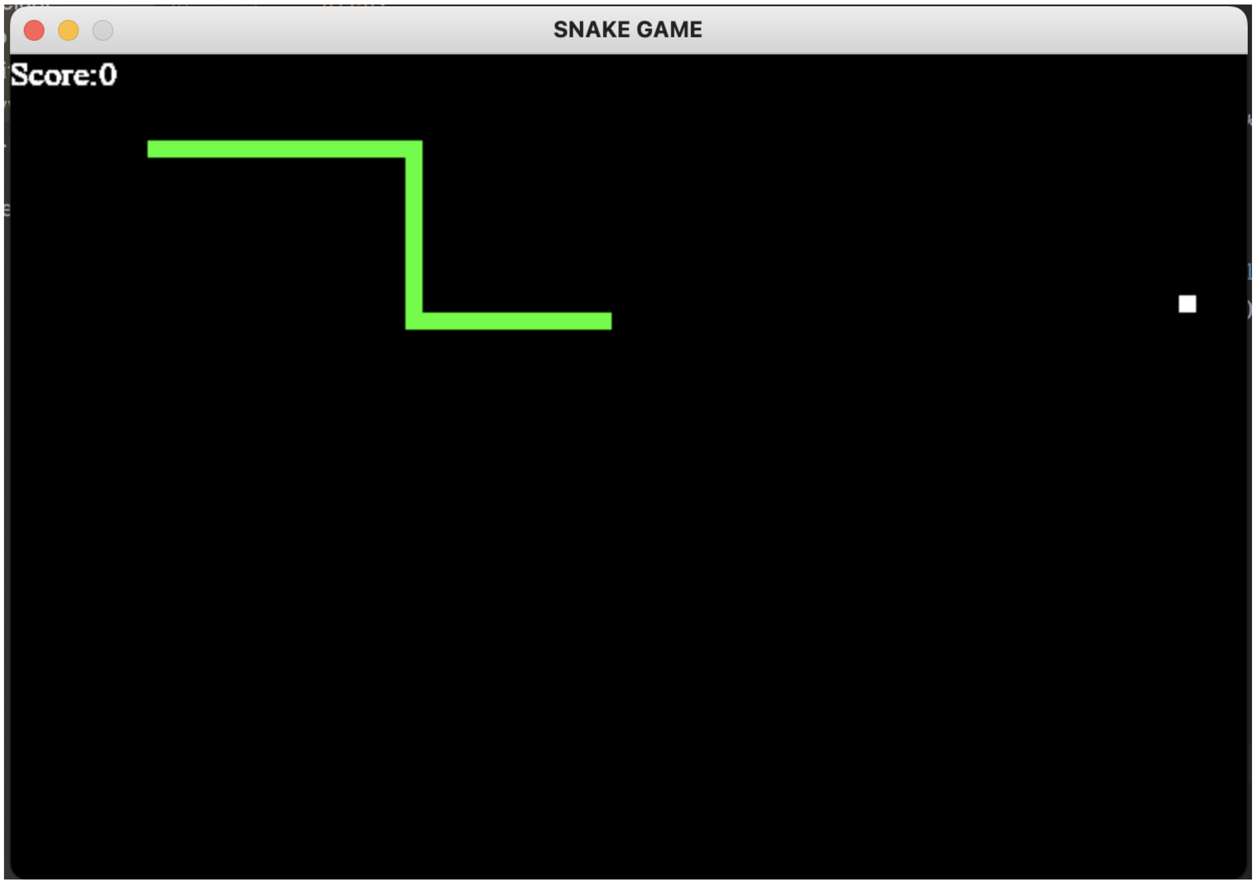
Introduction

The Snake Game, a classic arcade-style game, has been entertaining players for decades. Now, armed with the power of Python, you have the opportunity to create your own version of this addictive masterpiece. Python, with its simplicity and versatility, is the perfect language to bring the Snake Game to life.
Python provides a wide range of libraries and tools that make game development accessible and enjoyable. You'll utilize libraries like Pygame or Turtle Graphics to handle graphics and user input, allowing you to focus on the core gameplay mechanics. With Python's simplicity and readable syntax, you'll be able to understand and modify the game's logic with ease.

Source code

```python
import pygame
import time
import random
ss=10
wx=720
wy=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.init()
pygame.display.set_caption('SNAKE GAME')
game_window=pygame.display.set_mode((wx,wy))
fps=pygame.time.Clock()
sp=[100,50]
sb=[[100,50],[90,50],[80,50],[70,50]]
fp=[random.randrange(1,(wx//10))*10,random.randrange(1,(wy//10))*10]
fs=True
d='RIGHT'
ct=d
score=0
def showscore(choice,color,font,size):
 sf=pygame.font.SysFont(font,size)
 ss=sf.render('Score:'+str(score),True,color)
 sr=ss.get_rect()
 game_window.blit(ss,sr)
def game_over():
   mf=pygame.font.SysFont('times new roman',50)
   gos=mf.render('Your Score:'+str(score),True,red)
   gor=gos.get_rect()
   gor.midtop=(wx/2,wy/4)
   game_window.blit(gos,gor)
   pygame.display.flip()
   time.sleep(2)
   pygame.quit()
   quit()
#main fun
while True:
   for event in pygame.event.get():
       if event.type==pygame.KEYDOWN:
           if event.key==pygame.K_UP:
               ct='UP'
           if event.key==pygame.K_DOWN:
               ct='DOWN'
           if event.key==pygame.K_LEFT:
               ct='LEFT'
           if event.key==pygame.K_RIGHT:
               ct='RIGHT'
   if ct=='UP' and d!='DOWN':
       d='UP'
   if ct=='DOWN' and d!='UP':
       d='DOWN'
   if ct=='LEFT' and d!='RIGHT':
       d='LEFT'
   if ct=='RIGHT' and d!='LEFT':
       d='RIGHT'
   if d=='UP':
      sp[1]-=10
   if d=='DOWN':
      sp[1]+=10
   if d=='LEFT':
     sp[0]-=10
   if d=='RIGHT':
     sp[0]+=10
   sb.insert(0,list(sp))
   if sp[0]==fp[0] and sp[1]==fp[1]:
      score+=10
      fs=False
   else:
      sb.pop()
   if not fs:
        fp=[random.randrange(1,(wx//10))*10,random.randrange(1,(wy//10))*10]
        fs=True
        game_window.fill(black)
    for pos in sb:
        pygame.draw.rect(game_window,green,pygame.Rect(pos[0],pos[1],10,10))
        pygame.draw.rect(game_window,white,pygame.Rect(fp[0],fp[1],10,10))
    if sp[0]<0 or sp[0]>wx-10:
        game_over()
    if sp[1]<0 or sp[1]>wy-10:
        game_over()
    for block in sb[1:]:
        if sp[0]==block[0] and sp[1]==block[1]:
            game_over()
    showscore(1,white,'times new roman',20)
    pygame.display.update()
    fps.tick(ss)
```

Output



Explanation

1. Importing libraries: The necessary libraries are imported, including pygame, time, and random.
2. Initializing variables: Several variables are initialized, such as the screen width and height (wx and wy), colors (black, white, red, green, blue), and the initial position of the snake (sp). The snake body is represented by a list of segments (sb), and the food position is stored in fp.
3. Setting up the game window: The Pygame module is initialized, and the game window is created with the specified dimensions (wx and wy).
4. Defining helper functions: Two helper functions are defined. showscore is responsible for displaying the score on the game window, and game_over displays the final score, waits for a couple of seconds, and quits the game.
5. Game loop: The main game loop starts, which runs continuously until the game is exited. It handles user input events and updates the game state accordingly.
6. Handling user input: The game checks for keyboard input events and changes the direction of the snake (ct) based on the pressed arrow key.
7. Updating snake direction: The snake's direction (d) is updated based on the current direction (ct) and makes sure that the snake cannot reverse its direction instantly.
8. Updating snake position: The snake's position is updated based on the current direction (d). The snake's head (sp) is moved accordingly, and a new segment is inserted at the beginning of the snake's body (sb).
9. Handling food consumption: If the snake's head reaches the food position (fp), the score is increased, and a new food position is generated randomly. Otherwise, the last segment of the snake's body is removed (sb.pop()).
10. Checking game over conditions: The game checks for various game over conditions. If the snake hits the boundaries of the game window or collides with its own body, the game_over function is called to end the game.
11. Drawing game elements: The game window is filled with the background color. The snake's body segments and the food item are drawn on the screen using rectangles.
12. Displaying score: The showscore function is called to display the score on the game window.
13. Updating the display: The changes made to the game window are updated on the screen using pygame.display.update().
14. Setting the game's FPS: The frame rate of the game is controlled using fps.tick(ss) to ensure smooth gameplay.